Database Administration Level III

Based on August, 2011, Version 3 Occupational Standards (OS) and curriculum

Module Title : - Using Basic Structured Query Language

LG Code: - EIS DBA3 M09 1220 LO (1-4) LG (33-36)

TTLM Code: - EIS DBA3 TTLM09 12 20 v1

December 2020

Bishoftu, Ethiopia

Contents

L #33	LO #1- Write an SQL statement to retrieve and sort data

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Using a structured query language to create database structures and store data
- Retrieving all the data from a table.
- Retrieving data from specific columns in a single table.
- Using 'Order by' to sort query output.
- Retrieving restricted number of rows using 'where' clause
- Retrieving restricted number of rows using select statement
- Using comparison operators in the where clause to compare numeric, character, string, date and time data
- Using Boolean operatorswith the correct precedence
- Using Criteria in the 'where' clause to check for a range of values
- Using SQL syntax to suppress duplicate values from query results
- Taking action to exclude null values from a query result

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Retrieve all the data from a table.
- Retrieve data from specific columns in a single table.
- Use 'Order by' to sort query output.
- Retrieve restricted number of rows using 'where' clause
- Retrieve restricted number of rows using select statement
- Use comparison operators in the where clause to compare numeric, character, string, date and time data
- Use Boolean operatorswith the correct precedence

- Use Criteria in the 'where' clause to check for a range of values
- Use SQL syntax to suppress duplicate values from query results
- Take action to exclude null values from a query result

earning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If you earned a satisfactory evaluation proceed to “Operation sheets
6. Perform “the Learning activity performance test” which is placed following “Operation sheets” ,
7. If your performance is satisfactory proceed to the next learning guide,
8. If your performance is unsatisfactory, see your trainer for further instructions or go back to “Operation sheets”.

Information sheet1: using a structured query language to create database structures and store data

1.1 Using a structured query language to create database structures and store data

Introduction

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. The SQL Standard has gone through a lot of changes during the years, which have added a great deal of new functionality to the standard, such as support for XML, triggers, regular expression matching, recursive queries, standardized sequences and much more.

- **Uses of SQL**

SQL is widely popular because it offers the following advantages:-

- ✓ SQL can execute queries against a database
- ✓ SQL can retrieve data from a database
- ✓ SQL can insert records in a database
- ✓ SQL can update records in a database
- ✓ SQL can delete records from a database
- ✓ SQL can create new databases
- ✓ SQL can create new tables in a database
- ✓ SQL can create stored procedures in a database
- ✓ SQL can create views in a database
- ✓ SQL can set permissions on tables, procedures, and views

- **SQL Commands/Categories of SQL function**

  The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their

Table 1: DDL - Data Definition Language

Sr.No.	Command & Description
1	CREATE:-Creates a new table, a view of a table, or other object in the database.
2	ALTER:-Modifies an existing database object, such as a table.
3	DROP:-Deletes an entire table, a view of a table or other objects in the database.

Table2: DML - Data Manipulation Language

Sr.No.	Command & Description
1	SELECT:-Retrieves certain records from one or more tables.
2	INSERT:-Creates a record.
3	UPDATE:-Modifies records.
4	DELETE:-Deletes records.

Table 3: DCL - Data Control Language

Sr.No.	Command & Description
1	GRANT:-Gives a privilege to user.
2	REVOKE:-Takes back privileges granted from user.

Data Definition Language (DDL)

The DDL manages table and index structure. Specification notation for defining the database schema. DDL compiler generates a set of tables stored in a data dictionary and Data dictionary contains metadata (data about data). DDL statements are used to define the database structure or schema.

Data storage and definition language – special type of DDL in which the storage structure and access methods used by the database system are specified.

The most basic items of DDL are the CREATE, ALTER, RENAME and DROP statements:

- CREATE creates an object (a table, for example) in the database.
- DROP deletes an object in the database, usually irretrievably.
- ALTER modifies the structure an existing object in various ways—for example, adding a column to an existing table
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Data Manipulation Language (DML)

The DML Language for accessing and manipulating the data organized by the appropriate data model. It is the subset of SQL used to add, update and delete data and statements are used for managing data within schema objects.

The acronym CRUD refers to all of the major functions that need to be implemented in a relational database application to consider it complete.

Each letter in the acronym can be mapped to a standard SQL statement:

Table 4

Operation	SQL	Description
Create	INSERT INTO	inserts new data into a database
Read (Retrieve)	SELECT	extracts data from a database
Update	UPDATE	updates data in a database
Delete (Destroy)	DELETE	deletes data from a database
	MERGE	UPSERT operation (insert or update)
	CALL	call a PL/SQL or Java subprogram
	EXPLAIN PLAN	explain access path to data
	LOCK TABLE	control concurrency

Data Control Language (DCL)

DCL is a subset of the Structured Query Language (SQL) that allows database administrators to configure security access to relational databases. Some examples:

- GRANT: -to allow specified users to perform specified tasks
- REVOKE: - to cancel previously granted or denied permissions.
- DENY:-maybe used to explicitly prevent a user from receiving a particular permission.

This is helpful when a user may be a member of a role or group that is granted permission and you want to prevent that user from inheriting the permission by creating an exception.

Transaction Control Language (TCL)

TCL are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions. Used to control transactional processing in a database.

- COMMIT-save work done. To apply the transaction by saving the database changes
- SAVEPOINT- identifies a point (breakpoints) for a transaction to which you can later rollback or to allow partial rollbacks.
- ROLLBACK - restore database to original since the last COMMIT or to undo all changes of a transaction
- SET TRANSACTION - used to specify characteristics for the transaction that follows. For example, you can specify a transaction to be read only or read write.
- RELEASE SAVEPOINT–used to remove a SAVEPOINT that you have created.

Hardware Requirements

A processor with high speed of data processing and memory of large size (RAM and Hard disk space) is required to run the DBMS software.

Example: The Minimum hardware requirements for SQL Server 2008 installations include:

- Processor
    - ✓ Pentium 600 MHz or higher is required

- ✓ 1 GHz or higher is recommended.
- Memory
  - ✓ 512 MB is required
  - ✓ 1 GB or more is recommended
  - ✓ Express Edition requires 192 MB and recommended 512 MB or more.
- Disk space
  - ✓ Database components: 280 MB
  - ✓ Analysis services: 90 MB
  - ✓ Reporting services: 120 MB
  - ✓ Integration services: 120 MB
  - ✓ Client components: 850 MB

Operating System Requirements

The supported operating system for DBMS software may depends on the type and version of the software.

Example: The Supported operating systems for SQL Server 2008 installations include:

- Windows 7
- Windows Server 2003 Service Pack 2
- Windows Server 2008
- Windows Server 2008 R2
- Windows Vista Service Pack 1
- Windows XP Service Pack 3

DBMS Terminologies and Definitions

A database management system is the software that enables users to define, create, and maintain the database and also provides controlled access to this database.

Some of the most common database applications are:

- Microsoft Access
- Microsoft SQL
- Oracle, and Informix

Categories of SQL Application

Microsoft SQL Server is a powerful and reliable data management system that delivers a rich set of features, data protection, and light Web applications.

- Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, MS- Access, etc.
- SQL Application is a database language that allow a user to:
  - ✓ Define (create) the database, table structures, and controlling access to the data using DDL.
  - ✓ Perform both simple and complex queries using DML.

Install DBMS (SQL Server)

Download SQL Server 2008 Express Edition with tools (around 230 MB) and run the setup file.

If you are using windows XP SP3

Remark: Microsoft regularly updates *SQL Express*, so screens could change a little over time.

1. Choose between 32 bits version (XP, Vista) or 64 Bits version (Seven)
   Launch installer : the installer extracted the data
2. SQL Server installation Centre opens.
   Click on "System configuration Checker" to check for conditions that prevents a successful SQL Server Installation
3. Setup Support Rules opens. Click on "show details to ensure that all the tests passed successfully
   Then click on OK
   - If you have some tests failures
   - click on the test to have more information about the error
   - Ensure you have uninstalled correctly all previous versions of SQL Server (2000, 2005) and rebooted your computer before
   - Refer to online documentation on Microsoft SQL Server website
   - refer to Octopus users forum

4. Go Back to SQL Server installation Centre
   On the left column, click on "Installation"
   Then click on the first link : "new SQL Server stand-alone installation" (Unless you want to make a specific network installation)
5. Setup support rules reopen.
   Once all operations completed, click OK
6. The SQL Server 2008 Setup opens with a first "product key" tab
   as we use a specific free version of SQL Server, no activation key is needed.
   Simply click "next"
7. Read the license terms
   Check the box "I accept the license terms"
   Click on "Next"
8. The "setup support files" tab opens.
   Click on "install"
9. Once installed, the Setup Support Rules reopen.
   Check that all operation passed, then click "next"
10. On the "features selection" tab choose "Data Engine Services" and "Management tools" (this is the Management Studio Express)
    check the features directory (no need to change normally) click "next"
11. On the "instance Configuration" check that the parameters are like the screenshot on the left (nothing to change normally)
    Click "Next"
12. Disk space requirements tap
    - Click Next
13. Server Configuration:

For the first service "SQL Server Database Engine", on the account name, choose the first result of the list box (in our case NT AUTHORITY\NETWORK)

Then click "Next".

Database Engine Configuration:

- Chose "Mixed mode"

- Enter the password octopus
- Specify SQL Server administrator : Click on "add current user"
- Click "Next"

14. Error and Usage Reporting
    - Click next
15. Installation Rules :

    Once Operation completed, click "Next"
16. Ready to install:

    Click "Install"

    Installation Progress
    - Wait until Installation is finished
    - Ensure "Database Engine Services" and "Management Tools Basic" are successfully installed
    - Click "Next"
17. Complete:

    Once installed, click on "Close"
18. To ensure SQL Express is running correctly, run "SQL Server Configuration Manager"

    On the left column click on "SQL Server Services

    On the right column, ensure that SQL Server (SQLEXPRESS) is running correctly (otherwise right-click on the device to start it)
19. Browse the new directory "Microsoft SQL Server 2008" and open the SQL Server Management Studio to ensure it has been correctly installed
20. 22. If you used to work with SQL Server 2000 our SQL Server 2005 before, you will probably obtain this message

    "Do you want to import your customized user settings...". Simply click Yes (or no as you prefer it doesn't matters concerning Octopus)
21. The SQL Server Connection Windows appears.

    Choose the following parameters:
    - Server Type: Database Engine

- Server Name: (Name of your computer)\SQLEXPRESS*
- Authentication: SQL Server Authentication (if you can only see windows authentication, you probably missed to choose "Mixed Mode" at the Database Engine Configuration step
- Login: sa (by default)
- Password: the password you defined at the Database Engine Configuration step (octopus in our case)

22.* You can find the name of your computer on the control panel>System Does it opens? -> Congratulations, you can now go to the next step: Install Octopus

Use 'Order by' to sort query output.

Create Database

The SQL CREATE DATABASE statement is used to create new SQL database.

Syntax: Basic syntax of CREATE DATABASE statement is as follows:

CREATE DATABASE DatabaseName;

Always database name should be unique within the RDBMS.

Example:

If you want to create new database <testDB>, then CREATE DATABASE statement would be as follows:

SQL> CREATE DATABASE testDB;

Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases as follows:

MySQL> SHOW DATABASES; Or Ms SQL> select * from sys.DATABASES

Table 5

Database
information_schema
AMROOD
TUTORIALSPOINT
mysql
orig
test
testDB

7 rows in set (0.00 sec)

Drop Databases

he SQL DROP DATABASE statement is used to drop an existing database in SQL schema.

Syntax: Basic syntax of DROP DATABASE statement is as follows:

DROP DATABASE DatabaseName;

Always database name should be unique within the RDBMS.

Example: If you want to delete an existing database <testDB>, then DROP DATABASE statement would be as follows:

SQL> DROP DATABASE testDB;

NOTE: Be careful before using this operation because by deleting an existing database would result in loss of complete information stored in the database.

Make sure you have admin privilege before dropping any database. Once a database is dropped, you can check it in the list of databases as follows:

Ms SQL> select * from sys.DATABASES

Table 6

Database
information_schema
AMROOD
TUTORIALSPOINT
mysql
|orig
test

6 rows in set (0.00 sec)

Creating Tables

Creating a basic table involves naming the table and defining its columns and each column's data type. The SQL CREATE TABLE statement is used to create a new table.

Syntax: Basic syntax of CREATE TABLE statement is as follows:

CREATE TABLE table_name(

column1datatype,

column2datatype,

column3datatype,

columnNdatatype,

PRIMARY KEY(one or more columns));

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with an example below.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

Example: Following is an example, which creates a CUSTOMERS table with ID as primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table:

SQL> CREATE TABLE CUSTOMERS (ID   INT   NOT NULL, NAME VARCHAR (20) NOT NULL,AGE INT   NOT NULL, ADDRESS CHAR (25) ,SALARY   DECIMAL (18, 2), PRIMARY KEY (ID));

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use DESC command as follows: CUSTOMERS; in Ms SQL

Table 7

Field	Type	Null	Key Default	Extra
Id	Int(11)	No	Pri	
Name	Varchar(20)	No		
Age	Int(11)	No		
Address	Char(25)	Yes	Null	
Salary	Decimal(18,2)	Yes	Null	

5 rows in set (0.00 sec)

Now, you have CUSTOMERS table available in your database which you can use to store required information related to customers.

You can check complete details at Create Table Using another Table.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement. The new table has the same column definitions. All columns or specific columns can be selected. When you create a new table using existing table, new table would be populated using existing values in the old table.

Syntax: The basic syntax for creating a table from another table is as follows:

CREATE TABLE NEW_TABLE_NAME AS

SELECT [column1, column2...columnN] FROM EXISTING_TABLE_NAME [WHERE]

Here, column1, column2...are the fields of existing table and same would be used to create fields of new table.

Example: Following is an example, which would create a table SALARY using CUSTOMERS table and having field's customer ID and customer SALARY:

```
SELECT ID, SALARY
FROM CUSTOMERS;
```

This would create new table SALARY, which would have the following records:

Table 8

ID	SALARY
1	2000.00
2	1500.00
3	2000.00
4	6500.00
5	8500.00
6	4500.00
7	10000.00

Drop Table

The SQL DROP TABLE statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.

NOTE: You have to be careful while using this command because once a table is deleted then all the information available in the table would also be lost forever.

Syntax: Basic syntax of DROP TABLE statement is as follows:

DROP TABLE table_name;

Eg drop table customer;

Example: Let us first verify CUSTOMERS table and then we would delete it from the database:

SQL> DESC CUSTOMERS; or Sp_help CUSTOMERS; in Ms SQL\

Table 9

Field	Type	Null	Key Default	Extra
ID	int(11)	NO	PRI	
NAME	varchar(20)	NO		
AGE	int(11)	NO		
ADDRESS	char(25)	YES	NULL	
SALARY	decimal(18,2)	YES	NULL	

5 rows in set (0.00 sec)

This means CUSTOMERS table is available in the database, so let us drop it as follows:

SQL> DROP TABLE CUSTOMERS;

Query OK, 0 rows affected (0.01 sec)

Now, if you would try DESC command, then you would get error as follows:

SQL> DESC CUSTOMERS; or Sp_help CUSTOMERS; in Ms SQL

ERROR 1146 (42S02): Table 'TEST.CUSTOMERS' doesn't exist

Here, TEST is database name which we are using for our examples.

Alter Table

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You would also use ALTER TABLE command to add and drop various constraints on an existing table.

Syntax: The basic syntax of ALTER TABLE to add a new column in an existing table is as follows:

ALTER TABLE table_name ADD column_namedatatype;

The basic syntax of ALTER TABLE to DROP COLUMN in an existing table is as follows:

ALTER TABLE table_name DROP COLUMN column_name;

The basic syntax of ALTER TABLE to change the DATA TYPE of a column in a table is as follows:

ALTER TABLE table_name MODIFY COLUMN column_namedatatype;

The basic syntax of ALTER TABLE to add a NOT NULL constraint to a column in a table is as follows:

ALTER TABLE table_name MODIFY column_namedatatype NOT NULL;

The basic syntax of ALTER TABLE to ADD UNIQUE CONSTRAINT to a table is as follows:ALTER TABLE table_name

ADD CONSTRAINT MyUniqueConstraintUNIQUE(column1, column2...);

The basic syntax of ALTER TABLE to ADD CHECK CONSTRAINT to a table is as follows:

ALTER TABLE table_name

ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

The basic syntax of ALTER TABLE to ADD PRIMARY KEY constraint to a table is as follows:

ALTER TABLE table_name

ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of ALTER TABLE to DROP CONSTRAINT from a table is as follows:

ALTER TABLE table_name

DROP CONSTRAINT MyUniqueConstraint;

If you're using MySQL, the code is as follows:

ALTER TABLE table_name

DROP INDEX MyUniqueConstraint;

The basic syntax of ALTER TABLE to DROP PRIMARY KEY constraint from a table is as follows:

ALTER TABLE table_name

DROP CONSTRAINT MyPrimaryKey;

If you're using MySQL, the code is as follows:

ALTER TABLE table_name

DROP PRIMARY KEY;

Example: Consider the CUSTOMERS table having the following records:

Table 10

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Following is the example to ADD a new column in an existing table:

ALTER TABLE CUSTOMERS ADD SEX char(1);

Now, CUSTOMERS table is changed and following would be output from SELECT statement:

Table 11

Id	Name	Age	Address	Salary	Sex
1	Ramesh	32	Ahmedabad	2000.00	NULL
2	Khilan	25	Delhi	1500.00	NULL
3	kaushik	23	Kota	2000.00	NULL
4	Chaitali	25	Mumbai	6500.00	NULL
5	Hardik	27	Bhopal	8500.00	NULL
6	Komal	22	MP	4500.00	NULL
7	|Muffy	24	Indore	10000.00	NULL

Following is the example to DROP sex column from existing table:

ALTER TABLE CUSTOMERS DROP SEX;

Now, CUSTOMERS table is changed and following would be output from SELECT statement:

Table 12

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Inserting Records

The SQL INSERT INTO Statement is used to add new rows of data to a table in the database.

Syntax: There are two basic syntaxes of INSERT INTO statement as follows:

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)]

VALUES (value1, value2, value3,...valueN);

Here, column1, column2...columnN are the names of the columns in the table into which you want to insert data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table. The SQL Insert Into Syntax Would Be As Follows:

Insert Into Table_Name Values (Value1, Value2, Value3,...Valuen);

Example: Following Statements Would Create Six Records In Customers Table:

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (2, 'Khilan', 25, 'Delhi', 1500.00);

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (3, 'Kaushik', 23, 'Kota', 2000.00);

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (4, 'Chaitali', 25, 'Mumbai', 6500.00);

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (5, 'Hardik', 27, 'Bhopal', 8500.00);

Insert Into Customers (Id, Name, Age, Address, Salary)

Values (6, 'Komal', 22, 'Mp', 4500.00);

You Can Create A Record In Customers Table Using Second Syntax As Follows:

Insert Into Customers

Values (7, 'Muffy', 24, 'Indore', 10000.00);

All The Above Statements Would Produce The Following Records In Customers Table:

Table 13

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Populate one table using another table:

You can place the result set of any query into a new table by using the SELECT INTO statement.

- You can use the SELECT INTO statement to create a table and to insert rows into the table in a single operation
- Use the SELECT INTO statement to populate new tables in a database with imported data from another table.

`Syntax:` SELECT <select_list> INTO <new_table-name>FROM <sources_table_name> WHERE <search_condition>

Example: select fname,lname,ID into special_Table from EMPLOYEE where Gender ='female';

Note: - The SELECT INTO statement selects data from one table and inserts it into a different table.

- The SELECT INTO statement is most often used to create backup copies of tables.

Combining table Expressions

With the help of set operators, the results of individual table expressions can be combined. This type of combination is called UNION. SQL supports other set operators besides the UNION operator.

Here is the complete list:

- UNION
- UNION ALL
- INTERSECT
- INTERSECT ALL
- EXCEPT
- EXCEPT ALL

Combining tables with union

The *UNION* operator is used to combine the result-set of two or more SELECT statements.

- Each SELECT statement within the UNION must have the same number of columns and similar data types.
- Also, the columns in each SELECT statement must be in the same order.

Syntax: SELECT <column_name_list> FROM <table_name> UNION SELECT <column_name_list> FROM <table_name>

Example: select fname,ID from EMPLOYEE union select dname,dnumber from DEPARTEMENT

Note: The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL.

Rules for using UNION

The following rules must be applied to use the UNION operator:

- The SELECT clauses of all relevant table expressions must have the same number of expressions
- Each SELECT statement within the UNION must have similar data types. If this applies, the table expressions are union compatible.
- An ORDER BY clause can be specified only after the last table expression. The sorting is performed on the entire end result; after all intermediate results have been combined.
- The SELECT clauses should not contain DISTINCT because SQL automatically removes duplicate rows when using UNION.

Combining with INTERSECT

INTERSECT returns all rows that are both in the result of query1 and in the result of query2. Duplicate rows are eliminated unless ALL is specified.

If two table expressions are combined with the INTERSECT operator, the end result consists of those rows that appear in the results of both table expressions.

Example: SELECT * FROM EMPLOYEE INTERSECT SELECT * FROM PROJECT;

- Just as with the UNION operator, duplicate rows are automatically removed from the result.

Combining with EXCEPT

EXCEPT returns all rows those are in the result of query1 but not in the result of query2. Again, duplicates are eliminated unless ALL is specified.

If two table expressions are combined with the EXCEPT operator, the end result consists of only the rows that appear in the result of the first table expression but do not appear in the result of the second.

Example: SELECT * FROM EMPLOYEE EXCEPT SELECT * FROM PROJECT;

Just as with the UNION operator, duplicate rows are automatically removed from the result.

Keeping duplicate rows

All previous examples made it clear that duplicate rows are automatically removed from the end result if one of the set operators UNION, INTERSECT, or EXCEPT is used. Removing duplicate rows can be suppressed by using the ALL version of these operators. We illustrate this with the UNION ALL operator.

If two table expressions are combined with the UNION ALL operator, the end result consists of the resulting rows from both of the table expressions. The only difference between UNION and UNION ALL is that when you use UNION, the duplicate rows are automatically removed, and when you use UNION ALL, they are kept.

Set operators and NULL values

SQL automatically removes duplicate rows from the result if the set operators UNION, INTERSECT, and EXCEPT are specified. That is why the following (somewhat peculiar) SELECT statement produces only one row, even if both individual table expressions have one row as their intermediate result:

```
SELECT   PLAYERNO, LEAGUENO FROM    PLAYERS  WHERE    PLAYERNO = 27
UNION
SELECT   PLAYERNO, LEAGUENO FROM    PLAYERS WHERE    PLAYERNO = 27
```

Combining multiple set operators

We have already seen a few examples in which multiple set operators are used within a single SELECT statement. Here is another example.

Get the numbers of each player who incurred at least one penalty and who is not a captain; add the numbers of the players who live in Eltham.

- **Set operator and Theory**

We conclude this chapter with a rather theoretical discussion of set operators. We give a number of rules for working with multiple different set operators within one SELECT statement. All the rules are based on general rules (laws) that apply to mathematical operators and set theory. We define and explain each of these rules, and we use the following symbols and definitions:

- The symbol T_i represents the result of a random table expression (i is 1, 2, or 3).
- For each T_i, it holds that the SELECT clauses are union compatible.
- The symbol TØ represents the empty result of a table expression.
- The symbol ∪ represents the UNION operator.
- The symbol ∩ represents the INTERSECT operator.
- The symbol – represents the EXCEPT operator.
- The symbol $\cup^A$ represents the UNION ALL operator.
- The symbol $\cap^A$ represents the INTERSECT ALL operator.
- The symbol $-^A$ represents the EXCEPT ALL operator.
- The symbol = means “is equal to.”
- The symbol ≠ means “is not always equal to.”
- The symbol θ represents a random set operator.
- The symbol Ø represents an empty result.

Self-Check -1	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. ___________ Statements are used to perform tasks such as update data on a database, or retrieve data from a database.
   A. SQL (Structured Query Language)
   B. SML(Structured Mark-up Language)
   C. SEQUEL. SQL
   D. All
2. What can SQL do?
   A. SQL can execute queries against a database
   B. SQL can retrieve data from a database
   C. SQL can insert records in a database
   D. All
3. What does SQL stand for?
   A. Structured Question Language
   B. Structured queue Language
   C. Structured Query Language
   D. Structured QUOTE Language
4. What language is used in database?
   A. Data Definition Language (DDL)
   B. Data Manipulation Language (DML)
   C. Data Control Language (DCL)
   D. All
5. Which Transaction Control Language is not used to control transactional processing in a database?
   A. COMMIT
   B. SAVEPOINT
   C. GRANT
   D. All

Score = ____________
Rating: _____________

Name: ______________________Date: ________________

Short Answer Questions

Note: **Satisfactory rating - 5 points Unsatisfactory - below 5 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet2: retrieving all the data from a table

1.2 retrieving all the data from a table

Select Statements

SQL **SELECT** statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.

Syntax: The basic syntax of SELECT statement is as follows:

SELECT column1, column2,columnN FROM table_name;

Here, column1, column2...are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

SELECT * FROM table_name;

Example: Consider the CUSTOMERS table having the following records:

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Following is an example, which would fetch ID, Name and Salary fields of the customers available in CUSTOMERS table:

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

This would produce the following result:

Table 2

ID	NAME	SALARY
1	Ramesh	2000.00
2	Khilan	1500.00
3	kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	|Muffy	10000.00

If you want to fetch all the fields of CUSTOMERS table, then use the following query:

SQL> SELECT * FROM CUSTOMERS;

This would produce the following result:

Table 3

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	|Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Data Types

SQL data type is an attribute that specifies type of data of any object. Each column, variable and expression has related data type in SQL. You would use these data types while creating your tables. You would choose a particular data type for a table column based on your requirement.

SQL Server offers six categories of data types for your use:

Exact Numeric Data Types:

Table 4

Data Type	From	To
bigint	9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 -1
numeric	-10^38 +1	10^38 -1
money	922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Approximate Numeric Data Types:

Table 5

Data Type	From	To
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

Date and Time Data Types:

Table 6

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Stores a date like June 30, 1991	
time	Stores a time of day like 12:30 P.M.	

Note: Here, date-time has milliseconds accuracy whereas smalldatetime has 1 minute accuracy.

Character Strings Data Types:

Table 7

Data Type	From	To
char	char	Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
varchar	varchar	Maximum of 8,000 characters.(Variable-length non-Unicode data).
varchar(max)	varchar(max)	Maximum length of 231characters, Variable-length non-Unicode data (SQL Server 2005 only).
text	text	Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

Unicode Character Strings Data Types:

Table 8

DATA TYPE	Description
nchar	Maximum length of 4,000 characters.(Fixed length Unicode)
nvarchar	Maximum length of 4,000 characters.(Variable length Unicode)
nvarchar(max)	Maximum length of 231characters (SQL Server 2005 only).(Variable length Unicode)
ntext	Maximum length of 1,073,741,823 characters. (Variable length Unicode)

Binary Data Types:

Table 9

DATA TYPE	Description

binary	Maximum length of 8,000 bytes(Fixed-length binary data)
varbinary	Maximum length of 8,000 bytes.(Variable length binary data)
varbinary(max)	Maximum length of 231 bytes (SQL Server 2005 only). (Variable length Binary data)
image	Maximum length of 2,147,483,647 bytes. (Variable length Binary Data)

Misc Data Types:

Table 10

DATA TYPE	**Description**
sql_variant	Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
timestamp	Stores a database-wide unique number that gets updated every time a row gets updated
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML data. You can store xml instances in a column or a variable (SQL Server 2005 only).
cursor	Reference to a cursor object
table	Stores a result set for later processing

Self-Check -2	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. What are the elements NOT contained in the WHERE clause predicate of the SELECT query?
   A. Comparison operator
   B. Comparison condition
   C. Column Name
   D. Table Name
2. Which of the following clauses are mandatory in an SQL query?
   A. SELECT, FROM
   B. SELECT,FROM,WHERE
   C. SELECT,WHERE
   D. SELECT,WHERE,ORDER BY
3. The attribute or the attribute combination is the primary key of the table
   A. UNIQUE
   B. PRIMARY KEY
   C. FOREIGN KEY
   D. CHECK
4. Which SQL statement is used to extract data from a database?
   A. SELECT
   B. EXTRACT
   C. GET
   D. OPEN
5. Which SQL statement is used to update data in a database?
   A. UPDATE Your answer
   B. SAVE
   C. MODIFY
   D. SAVE AS

Score = ____________

Rating: _____________

Name: ___________________________ Date: ________________

Short Answer Questions

Note: **Satisfactory rating - 5 points Unsatisfactory - below 5 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet3: Retrieving data from specific columns in a single table

1.3 Retrieving data from specific columns in a single table

SELECT statements

The Select statement is the most commonly used SQL command that allows you to retrieve records from one or more tables in your database.
The basic SELECT statement in sql has 3 clauses: SELECT, FROM and WHERE

- The SELECT clause specifies the table columns that are retrieved.
- The FROM clause specifies the table or tables from which columns and rows are returned.
- The WHERE clause specifies the condition restricting the query. You can restrict the number of rows by using comparison operators, character strings, and logical operators as search conditions.
- The WHERE clause is optional; if missing, all table rows are accessed.
- Syntax of SQL SELECT Statement:
- SELECT <column_list> FROM <table_name_list>< [WHERE Clause]>< [search_condition]>

Examples
Following are examples of SQL SELECT statements:

To select all columns from a table (Customers) for rows where the Last_Name column has Smith for its value, you would send this SELECT statement to the server back end:

```
SELECT * FROM Customers WHERE Last_Name='Smith';
```

The server back end would reply with a result set similar to this:

Table 1

Cust_No	Last_Name	First_Name
1001	Smit	John
2039	Smith	David
2098	Smith	Matthew

3 rows in set (0.05 sec)

To return only the Cust_No and First_Name columns, based on the same criteria as above, use this statement:

```
SELECT Cust_No, First_Name FROM Customers WHERE Last_Name='Smith';
```

The subsequent result set might look like:

Table 2

Cust_No	First_Name
1001	John
2039	David
2098	Matthew

3 rows in set (0.05 sec)

To make a WHERE clause find inexact matches, add the pattern-matching operator LIKE. The LIKE operator uses the % (percent symbol) wild card to match zero or more characters, and the underscore (_) wild card to match exactly one character. For example:

To select the First_Name and Nickname columns from the Friends table for rows in which the Nickname column contains the string "brain", use this statement:

```
SELECT First_Name, Nickname FROM Friends WHERE Nickname LIKE '%brain%';
```

The subsequent result set might look like:

Table 3

Last_Name	NickName
Ben	Brainiac
Glen	peabrain
steven	Nobrainer

3 rows in set (0.03 sec)

To query the same table, retrieving all columns for rows in which the First_Name column's value begins with any letter and ends with "en", use this statement:

```
SELECT * FROM Friends WHERE First_Name LIKE '_en';
```

The result set might look like:

Table 4

First_Name	Last_Name	Nickname

Ben	Smith	Brainiac
Jen	Peters	Sweetpea

2 rows in set (0.03 sec)

If you used the % wild card instead (for example, '%en') in the example above, the result set might look like:

Table 5

First_Name	Last_Name	Nickname
Ben	Smith	Brainiac
Glen	Jones	Peabrain
Jen	Peters	Sweetpea
Ben	Smith	Brainiac

4 rows in set (0.05 sec)

Self-Check -3	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. With SQL, how do you select all the columns from a table named "Persons"?
   A. SELECT * FROM Persons Your B. SELECT Persons
   SELECT [all] FROM Persons C. SELECT *.Persons
2. With SQL, how do you select all the records from a table named "Persons" where the value of the column "FirstName" is "Peter"?
   A. SELECT * FROM Persons WHERE FirstName='Peter' Your answer
   B. SELECT [all] FROM Persons WHERE FirstName='Peter'
   C. SELECT * FROM Persons WHERE FirstName<>'Peter'
   D. SELECT [all] FROM Persons WHERE FirstName LIKE 'Peter'
3. Which of the following is used to end a SQL query?
   A. : B. ; C. . D. ..
4. The result of a SQL SELECT statement is a(n) ________ .
   A. report C. file
   B. form D. table

Score = ___________
Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

Note: Satisfactory rating - 4 points Unsatisfactory - below 4 points
You can ask you teacher for the copy of the correct answers.

Information Sheet 4: using Order by to sort query output

1.4 Using Order by to sort query output

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

Syntax
The basic syntax of the ORDER BY clause is as follows −
SELECT column-list FROMtable_name[WHERE condition]

[ORDER BY column1, column2, ..columnN] [ASC | DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort that column should be in the column-list.Example

Consider the CUSTOMERS table having the following records –

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code block has an example, which would sort the result in an ascending order by the NAME and the SALARY −

```
SQL> SELECT * FROM CUSTOMERS
  ORDER BY NAME, SALARY;
```

This would produce the following result –

Table 2

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00

5	Hardik	27	Bhopal	8500.00
3	kaushik	23	Kota	2000.00
2	Khilan	25	Delhi	1500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00
1	Ramesh	32	Ahmedabad	2000.00

The following code block has an example, which would sort the result in the descending order by NAME.

```
SQL> SELECT * FROM CUSTOMERS
  ORDER BY NAME DESC;
```

This would produce the following result –

Table 3

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
7	Muffy	24	Indore	10000.00
6	Komal	22	MP	4500.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
4	Chaitali	25	Mumbai	6500.00

Self-Check -4	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. Which one of the following sorts rows in SQL? GROUP BY
   - **A.** SORT BY
   - **B.** ALIGN BY
   - C. ORDER BY
   - D. Group by
2. To sort the results of a query use:
   - A. SORT BY
   - B. GROUP BY
   - C. ORDER BY
   - D. None of the above is correct

.

Score = ____________
Rating: _____________

Name: ___________________________ **Date:** ________________

Short Answer Questions

Note: Satisfactory rating - 4 points Unsatisfactory - below 4 points
You can ask you teacher for the copy of the correct answers.

Information Sheet 5: Retrieving restricted number of rows using 'where' clause

1.5 Retrieving restricted number of rows using 'where' clause

- **Where Clause**

The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables. If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc.

Syntax: The basic syntax of SELECT statement with WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

You can specify a condition using comparison or logical operators like >, <, =, LIKE, NOT, etc. Below examples would make this concept clear.

Example: Consider the CUSTOMERS table having the following records:

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	|32	Ahmedabad	2000.00
2	Khilan	|25	Delhi	1500.00
3	kaushik	|23	Kota	2000.00
4	Chaitali	|25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	|22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example which would fetch ID, Name and Salary fields from the CUSTOMERS table where salary is greater than 2000:

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

This would produce the following result:

Table 2

ID	NAME	SALARY
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

Following is an example, which would fetch ID, Name and Salary fields from the CUSTOMERS table for a customer with name Hardik. Here, it is important to note that all the strings should be given inside single quotes ('') whereas numeric values should be given without any quote as in above example:

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';
```

This would produce the following result:

Table 3

ID	NAME	SALARY
5	Hardik	8500.00

SQL TOP

The SQL TOP clause is used to fetch a TOP N number or X percent records from a table.

Note: All the databases do not support TOP clause. For example MySQL supports LIMIT clause to fetch limited number of records and Oracle uses ROWNUM to fetch limited number of records.

Syntax:

The basic syntax of TOP clause with SELECT statement would be as follows:

```
SELECT TOP number|percentcolumn_name(s)
FROM table_name
WHERE [condition]
```

Example:

Consider the CUSTOMERS table having the following records:

Table 4

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	|32	Ahmedabad	2000.00
2	Khilan	|25	Delhi	1500.00
3	kaushik	|23	Kota	2000.00
4	Chaitali	|25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	|22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example on SQL server, which would fetch top 3 records from CUSTOMERS table:

SQL> SELECT TOP 3 * FROM CUSTOMERS;

This would produce the following result:

Table 5

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00

If you are using MySQL server, then here is an equivalent example:

SQL> SELECT * FROM CUSTOMERS

LIMIT 3;

This would produce the following result:

Table 6

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00

3	kaushik	23	Kota	2000.00

If you are using Oracle server, then here is an equivalent example:

SQL> SELECT * FROM CUSTOMERS

WHERE ROWNUM <= 3;

This would produce the following result:

Table 7

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00

Updating Tables

The SQL UPDATE Query is used to modify the existing records in a table.

You can use WHERE clause with UPDATE query to update selected rows otherwise all the rows would be affected.

Syntax: The basic syntax of UPDATE query with WHERE clause is as follows:

UPDATE table_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

You can combine N number of conditions using AND or OR operators.

Example: Consider the CUSTOMERS table having the following records:

Table 8

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example, which would update ADDRESS for a customer whose ID is 6:

SQL> UPDATE CUSTOMERS

SET ADDRESS = 'Pune'

WHERE ID = 6;

Now, CUSTOMERS table would have the following records:

Table 9

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

If you want to modify all ADDRESS and SALARY column values in CUSTOMERS table, you do not need to use WHERE clause and UPDATE query would be as follows:

SQL> UPDATE CUSTOMERS

SET ADDRESS = 'Pune', SALARY = 1000.00;

Now, CUSTOMERS table would have the following records:

Table 10

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	1000.00
2	Khilan	25	Pune	1000.00
3	kaushik	23	Pune	1000.00
4	Chaitali	25	Pune	1000.00
5	Hardik	27	Pune	1000.00
6	Komal	22	Pune	1000.00
7	Muffy	24	Pune	1000.00

Deleting New Rows

The SQL DELETE Query is used to delete the existing records from a table.

You can use WHERE clause with DELETE query to delete selected rows, otherwise all the records would be deleted.

Syntax: The basic syntax of DELETE query with WHERE clause is as follows:

DELETE FROM table_name

WHERE [condition];

You can combine N number of conditions using AND or OR operators.

Example: Consider the CUSTOMERS table having the following records:

Table 11

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	|Muffy	24	Indore	10000.00

Following is an example, which would DELETE a customer, whose ID is 6:

SQL> DELETE FROM CUSTOMERS

WHERE ID = 6;

Now, CUSTOMERS table would have the following records:

Table 12

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00

2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	|Muffy	24	Indore	10000.00

If you want to DELETE all the records from CUSTOMERS table, you do not need to use WHERE clause and DELETE query would be as follows:

SQL> DELETE FROM CUSTOMERS;

Now, CUSTOMERS table would not have any record.

Like Operator

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator:

- The percent sign (%)
- The underscore (_)

SQL Wildcard Characters

In SQL, wildcard characters are used with the SQL LIKE operator.

SQL wildcards are used to search for data within a table.

With SQL, the wildcards are:

Table 13

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match
[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

Syntax: The basic syntax of % and _ is as follows:

SELECT FROM table_name WHERE column LIKE 'XXXX%' or

SELECT FROM table_name WHERE column LIKE '%XXXX%' or

SELECT FROM table_name WHERE column LIKE 'XXXX_' or

SELECT FROM table_name WHERE column LIKE '_XXXX' or

SELECT FROM table_name WHERE column LIKE '_XXXX_'

You can combine N number of conditions using AND or OR operators. Here, XXXX could be any numeric or string value.

Example: Here are number of examples showing WHERE part having different LIKE clause with '%' and '_' operators:

Table 14

Statement	Description
WHERE SALARY LIKE '200%'	Finds any values that start with 200
WHERE SALARY LIKE '%200%'	Finds any values that have 200 in any position
WHERE SALARY LIKE '_00%'	Finds any values that have 00 in the second and third positions
WHERE SALARY LIKE '2_%_%'	Finds any values that start with 2 and are at least 3 characters in length
WHERE SALARY LIKE '%2'	Finds any values that end with 2
WHERE SALARY LIKE '_2%3'	Finds any values that have a 2 in the second position and end with a 3
WHERE SALARY LIKE '2___3'	Finds any values in a five-digit number that start with 2 and end with 3

Let us take a real example, consider the CUSTOMERS table having the following records:

Table 15

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	|32	Ahmedabad	2000.00
2	Khilan	|25	Delhi	1500.00
3	kaushik	|23	Kota	2000.00
4	Chaitali	|25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	|22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example, which would display all the records from CUSTOMERS table where SALARY starts with 200:

SQL> SELECT * FROM CUSTOMERS

WHERE SALARY LIKE '200%';

This would produce the following result:

Table 16

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00

SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

SQL IN Syntax:

SELECT *column_name(s)* FROM *table_name* WHERE *column_name* IN (*value1*,*value2*,...);

Below is a selection from the "Customers" table:

Table 17

Cust_ID	CustomerName	ContactName	Address	City	PostalCode	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglundssnabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

IN Operator Example: The following SQL statement selects all customers with a City of "Paris"or "London":

SELECT * FROM Customers WHERE City IN ('Paris','London');

SQL BETWEEN Operator

The BETWEEN operator is used to select values within a range. The BETWEEN operators select values within a range. The values can be numbers, text, or dates.

SQL BETWEEN Syntax

SELECT *column_name(s)* FROM *table_name* WHERE *column_name*BETWEEN *value1* AND *value2;*

Example: The following SQL statement selects all products with a price BETWEEN 10 and 20:

SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

NOT BETWEEN Operator Example: to display the products outside the range of the previous example, use NOT BETWEEN:

SELECT * FROM Products WHERE Price NOT BETWEEN 10 AND 20;

BETWEEN Operator with IN Example: The following SQL statement selects all products with a price BETWEEN 10 and 20, but products with a CategoryID of 1,2, or 3 should not be displayed:

SELECT * FROM Products WHERE (Price BETWEEN 10 AND 20) AND NOT CategoryID IN (1,2,3);

BETWEEN Operator with Text Value Example: - The following SQL statement selects allproducts with a ProductName beginning with any of the letter BETWEEN 'C' and 'M':

SELECT * FROM Products WHERE ProductName BETWEEN 'C' AND 'M';

NOT BETWEEN Operator with Text Value Example:-The following SQL statement selects all products with a ProductName beginning with any of the letter NOT BETWEEN 'C' and 'M':

SELECT * FROM Products WHERE ProductName NOT BETWEEN 'C' AND 'M';

Sample Table: Below is a selection from the "Orders" table:

Table 18

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/9/1996	1
10252	76	4	7/10/1996	2

BETWEEN Operator with Date Value Example: - The following SQL statement selects all orders with an OrderDate BETWEEN '04-July-1996' and '09-July-1996':

SELECT * FROM Orders WHERE OrderDate BETWEEN #07/04/1996# AND #07/09/1996#;

SQL Constraints

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL.

NOT NULL Constraint: Ensures that a column cannot have NULL value.

DEFAULT Constraint: Provides a default value for a column when none is specified.

UNIQUE Constraint: Ensures that all values in a column are different.

PRIMARY Key: Uniquely identified each rows/records in a database table.

FOREIGN Key: Uniquely identified a rows/records in any another database table.

CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.

INDEX: Use to create and retrieve data from the database very quickly.

Data Integrity

The following categories of data integrity exist with each RDBMS −

Entities Integrity-There are no duplicate rows in a table.

Domain Integrity -Enforces valid entries for a given column by restricting the type, the format, or the range of values.

Referential integrity -Rows cannot be deleted, which are used by other records.

User-Defined Integrity -Enforces some specific business rules that do not fall into entity, domain or referential integrity.

UNIQUE Constraint:

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

CREATE TABLE Persons(IDint NOT NULL UNIQUE,LastNamevarchar(255) NOT NULL ,FirstNamevarchar(255),Ageint);

DROP a UNIQUE Constraint:

SyntaxDROP DATABASE *databasename*;

PRIMARY Key

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a composite key.

If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

Note: You would use these concepts while creating database tables.

Example: Here is the syntax to define ID attribute as a primary key in a CUSTOMERS table.

CREATE TABLE CUSTOMERS (ID INT NOT NULL, NAME VARCHAR (20) NOT NULL,AGE INT NOT NULL,ADDRESS CHAR (25) ,SALARY DECIMAL (18, 2) PRIMARY KEY (ID));

Delete Primary Key:

You can clear the primary key constraints from the table, Use Syntax:

ALTER TABLE CUSTOMERS DROP PRIMARY KEY ;

FOREIGN Key

A foreign key is a key used to link two tables together. This is sometimes called a referencing key. Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

DROP a FOREIGN KEY Constraint:

To drop a FOREIGN KEY constraint, use the following SQL:

```
ALTER TABLE ORDERS
  DROP FOREIGN KEY;
```

CHECK Constraint

The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered into the table.

Example: For example, the following SQL creates a new table called CUSTOMERS and adds five columns. Here, we add a CHECK with AGE column, so that you can not have any CUSTOMER below 18 years:

CREATE TABLE CUSTOMERS(ID INT NOT NULL, NAME VARCHAR (20) NOT NULL,AGE INT NOT NULL CHECK (AGE >= 18),ADDRESS CHAR (25) ,SALARY DECIMAL (18, 2), PRIMARY KEY (ID));

INDEX

The INDEX is used to create and retrieve data from the database very quickly. Index can be created by using single or group of columns in a table. When index is created, it is assigned a ROWID for each row before it sorts out the data.

Proper indexes are good for performance in large databases, but you need to be careful while creating index. Selection of fields depends on what you are using in your SQL queries.

Example:For example, the following SQL creates a new table called CUSTOMERS and adds five columns:

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,NAME VARCHAR (20)NOT NULL,AGE INT NOT NULL,ADDRESS CHAR (25),SALARY   DECIMAL (18,2),PRIMARY KEY (ID));

Now, you can create index on single or multiple columns using the following syntax:

```
CREATE INDEX index_name
   ON table_name( column1, column2.....);
```

To create an INDEX on AGE column, to optimize the search on customers for a particular age, following is the SQL syntax:

```
CREATE INDEX idx_ageON CUSTOMERS ( AGE);
```

DROP an INDEX Constraint:

To drop an INDEX constraint, use the following SQL:

```
ALTER TABLE CUSTOMERS
DROP INDEX idx_age;
```

Self-Check -5	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. Choose the database elements whose values can be compared in a WHERE clause of a SELECT query.
   A. Column C. Procedure
   B. Sequence D. Literal
2. Which of the following clause is used to limit the number of rows retrieved from a SELECT query?
   A. LIMIT C. AND
   B. WHERE D. FROM
3. Which of the following values can NOT be returned after evaluation of WHERE clause condition?
   A. UNKNOWN C. FALSE
   B. TRUE D. NULL
4. Which of the following is false regarding the WHERE clause?
   A. The WHERE can compare values in columns, literal, arithmetic expressions, or functions.
   B. The WHERE clause contains column name
   C. Column aliases can be used in the WHERE clause.
   D. The WHERE clause cannot contain list of values or constants.
5. What is the minimum number of WHERE clauses that must be present in a SELECT query?
   A. 1 C. 0
   B. 2 D. 3

Score = ___________
Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

***Note:* Satisfactory rating - 5 points Unsatisfactory - below 5 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet6: Retrieving restricted number of rows using select statement

1.6 Retrieving restricted number of rows using select statement

Data retrieval from data base is done through appropriate and efficient use of SQL. Three concepts from relational theory encompass the capability of the SELECT statement: projection, selection, and joining.

- **Projection**: A project operation selects only certain columns (fields) from a table. The result table has a subset of the available columns and can include anything from a single column to all available columns.
- **Selection**: A select operation selects a subset of rows (records) in a table (relation) that satisfy a selection condition. The ability to select rows from out of complete result set is called Selection. It involves conditional filtering and data staging. The subset can range from no rows, if none of the rows satisfy the selection condition, to all rows in a table.
- **Joining:** A join operation combines data from two or more tables based on one or more common column values. A join operation enables an information system user to process the relationships that exist between tables. The join operation is very powerful because it allows system users to investigate relationships among data elements that might not be anticipated at the time that a database is designed.

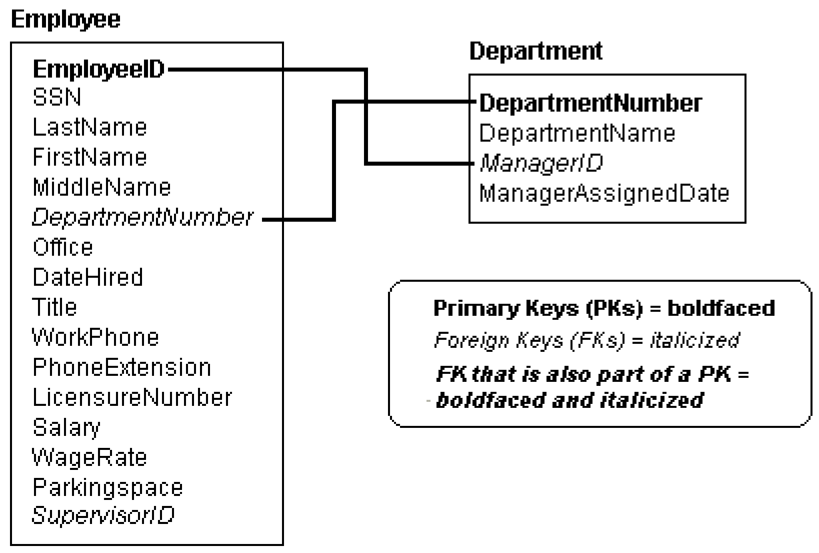


Fig 1.6.1

Consider the above table structures. Fetching first_name name, department_id and salary for a single employee from EMPLOYEES table is Projection. Fetching employee details whose salary is less than 5000, from EMPLOYEES table is Selection. Fetching employee's first name, department name by joining EMPLOYEES and DEPARTMENTS is joining.

Basic SELECT statement

The basic syntax for a SELECT statement is presented below.

```
SELECT [DISTINCT | ALL] {* | select_list}
FROM {table_name [alias] | view_name}
   [{table_name [alias]  |view_name}]...
[WHERE condition]
[GROUP BY condition_list]
[HAVING condition]
[ORDER BY {column_name | column_#  [ ASC | DESC ] } ...
```

The SELECT clause is mandatory and carries out the relational project operation.

The FROM clause is also mandatory. It identifies one or more tables and/or views from which to retrieve the column data displayed in a result table.

The WHERE clause is optional and carries out the relational select operation. It specifies which rows are to be selected.

The GROUP BY clause is optional. It organizes data into groups by one or more column names listed in the SELECT clause.

The optional HAVING clause sets conditions regarding which groups to include in a result table. The groups are specified by the GROUP BY clause.

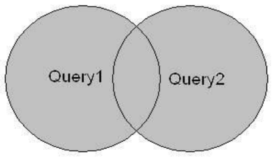
The ORDER BY clause is optional. It sorts query results by one or more columns in ascending or descending order.

Self-Check -6	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I multiple choice

1. Which of the following statements are correct about the WHERE clause?
   B. Column Alias can be used in WHERE clause to refer a column
   C. Comparison operator is an optional element in WHERE clause condition
   D. Functions can be used as operands in the WHERE clause
   E. There can be multiple WHERE clause in a SELECT query
2. What is true about SET operators?
   A. They change values of rows
   B. They combine the results of only two component queries into one result
   C. They combine the results of two or more component queries into one result
   D. They combine the results of 10 component queries into two result sets.
3. What does the selection of columns in a SELECT statement known as?
   A. Retrieval C. Projection
   B. Selection D. Limiting
4. What does the restriction of rows returned by a SELECT statement known as
   A. Retrieval C. Restricting
   B. Projection D. Selection
5. Which SET operator does the following figure indicate?



A. UNION C. INTERSECT
B. UNION ALL D. MINUS

Score = ____________
Rating: _____________

Name: ______________________ Date: ________________
Short Answer Questions

Note: Satisfactory rating - 5 points Unsatisfactory - below 5 points
You can ask you teacher for the copy of the correct answers.

Information sheet 7: Using comparison operators in the where clause to compare numeric, character, string, date and time data

1.7 Using comparison operators in the where clause to compare numeric, character, string, date and time data

- **Expressions**

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value.

SQL EXPRESSIONs are like formulas and they are written in query language. You can also use them to query the database for specific set of data.

Syntax:

Consider the basic syntax of the SELECT statement as follows:

SELECT column1, column2, columnN FROMtable_name

WHERE [CONDITION|EXPRESSION];

There are different types of SQL expressions, which are mentioned below

- **Comparison Operators**

A comparison (or relational) operator is a mathematical symbol which is used to compare between two values.

Comparison operators are used in conditions that compare one expression with another. The result of a comparison can be TRUE, FALSE, or UNKNOWN (an operator that has one or two NULL expressions returns UNKNOWN).

The following table describes different types of comparison operators –

Table 1

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.

>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

Syntax

SELECT [column_name | * | expression] <comparison operator>

[column_name | * | expression]

SELECT [column_name | * | expression] <comparison operator>

[column_name | * | expression]

FROM <table_name>

WHERE < expression >[comparison operator] < expression >;

- **Numeric functions**

✓ Numeric functions perform manipulation of numbers that normally are contained in a table column.

✓ There is a differentiated range of built-in numeric functions in the various RDBMS, yet are there some functions which have the same name and executes the same.

✓ There are also functions that perform the same but does not have the same name in different RDBMS.

Here is a collection of the most important built-in numeric functions:

Table 2

Function	RDBMS	Description
ABS()	SQL server, MySQL, Oracle, PostgreSQL	A mathematical function that returns the absolute (positive) value of the specified numeric expression.
CEIL()	MySQL, PostgreSQL, Oracle	Returns the smallest integer greater than, or equal to, the specified numeric expression.
CEILING()	SQL server, MySQL, PostgreSQL	
EXP()	SQL server, MySQL, Oracle, PostgreSQL	Returns the value of e (the base of natural logarithms) raised to the power of X.
FLOOR()	SQL server, MySQL, Oracle, PostgreSQL	Returns the largest integer less than or equal to the specified numeric expression.
LN()	MySQL, PostgreSQL, Oracle	Returns the natural logarithm of the specified float expression.
LOG()	MySQL, SQL server	
POWER()	SQL server, MySQL, Oracle, PostgreSQL	Returns the value of the specified expression to the specified power.
ROUND()	SQL server, MySQL, Oracle, PostgreSQL	Returns a numeric value, rounded to the specified length or precision.
SIGN()	SQL server, MySQL, Oracle, PostgreSQL	Returns the positive (+1), zero (0), or negative (-1) sign of a numeric expression.
SQRT()	SQL server, MySQL, Oracle	Return the square root of the argument
SIN()	SQL server, MySQL, Oracle	Return the sine of the argument
COS()		Return the cosine

- **Character**

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

- ✓ The percent sign (%)
- ✓ The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

Syntax

The basic syntax of % and _ is as follows −

SELECT FROM table_name WHERE column LIKE 'XXXX%' or

SELECT FROM table_name WHERE column LIKE '%XXXX%' or

SELECT FROM table_name WHERE column LIKE 'XXXX_' or

SELECT FROM table_name WHERE column LIKE '_XXXX' or

SELECT FROM table_name WHERE column LIKE '_XXXX_'

- **String functions**

- ✓ String functions perform manipulation of text that normally are contained in a table column.

- ✓ There is a differentiated range of built-in string functions in the various RDBMS, yet are there some functions which have the same name and executes the same.
- ✓ There are also functions that perform the same but does not have the same name in different RDBMS.

Here is a collection of the most important built-in string functions:

Table 3

Function	**RDBMS**	**Description**
CONCAT()	SQL server, MySQL, Oracle, PostgreSQL	Returns a string that is the result of concatenating two or more string values. Oracle takes only two string values.
LEN()	SQL server	Returns the length of a string, measured in bytes.
LENGTH()	MySQL, Oracle, PostgreSQL	
LOWER()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character expression after converting uppercase character data to lowercase.
LTRIM()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character expression after it removes leading blanks. Oracle and PostgreSQL has an additional parameter that specifies what you want to remove (the default is leading blanks).
REPLACE()	SQL server, MySQL, Oracle, PostgreSQL	Replaces all occurrences of a specified string value with another string value.
RTRIM()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character string after truncating all trailing blanks. Oracle and PostgreSQL has an additional parameter that specifies what you want to remove (the default is trailing blanks).
SUBSTR()	MySQL, Oracle, PostgreSQL	Returns a part of a string expression.
SUBSTRING()	SQL server, MySQL, PostgreSQL	
TRIM()	MySQL, Oracle	Returns a character string after truncating all leading blanks and trailing blanks. Oracle and PostgreSQL has additionally specifying what you want to remove leading, trailing or both (default is both and blanks). With SQL server you have to use a combination of LTRIM() and RTRIM().
UPPER()	SQL server, MySQL, Oracle	Returns a character expression with lowercase character data converted to uppercase.

- **Date and Time**

Date and Time functions perform manipulation of dates and times that normally are contained in a table column.

There is a differentiated range of built-in date and time functions in the various RDBMS. You will in most DBMS find common functionality in handling date and time, but the solution to achieve the results will vary.

Here is a collection of the most important built-in date and time functions:

Table 4

Function	**RDBMS**	**Description**
NOW()	MySQL, PostgreSQL	Returns the current date and time. PostgreSQL will return the current date and time with timezone.
GETDATE()	SQL server	
CURRENT_DATE	Oracle	
DATE_ADD() & DATE_SUB()	MySQL	These functions are used to calculate a new date from a date by either adding a type of date units or subtract a type of date units.
DATEADD()	SQL server	
{date +/- part of day}	Oracle	
EXTRACT()	MySQL, Oracle, PostgreSQL	These functions are used to extract partial information in a date.
DATEPART()	SQL server	

Self-Check -7	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: II multiple choice

1. Which of the following are valid operators for the WHERE clause?

   A. >= B. IS NULL C. != D. IS LIKE

2. Which of the following is a comparison operator in SQL?

   A. Double equal sign (==)
   B. LIKE
   C. BETWEEN
   D. Single equal sign (=)

3. Which function is used to divides one numeric expression by another and get the remainder?

   A. POWER
   B. MOD
   C. ROUND
   D. REMAINDER

4. Which of the following is used to replace a specific character in a given string in Oracle DB?

   A. LTRIM B. TRIM C. TRUNC D. REPL

5. Which of the following is a legal expression in SQL?

A. SELECT NULL FROM SALES;
B. SELECT NAME FROM SALES;
C. .SELECT * FROM SALES WHEN PRICE = NULL;
D. D.SELECT # FROM SALES;

6. Which of the below alphanumeric characters are used to signify concatenation operator in SQL?

   A. + B. || C. - D. ::

7. What is the order of evaluation of set operators?

   A. Left to Right
   B. Right to Left
   C. Random Evaluation
   D. Top to Bottom

Score = ____________

Rating: ______________

Name: __________________________ Date: ________________

Short Answer Questions

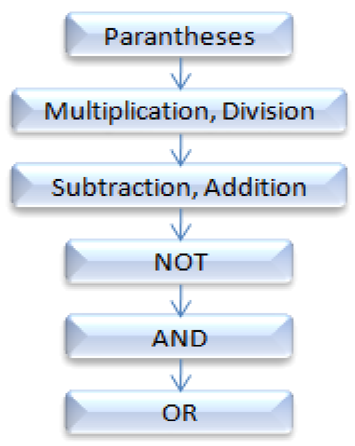
Note: **Satisfactory rating - 7 points Unsatisfactory - below 7 points**

You can ask you teacher for the copy of the correct answers.

Information Sheet: 8 Using Boolean operators with the correct precedence

1.8 Using Boolean operators with the correct precedence

SQL Operator Precedence



- Operator precedence describes the order in which operations are performed when an expression is evaluated.
- Operations with a higher precedence are performed before those with a lower precedence.
- **A parenthesis has** the highest precedence and **OR** has the lowest.

Boolean operators in SQL

There are three Logical Operators namely, AND, OR, and NOT. These operators compare two conditions at a time to determine whether a row can be selected for the output. When you are retrieving data using a SELECT statement, you can use logical operators in the WHERE clause, which allows you to combine more than one condition.

Table 1

Operator	**Description**
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard

	operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

Syntax

SELECT [column_name | * | expression] [logical operator]
[column_name | * | expression .....]
FROM <table_name>
WHERE <expressions> [logical operator |
arithmetic operator | ...] <expressions>;

For example: if you want to find the names of students who are studying either Maths or Science, the query would be like,
SELECT first_name, last_name, subject FROMstudent_details WHERE subject = 'Maths' OR subject = 'Science'

For Example: To find the names of the students between the age 10 to 15 years, the query would be like:
SELECT first_name, last_name, age FROMstudent_details WHERE age >= 10 AND age <= 15;

Self-Check -8	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: I write the correct query syntax/script statement for the below questions

Sample table

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	JozyAltidor	Moscow	200	5007
3001	Brad Guzan	London		5005

1. Write a SQL statement to display all the customers, who are either belongs to the city New York or not had a grade above 100.
2. Write a query statement to display all customers in New York who have a grade value above 100
3. Write a SQL statement to display all customers, who are either belongs to the city New York or had a grade above 100.
4. Write a query to display all customers with a grade above 100.

Score = ____________

Rating: _____________

Name: ___________________________ Date: ________________

Short Answer Questions

Note: **Satisfactory rating - 4 points Unsatisfactory - below 4 points**
You can ask you teacher for the copy of the correct answers.

Information sheet 9: using criteria in the 'where' clause to check for a

range of values

1.9 Using Criteria in the 'where' clause to check for a range of values

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.

Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

You can specify a condition using the comparison or logical operators like >, <, =, **LIKE, NOT**, etc. The following examples would make this concept clear.

Example

Consider the CUSTOMERS table having the following records –

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	|32	Ahmedabad	2000.00
2	Khilan	|25	Delhi	1500.00
3	kaushik	|23	Kota	2000.00
4	Chaitali	|25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	|22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code is an example which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 −

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

This would produce the following result –

Table 2

ID	NAME	SALARY
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

The following query is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name **Hardik**.

Here, it is important to note that all the strings should be given inside single quotes (''). Whereas, numeric values should be given without any quote as in the above example.

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';
```

This would produce the following result –

Table 3

ID	NAME	SALARY
5	Hardik	8500.00

1.9.1 Using 'where' clause to select values from a list

HAVING specifies a search condition for a group or an aggregate and typically used in a GROUP BY clause.

When GROUP BY is not used, HAVING behaves like a WHERE clause.

HAVING syntax included in the SELECT statement:

```
SELECT [ALL | DISTINCT] select_list
  FROM table/view_list
  [WHERE search_conditions]
  [GROUP BY group_by_list]
  [HAVING having_conditions]
  [ORDER BY order_by_list ]
```

Steps when a query includes WHERE, GROUP BY, aggregates, and HAVING.

The WHERE clause acts first to find the rows you want.

The GROUP BY clause divides these rows into groups.

After the groups are formed, SQL calculates the aggregate values (SUM, MIN, etc.) for each group.

HAVING checks the results from the rows produced by the grouping to see which ones qualify for a final view..

At last the ORDER BY comes in action to sort on any expressions.

In this example HAVING eliminates those sets that include only one book:

```
select publisher, min(price) as "Low Price",
max(price) as "High price",
count(*) "Numbers"
from bookstore
group by publisher
having count(*) > 1
order by  publisher, count(*);
```

Result

Table 4

publisher	Low Price	High price	Numbers
Wrox	29	42	5

1.9.2 using where clause to check values that match a pattern

LIKE is used in a WHERE statement to find specified pattern in a column. Rows that satisfy the pattern are selected.

LIKE Example:

```
selectcustomer_id, telephone_type, Telephone_number
```

from telephones

wheretelephone_number like '_3%-54' ;

The character, **%**(percent), has the meaning, any character type and any number of characters

while _(underline) has the meaning any character type at the position it appears! So this example will list rows with some columns from telephones table where the Telephone_number has the second character as 3 and ends with the three characters -54.

result

Table 5

customer_id	telephone_type	telephone_number
10003	Cell	(371) 153-54
10003	Work	(371) 143-54

NOT LIKE Example:

selectcustomer_id, telephone_type, Telephone_number

from telephones

wheretelephone_number not like '-54' and telephone_type='Home' ;

This example will list rows with some columns from telephones table where the Telephone_number not ends with the three characters -54 and telephone is 'Home' type.

The result should be:

Table 6

customer_id	telephone_type	telephone_number
10001	Home	(978) 667-94
10002	Home	(679) 234-94

Self-Check -9	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part II Multiple choice

1. Which one is correct statement about the Comparison Operators?
   A. A comparison (or relational) operator is a mathematical symbol which is used to compare between two values
   B. Comparison operators are used in conditions that compare one expression with another
   C. The result of a comparison can be TRUE, FALSE, or UNKNOWN (an operator that has one or two NULL expressions returns UNKNOWN).
   D. All
2. Which one is incorrect statement about the Comparison Operators?
   A. There are three Logical Operators namely, AND, OR, and NOT.
   B. These operators compare two conditions at a time to determine whether a row can be selected for the output
   C. When you are retrieving data using a SELECT statement, you can use logical operators in the WHERE clause, which allows you to combine more than one condition.
   D. All
3. Which one is correct statement about the Mathematical Functions and Operators?
   A. Mathematical operators are provided for many Postgre SQL types
   B. For types without common mathematical conventions for all possible permutations
   C. Arithmetic operators perform mathematical operations on two expressions of one or more of the data types of the numeric data type category.
   D. All
4. The FROM and WHERE clause specifies the target table(s) for__ statements
   A. SELECT
   B. UPDATE
   C. DELETE
   D. ALL

5. Which of following will be used to join rows with other tables if the column values fall in a range defined by inequality operators?
   A. Equijoin
   B. Simple join
   C. Non-equijoin
   D. None of the above

Score = ___________

Rating: ____________

Name: _________________________Date: _______________

Short Answer Questions

***Note:* Satisfactory rating - 5 points Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Information Sheet: 10 Use SQL syntax to suppress duplicate values from query results

1.10 Using SQL syntax to suppress duplicate values from query results

The SQL **DISTINCT** keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only those unique records instead of fetching duplicate records.

Syntax

The basic syntax of DISTINCT keyword to eliminate the duplicate records is as follows −

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

Example

Consider the CUSTOMERS table having the following records –

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

First, let us see how the following SELECT query returns the duplicate salary records.

```
SQL> SELECT SALARY FROM CUSTOMERS
   ORDER BY SALARY;
```

This would produce the following result, where the salary (2000) is coming twice which is a duplicate record from the original table.

Table 2

SALARY
1500.00
2000.00
2000.00
4500.00
6500.00
8500.00
10000.00

Now, let us use the DISTINCT keyword with the above SELECT query and then see the

result.

```
SQL> SELECT DISTINCT SALARY FROM CUSTOMERS ORDER BY SALARY;
```

This would produce the following result where we do not have any duplicate entry.

Table 3

SALARY
1500.00
2000.00
4500.00
6500.00
8500.00
10000.00

Self-Check -10	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: multiple choices

1. Which of the following clause is used to suppress duplicates in a SELECT statement?

A. INTERSECT C. DISTINCT

B. DUPLICATE D. UNIQUE

2. Chose the statements which correctly specify a rule to write a SQL statement

A. SQL statements are case sensitive

B. Keywords can be abbreviated to build a standard

C. SQL statements are case in-sensitive

D. clauses must be placed together

3. Identify the capabilities of SELECT statement.

A. Projection C. Data Control

B. Selection D. Transaction

Score = ___________

Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet: 11Taking action to exclude null values from a query result

1.11 Taking action to exclude null values from a query result

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

The SQL **DISTINCT** keyword, which we have already discussed is used in conjunction with the SELECT statement to eliminate all the duplicate records and by fetching only the unique records.

Syntax

The basic syntax of a DISTINCT keyword to eliminate duplicate records is as follows.

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

Example

Consider the CUSTOMERS table having the following records.

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	|32	Ahmedabad	2000.00
2	Khilan	|25	Delhi	1500.00
3	kaushik	|23	Kota	2000.00
4	Chaitali	|25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	|22	MP	4500.00
7	Muffy	24	Indore	10000.00

First, let us see how the following SELECT query returns duplicate salary records.

```
SQL> SELECT SALARY FROM CUSTOMERS
  ORDER BY SALARY;
```

This would produce the following result where the salary of 2000 is coming twice which is a duplicate record from the original table.

Table 2

SALARY
1500.00
2000.00
2000.00
4500.00
6500.00
8500.00
10000.00

Now, let us use the DISTINCT keyword with the above SELECT query and see the result.

```
SQL> SELECT DISTINCT SALARY FROM CUSTOMERS
  ORDER BY SALARY;
```

This would produce the following result where we do not have any duplicate entry.

Table 3

SALARY
1500.00
2000.00
4500.00
6500.00
8500.00
10000.00

Self-Check -11	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I: Multiple choices

1. Which clause should you use to exclude group results in a query using group functions?

   A. WHERE B. HAVING C. GROUP BY D. ORDER BY

2. What of the following can be used to fetch non-matching rows along with the matching rows between a source and a target table in Oracle DB?

   A. EQUI-JOIN C. NATURAL JOIN

   B. SELF-JOIN D. OUTER-JOIN

3. Chose the statements which correctly define a NULL value.

   A. NULL is a special value with zero bytes
   B. NULL is no value or unknown value
   C. NULL is represented by a blank space
   D. NULL is not same as zero

4. Determine the output of the below query

```
SELECT sal + NULL FROM empHERE empno = 7369;
```

   A. sal + NULL C. 0
   B. NULL D. 1250

5. Which of the following values can NOT be returned after evaluation of WHERE clause condition?

   A. UNKNOWN C. FALSE
   B. TRUE D. NULL

Score = ____________
Rating: _____________

Name: ___________________________ Date: ________________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Operation sheet 1

Name______________________ ID number__________________

Started-time_________________ Finished-time________________

Instruction

Install DBMS (SQL Server)

- To install SQL Server 2008 Express, you must have administrative rights on the computer.
- The different editions/version of SQL Server share several common software requirements such as Microsoft Windows installer and Microsoft .Net Framework.

Example: Before installing **SQL Server 2008 Express SP1**, first you have to install **Microsoft Windows installer 4.5** and
Microsoft .NET Framework version 3.5 SP1.

The SQL Server Installation Wizard provides a single feature tree to install all SQL Server components such as:

- Database Engine
- Analysis Services
- Reporting Services
- Integration Services
- Master Data Services
- Data Quality Services
- Management tools
- Connectivity components

You can install each component individually or select a combination of the components listed above.

A Step by Step guide to installing SQL Server 2008 simply and successfully:

- Thiswill teach you the basics required for a typical (problem-free) installation of SQL Server 2008.
- Before you start the installation, you will need to install the .Net 3.5 Framework and windows installer 4.5.

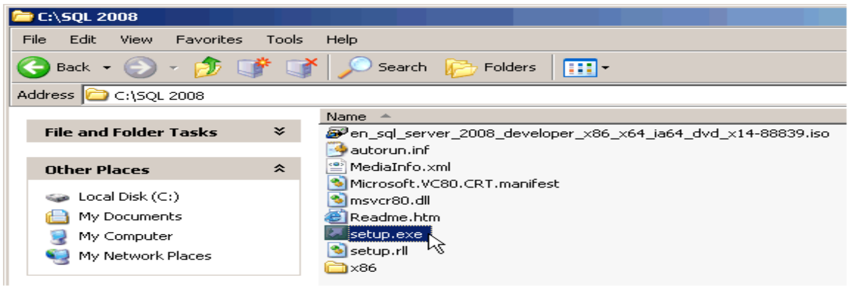
STEP 1: Copy the installation files

First it is recommended that you copy the entire directory structure from the SQL Server 2008 installation disc to the C: drive of the machine you are going to install it on.

This has three advantages:

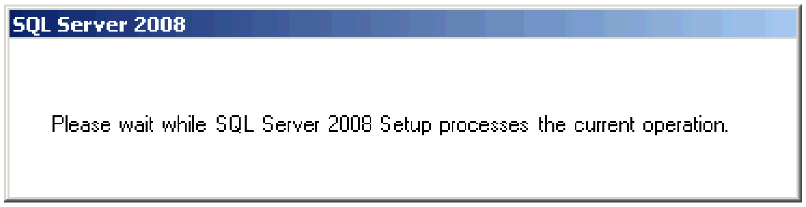
- It makes the installation process much faster than running it from CD/DVD once it gets started.
- It allows you to easily add or remove components later, without having to hunt around for the CD/DVD.
- If your media is damaged and a file won't copy, you get to find out now, rather than halfway through the installation.

Here's what my system looks like after the copy:



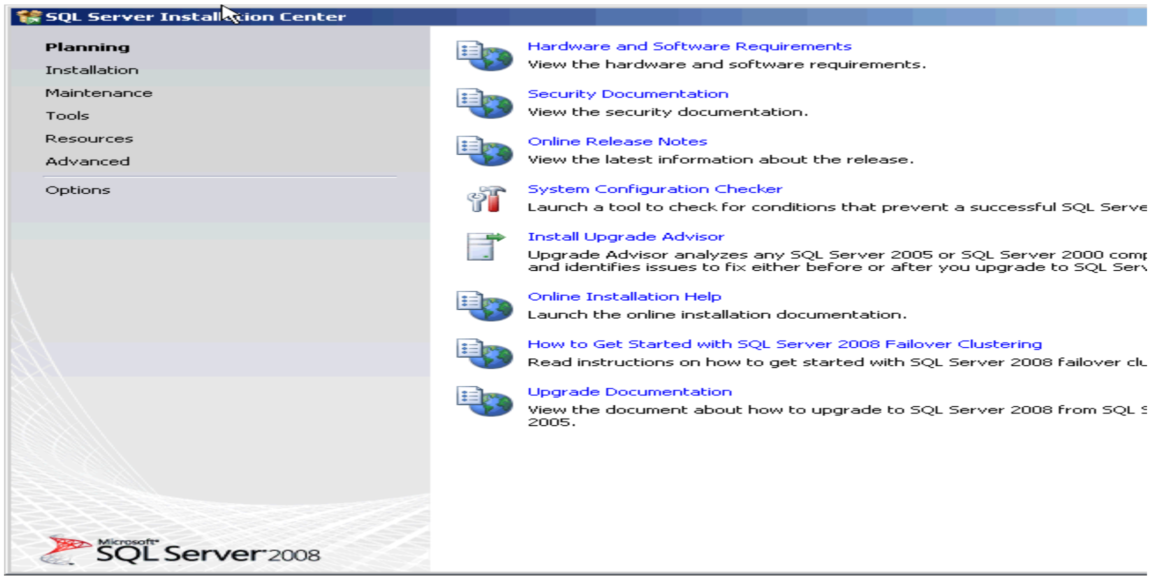
en_sql_server_2008_developer_x86_x64_ia64_dvd_x14-88839.iso
autorun.inf
MediaInfo.xml
Microsoft.VC80.CRT.manifest
msvcr80.dll
Readme.htm
setup.exe
setup.rll
x86

STEP 2: Double click on the setup.exe file. After a few seconds a dialog box appears:



Please wait while SQL Server 2008 Setup processes the current operation.

This will disappear from the screen and then the main installation page appears:


SQL Server Installation Center

Planning
Installation
Maintenance
Tools
Resources
Advanced
Options

Hardware and Software Requirements
View the hardware and software requirements.

Security Documentation
View the security documentation.

Online Release Notes
View the latest information about the release.

System Configuration Checker
Launch a tool to check for conditions that prevent a successful SQL Serve

Install Upgrade Advisor
Upgrade Advisor analyzes any SQL Server 2005 or SQL Server 2000 comp and identifies issues to fix either before or after you upgrade to SQL Serv

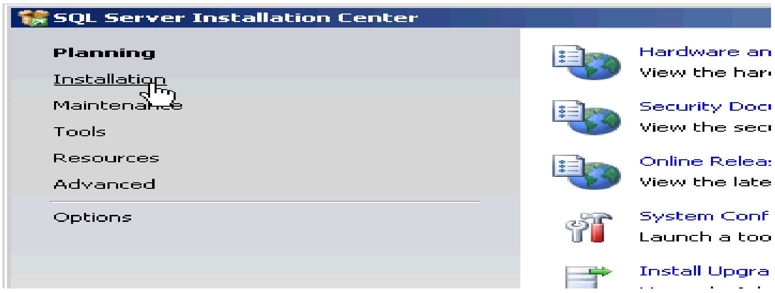
Online Installation Help
Launch the online installation documentation.

How to Get Started with SQL Server 2008 Failover Clustering
Read instructions on how to get started with SQL Server 2008 failover clu

Upgrade Documentation
View the document about how to upgrade to SQL Server 2008 from SQL S 2005.

Microsoft SQL Server 2008

STEP 3: Click on the **Installation** hyperlink on the left hand side of the screen:


SQL Server Installation Center

Planning
Installation
Maintenance
Tools
Resources
Advanced
Options

Hardware an
View the har

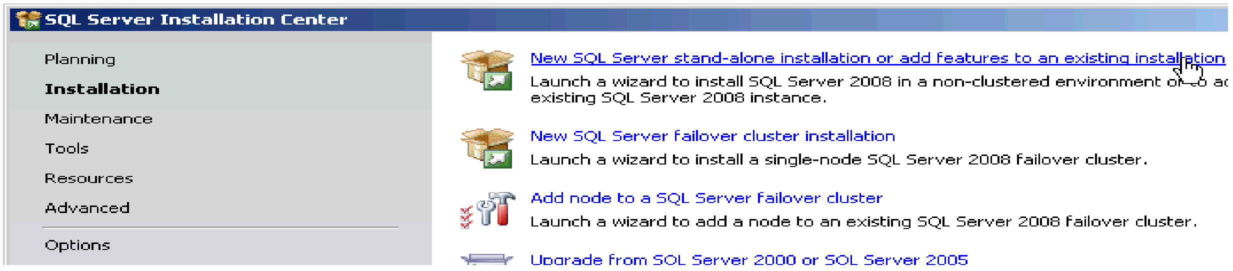
Security Doc
View the sec

Online Relea
View the late

System Conf
Launch a too

Install Upgra

STEP 4 : Click on the **"New Server stand-alone installation"** link on the right side of the screen:


SQL Server Installation Center

Planning
Installation
Maintenance
Tools
Resources
Advanced
Options

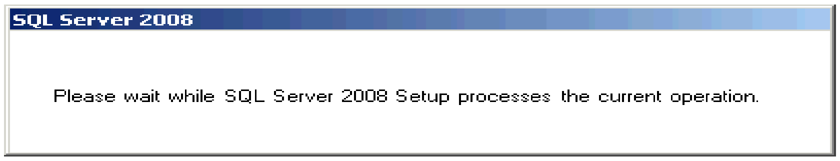
New SQL Server stand-alone installation or add features to an existing installation
Launch a wizard to install SQL Server 2008 in a non-clustered environment or to ad existing SQL Server 2008 instance.

New SQL Server failover cluster installation
Launch a wizard to install a single-node SQL Server 2008 failover cluster.

Add node to a SQL Server failover cluster
Launch a wizard to add a node to an existing SQL Server 2008 failover cluster.

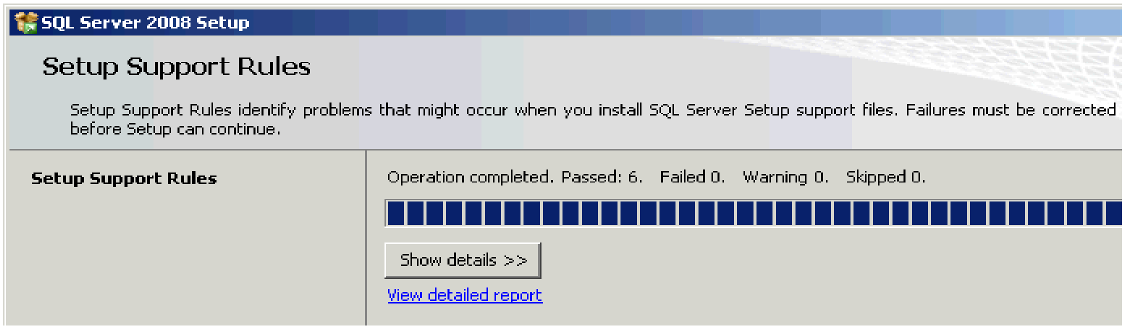
Upgrade from SQL Server 2000 or SQL Server 2005

The following dialog appears on the screen whilst the install program prepares for installation:


SQL Server 2008

Please wait while SQL Server 2008 Setup processes the current operation.

After a minute or so (the timing will vary according to your system), the following screen appears:


SQL Server 2008 Setup

Setup Support Rules

Setup Support Rules identify problems that might occur when you install SQL Server Setup support files. Failures must be corrected before Setup can continue.

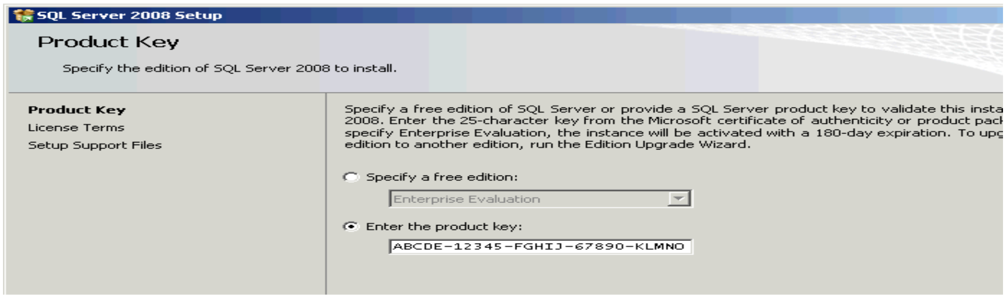
Setup Support Rules

Operation completed. Passed: 6. Failed 0. Warning 0. Skipped 0.

STEP 5 (optional): If any checks have failed, click on the Show details button or "View detailed report link" to find out the cause, correct it, and then click on the Re-run button to perform the checks again.

STEP 6: Product key

If all checks have passed, click on the OK button. After a few moments, the option to select the edition and to enter the license key (or “product key”) will appear. Note that the product key box may already be populated, depending on which edition you have. Don’t enter the product key we’ve shown here, it won’t work on your system!:


SQL Server 2008 Setup

Product Key

Specify the edition of SQL Server 2008 to install.

Specify a free edition of SQL Server or provide a SQL Server product key to validate this insta 2008. Enter the 25-character key from the Microsoft certificate of authenticity or product pack specify Enterprise Evaluation, the instance will be activated with a 180-day expiration. To upg edition to another edition, run the Edition Upgrade Wizard.

Specify a free edition: Enterprise Evaluation

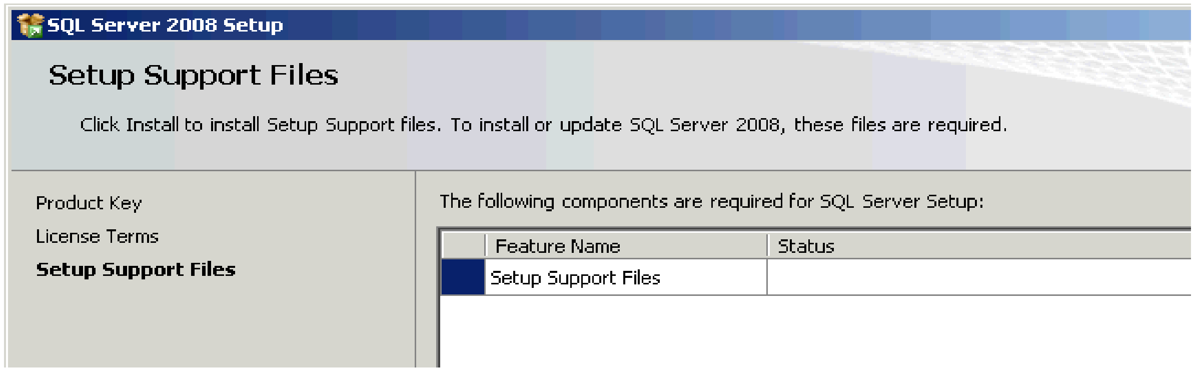
[✓ Enter the product key:] ABCDE-12345-FGHIJ-67890-KLMNO

STEP 7: License Terms

Enter the product key into the box, or choose the free edition if you're evaluating SQL Server 2008, and click on the Next button:

Click in the **"I accept the license terms"** check box, and then click on the **Next** button again.

STEP 8: click on the **Install** button:



SQL Server 2008 Setup

Setup Support Files

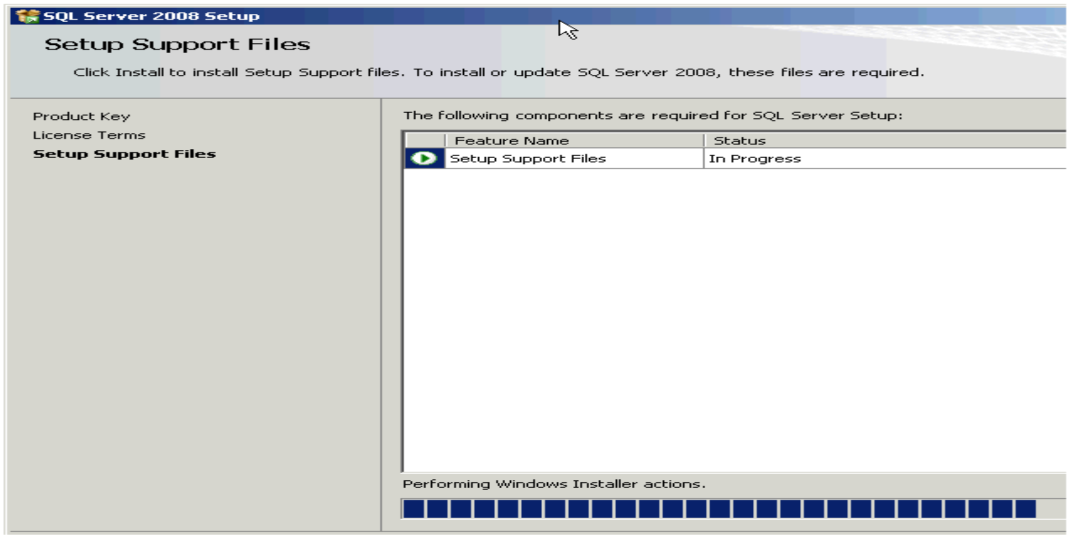
Click Install to install Setup Support files. To install or update SQL Server 2008, these files are required.

Product Key
License Terms
Setup Support Files

The following components are required for SQL Server Setup:

Feature Name	Status
Setup Support Files	

The following screen will appear whilst Windows Installer prepares itself for the installation. This will take a short while:



SQL Server 2008 Setup

Setup Support Files

Click Install to install Setup Support files. To install or update SQL Server 2008, these files are required.

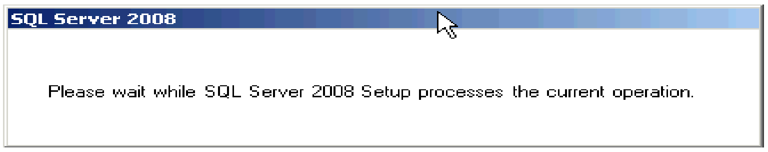
Product Key
License Terms
Setup Support Files

The following components are required for SQL Server Setup:

Feature Name	Status
Setup Support Files	In Progress

Performing Windows Installer actions.

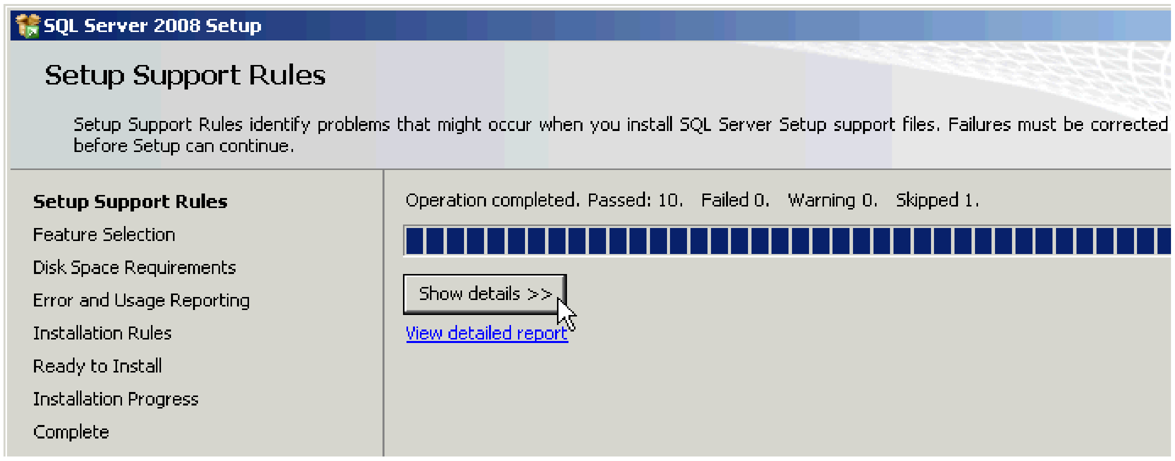
After 30 seconds or so the dialog appears again:



SQL Server 2008

Please wait while SQL Server 2008 Setup processes the current operation.

STEP 9: Setup Support Rules. If all is well, the following screen appears:



SQL Server 2008 Setup

Setup Support Rules

Setup Support Rules identify problems that might occur when you install SQL Server Setup support files. Failures must be corrected before Setup can continue.

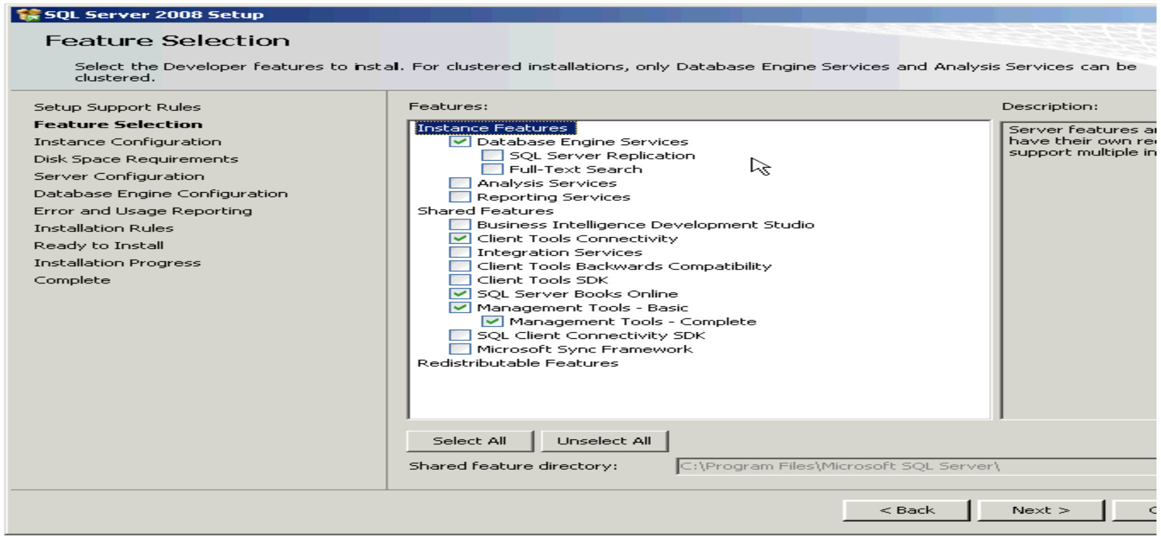
- **Setup Support Rules**
- Feature Selection
- Disk Space Requirements
- Error and Usage Reporting
- Installation Rules
- Ready to Install
- Installation Progress
- Complete

Operation completed. Passed: 10. Failed 0. Warning 0. Skipped 1.

Click on the **Next** button again.

STEP 10: Feature Selection. Select the features you want to install.

At a minimum, the following are useful (I'd argue essential), but what you need will depend on your needs:



SQL Server 2008 Setup

Feature Selection

Select the Developer features to instal. For clustered installations, only Database Engine Services and Analysis Services can be clustered.

- Setup Support Rules
- **Feature Selection**
- Instance Configuration
- Disk Space Requirements
- Server Configuration
- Database Engine Configuration
- Error and Usage Reporting
- Installation Rules
- Ready to Install
- Installation Progress
- Complete

Features:

- Instance Features
  - ✓ Database Engine Services
    - ☐ SQL Server Replication
    - ☐ Full-Text Search
  - ☐ Analysis Services
  - ☐ Reporting Services
- Shared Features
  - ☐ Business Intelligence Development Studio
  - ✓ Client Tools Connectivity
  - ☐ Integration Services
  - ☐ Client Tools Backwards Compatibility
  - ☐ Client Tools SDK
  - ✓ SQL Server Books Online
  - ✓ Management Tools - Basic
    - ✓ Management Tools - Complete
  - ☐ SQL Client Connectivity SDK
  - ☐ Microsoft Sync Framework
- Redistributable Features

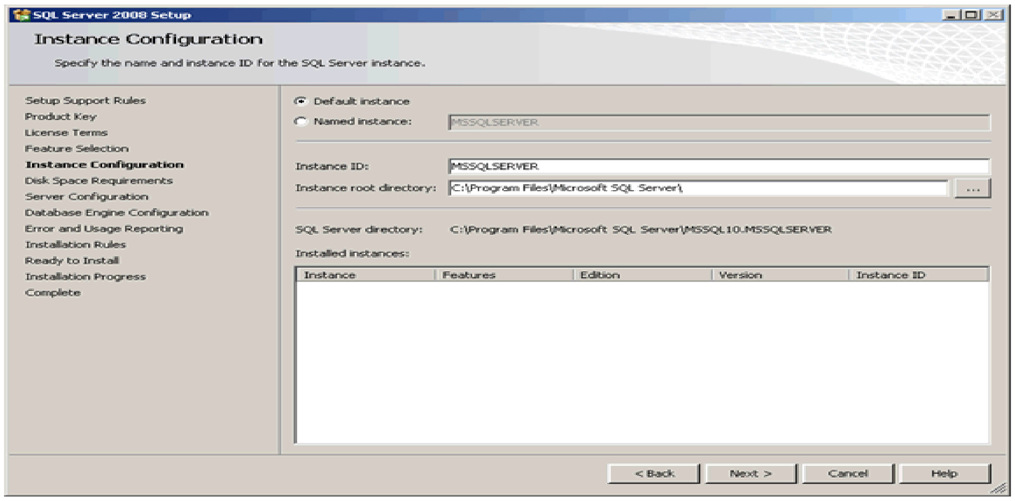
Description:

Server features ar
have their own re
support multiple in

Shared feature directory: C:\Program Files\Microsoft SQL Server\

Click on the **Next** button.

STEP 11: Instance Configuration. After a short while the following screen appears:



SQL Server 2008 Setup

Instance Configuration

Specify the name and instance ID for the SQL Server instance.

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

(●) Default instance
( ) Named instance: MSSQLSERVER

Instance ID: MSSQLSERVER
Instance root directory: C:\Program Files\Microsoft SQL Server\

SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER

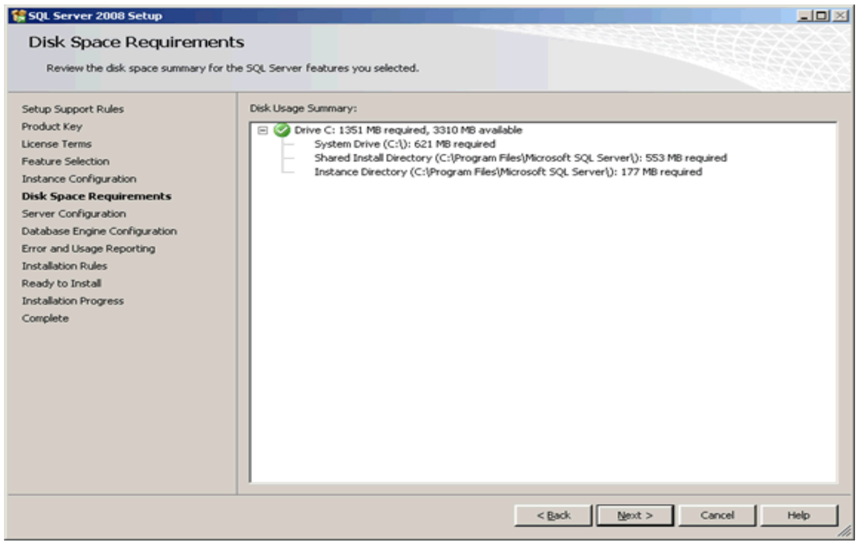
Installed instances:

Instance	Features	Edition	Version	Instance ID

For most installations, keep the default settings. Click on the **Next** button.

STEP 12: Disk Space Requirements

This screen just tells you if you have sufficient disk space on the drive you're installing to, and what's going to be installed where.



SQL Server 2008 Setup

Disk Space Requirements

Review the disk space summary for the SQL Server features you selected.

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Disk Usage Summary:

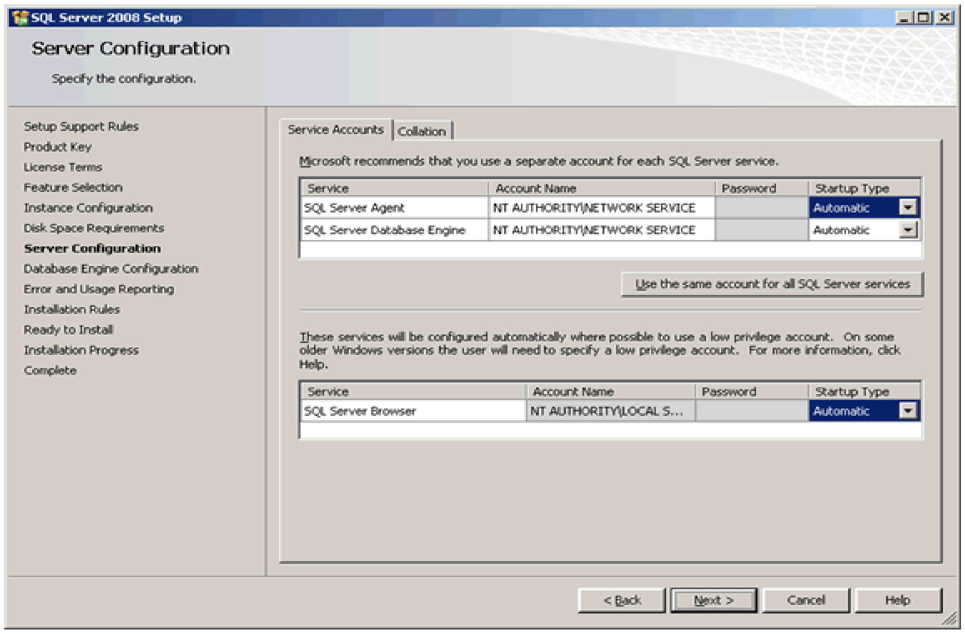
```
Drive C: 1351 MB required, 3310 MB available
    System Drive (C:\): 621 MB required
    Shared Install Directory (C:\Program Files\Microsoft SQL Server\): 553 MB required
    Instance Directory (C:\Program Files\Microsoft SQL Server\): 177 MB required
```

Click on **Next**.

STEP 13: Server Configuration. This step allows you to set up the service accounts that will be used to run SQL Server. If you have created Windows NT or Active Directory accounts for use with services, use these.

If not, then just to get the installation up and working, use the built-in Network Service account for all three services listed (this account does not require a password).

This allows SQL Server to start up after installation. However, it can be easily changed later to another account through the Services applet (Control Panel -> Administrator Tools -> Services):



SQL Server 2008 Setup

Server Configuration

Specify the configuration.

- Setup Support Rules
- Product Key
- License Terms
- Feature Selection
- Instance Configuration
- Disk Space Requirements
- **Server Configuration**
- Database Engine Configuration
- Error and Usage Reporting
- Installation Rules
- Ready to Install
- Installation Progress
- Complete

Service Accounts | Collation

Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Agent	NT AUTHORITY\NETWORK SERVICE		Automatic
SQL Server Database Engine	NT AUTHORITY\NETWORK SERVICE		Automatic

Use the same account for all SQL Server services

These services will be configured automatically where possible to use a low privilege account. On some older Windows versions the user will need to specify a low privilege account. For more information, click Help.

Service	Account Name	Password	Startup Type
SQL Server Browser	NT AUTHORITY\LOCAL S...		Automatic

< Back | Next > | Cancel | Help

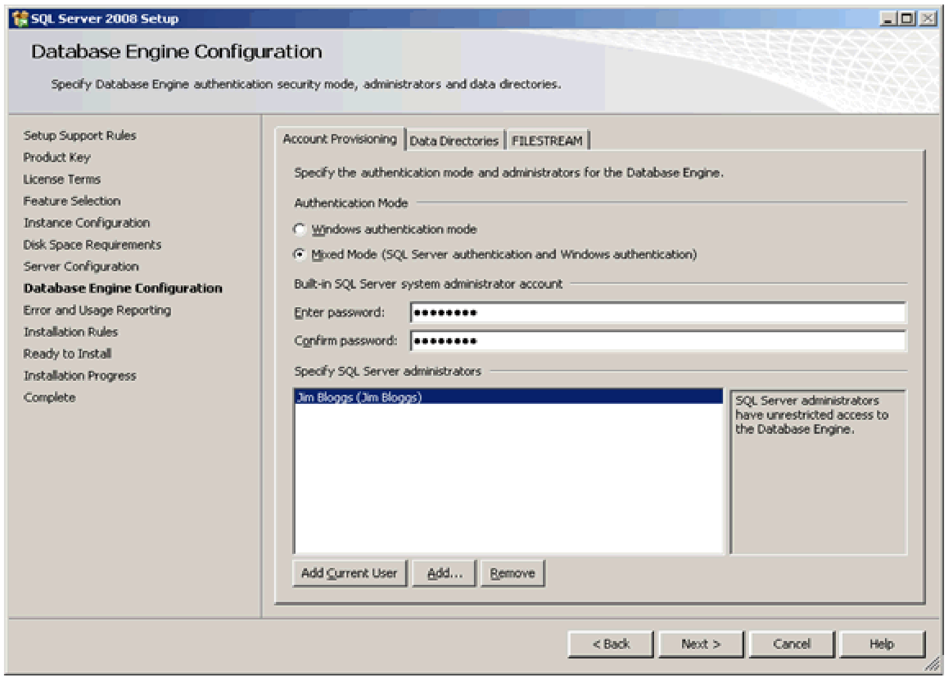
In addition, remember to change the **Startup Type** to **Automatic**, for all three services. This automatically starts the SQL Server database engine, SQL Agent and SQL Browser services when the server is re-booted.

The first service runs the SQL Server database engines executable process. The other two services allow scheduled jobs to run after installation (and after a re-boot), and allow the SQL Server to be found by clients on the network.

Finally, click on **Next**.

STEP 14: Database Engine Configuration – Account Provision.

This screen allows you to set up database engine security.


SQL Server 2008 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

( ) Windows authentication mode

(•) Mixed Mode (SQL Server authentication and Windows authentication)

Built-in SQL Server system administrator account

Enter password: ••••••••

Confirm password: ••••••••

Specify SQL Server administrators

Jim Bloggs (Jim Bloggs)

SQL Server administrators have unrestricted access to the Database Engine.

Change the **Authentication Mode** to **Mixed Mode** unless you are **certain** you only need Windows-only authentication.

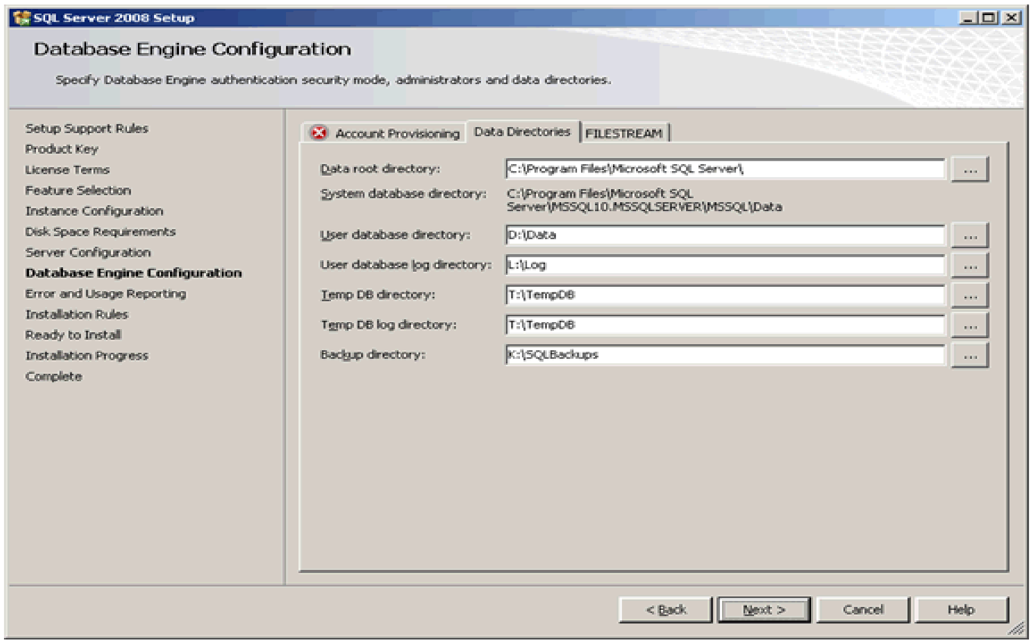
- **Many third party applications rely on SQL Server logins to operate correctly, so if you are setting up a server for a third party application, rather than one developed in-house, enabling Mixed Mode authentication is a good idea.**

If you pick Mixed Mode security, you must also enter a password for the sysadmin account (sa).

Enter and confirm a secure password for the sa account and keep it somewhere safe.

Note that you MUST also provide a Windows NT account on the local machine as a SQL Server administrator. If you do not want Windows system administrators to be able walk up to the box and login to SQL Server, create a new, local, dummy Windows user and add this account instead. Otherwise, add in the local administrator account, or your own Windows account on the domain in which the SQL Server will reside.

STEP 15: Database Engine Configuration – Data Directories. Click on the Data Directories tab.


SQL Server 2008 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

Data root directory:	C:\Program Files\Microsoft SQL Server\
System database directory:	C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Data
User database directory:	D:\Data
User database log directory:	L:\Log
Temp DB directory:	T:\TempDB
Temp DB log directory:	T:\TempDB
Backup directory:	K:\SQLBackups

Change the directories to specify which drives in your system will be used for the various types of database files.

Generally it's advisable to put the User database directory and User log directory on separate physical drives for performance, but it will depend on how Windows has been configured and how many disk drives you have available.

If you are installing on a single drive laptop or desktop, then simply specify:

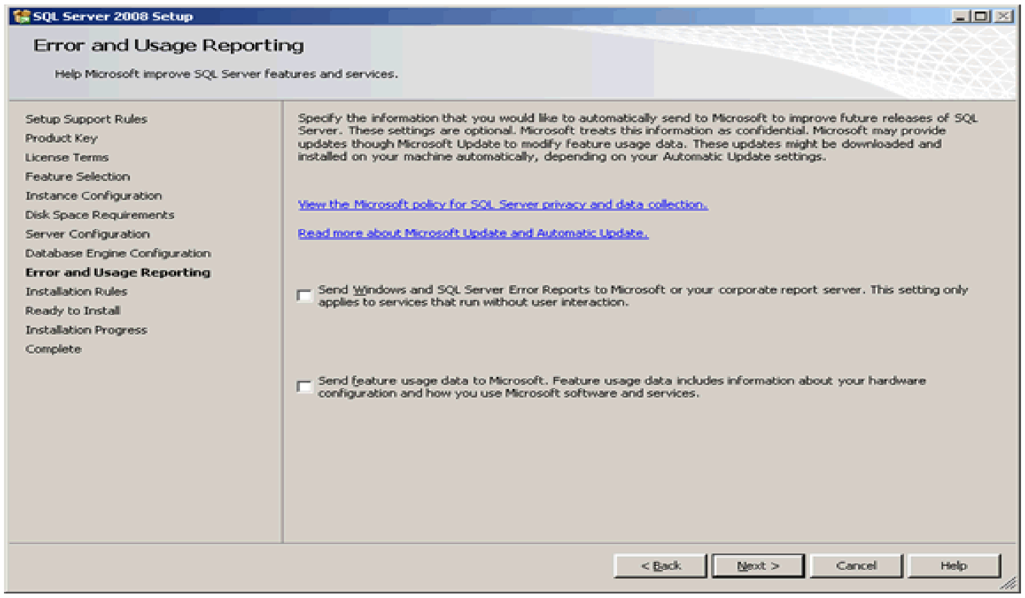
Data root directory	C:\Program Files\Microsoft SQL Server
User database directory	C:\Data
User log directory	C:\Logs
Temp DB directory	C:\TempDB
Temp Log directory	C:\TempDB
Backup directory	C:\Backups

Do not click on the **FILESTREAM** tab unless you know you need to change these options, as it is not generally required for most installations, but can easily be changed by using sp_configure 'filestream_access_level', ''after SQL Server has been installed.

Click on **Next**.

STEP 16: Error Usage Reporting

This screen simply asks if you want to send error information to Microsoft and can safely be skipped if you do not want to share any information.



SQL Server 2008 Setup

Error and Usage Reporting

Help Microsoft improve SQL Server features and services.

Specify the information that you would like to automatically send to Microsoft to improve future releases of SQL Server. These settings are optional. Microsoft treats this information as confidential. Microsoft may provide updates though Microsoft Update to modify feature usage data. These updates might be downloaded and installed on your machine automatically, depending on your Automatic Update settings.

View the Microsoft policy for SQL Server privacy and data collection.

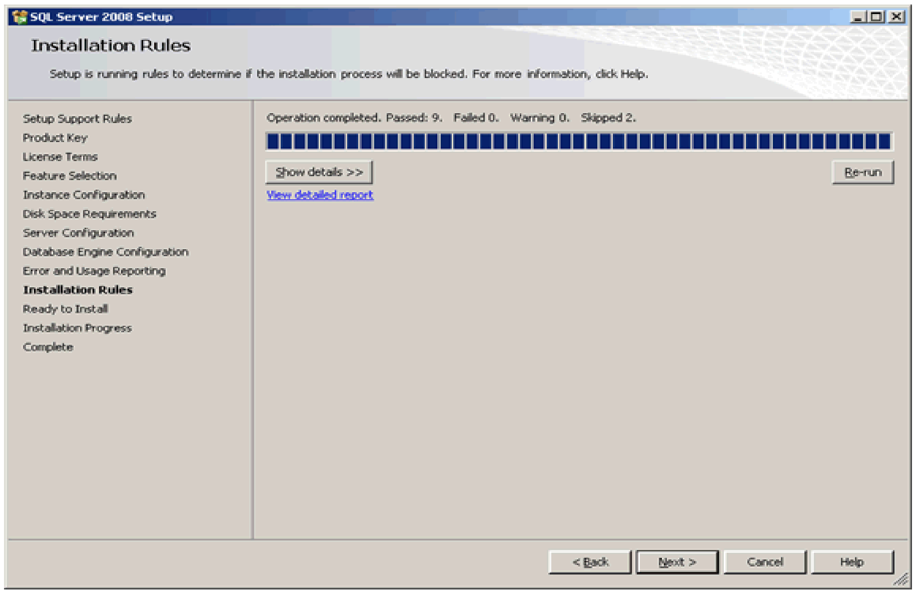
Read more about Microsoft Update and Automatic Update.

- [ ] Send Windows and SQL Server Error Reports to Microsoft or your corporate report server. This setting only applies to services that run without user interaction.
- [ ] Send feature usage data to Microsoft. Feature usage data includes information about your hardware configuration and how you use Microsoft software and services.

Click boxes if you want to help Microsoft help you. Click on **Next** again…

STEP 16: Installation Rules

This screen simply checks if there are any processes or other installations running which will stop the installation of SQL Server 2008.



Installation Rules

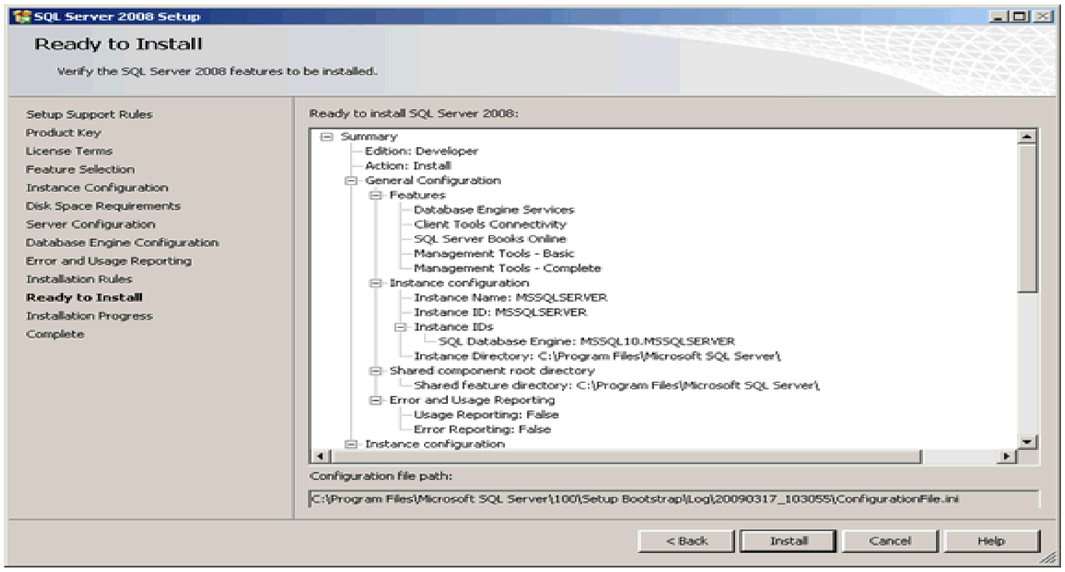
Setup is running rules to determine if the installation process will be blocked. For more information, click Help.

Operation completed. Passed: 9. Failed 0. Warning 0. Skipped 2.

Click on **Next** again – you're almost ready to install:

STEP 17: Ready to Install

This screen summarizes what you are about to install and gives you a last chance to cancel or change anything that's wrongly configured:



Ready to Install

Verify the SQL Server 2008 features to be installed.

Ready to install SQL Server 2008:

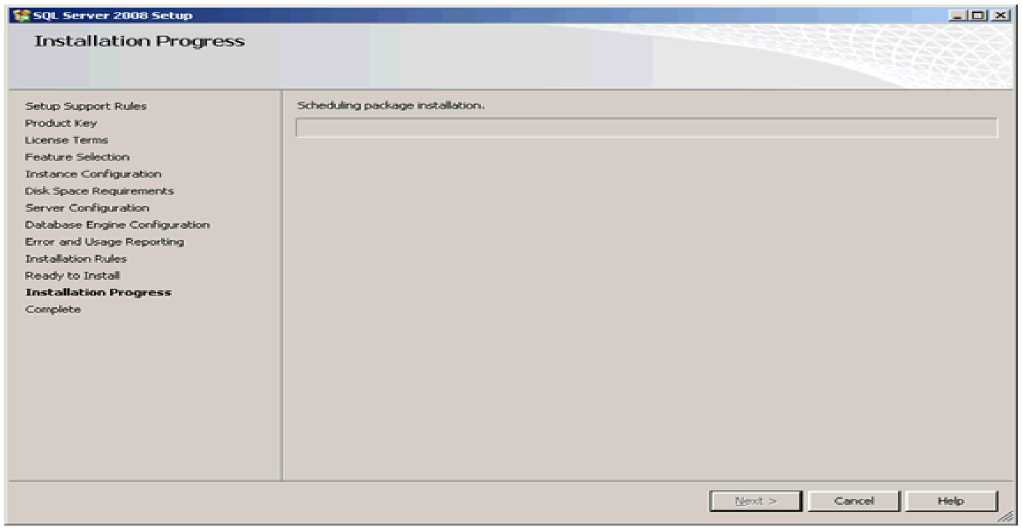
```
Summary
  Edition: Developer
  Action: Install
  General Configuration
    Features
      Database Engine Services
      Client Tools Connectivity
      SQL Server Books Online
      Management Tools - Basic
      Management Tools - Complete
    Instance configuration
      Instance Name: MSSQLSERVER
      Instance ID: MSSQLSERVER
      Instance IDs
        SQL Database Engine: MSSQL10.MSSQLSERVER
      Instance Directory: C:\Program Files\Microsoft SQL Server\
    Shared component root directory
      Shared feature directory: C:\Program Files\Microsoft SQL Server\
    Error and Usage Reporting
      Usage Reporting: False
      Error Reporting: False
  Instance configuration
```

Configuration file path:

C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\20090317_103055\ConfigurationFile.ini

Check that what's being installed is what you want and then click on **Install** when you're sure you want to start the installation process:

Installation Progress: SQL Server 2008 will now install. How long it takes depends on the speed of your machine.

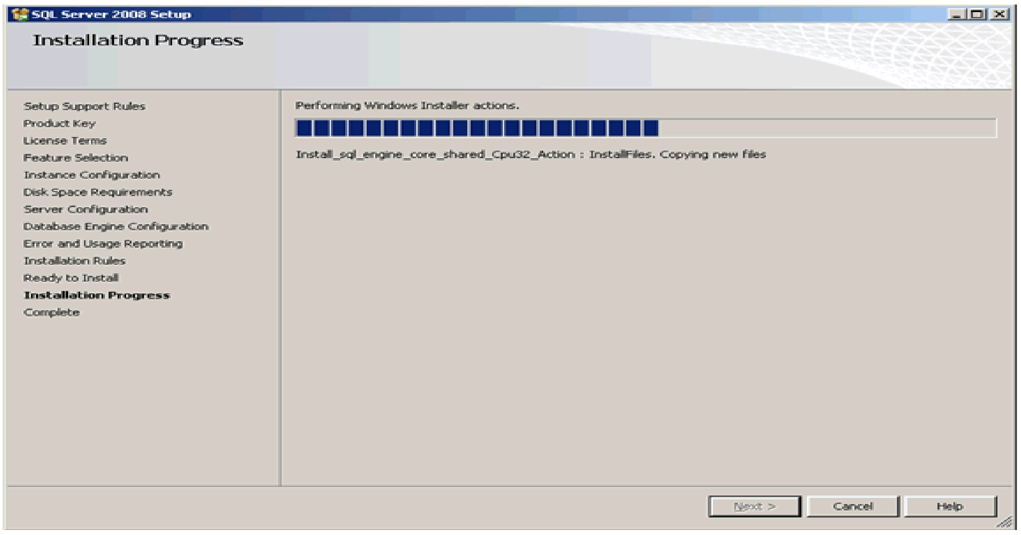

SQL Server 2008 Setup

Installation Progress

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Scheduling package installation.

…More Installation Progress


SQL Server 2008 Setup

Installation Progress

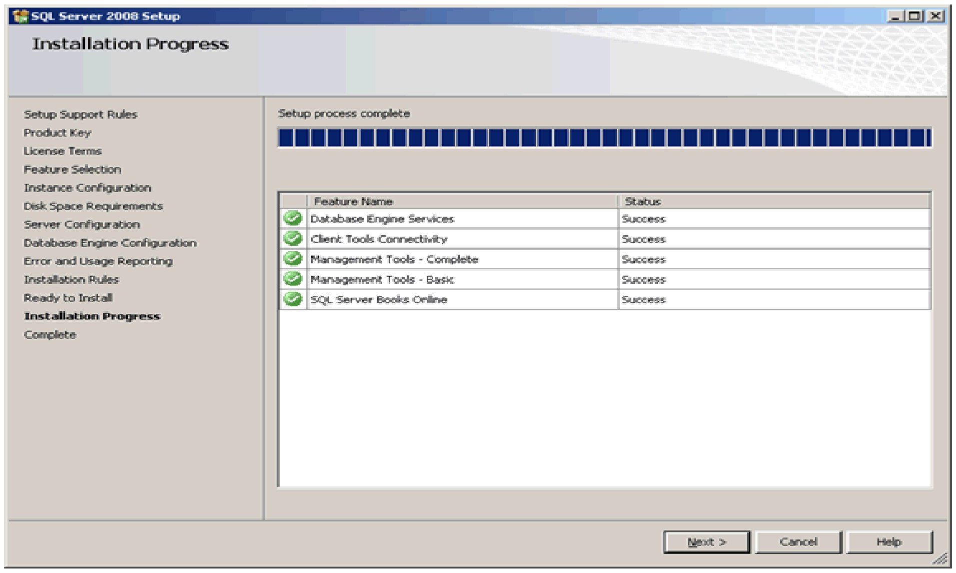
Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Performing Windows Installer actions.

Install_sql_engine_core_shared_Cpu32_Action : InstallFiles. Copying new files

... and Finally

Finally, the installation will complete:

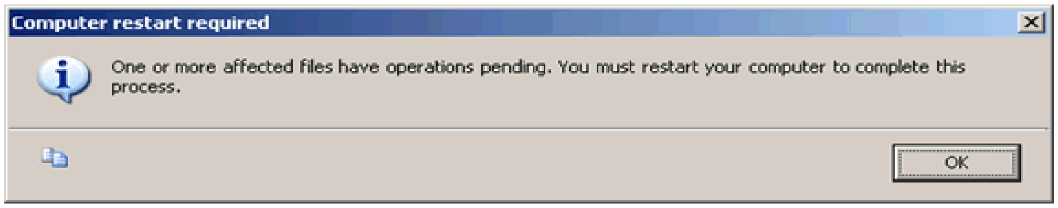

SQL Server 2008 Setup

Installation Progress

Setup process complete

Feature Name	Status
Database Engine Services	Success
Client Tools Connectivity	Success
Management Tools - Complete	Success
Management Tools - Basic	Success
SQL Server Books Online	Success

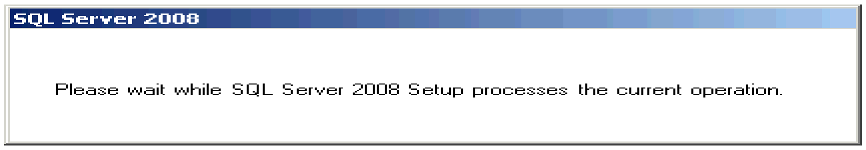
...and the following dialog box will appear:


Computer restart required

One or more affected files have operations pending. You must restart your computer to complete this process.

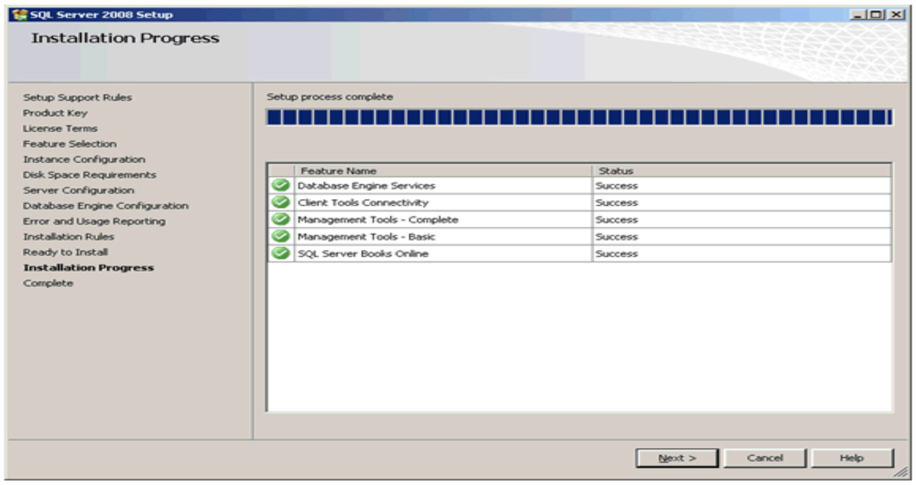
Click on OK, the machine will NOT reboot.

The following will appear:


SQL Server 2008

Please wait while SQL Server 2008 Setup processes the current operation.

…followed by:


SQL Server 2008 Setup

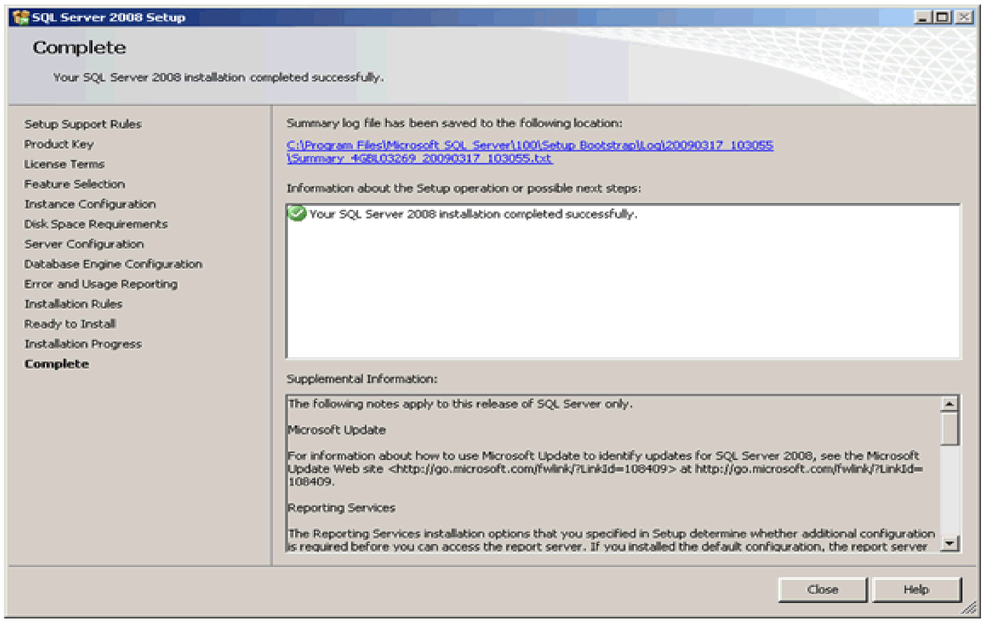
Installation Progress

Setup process complete

Feature Name	Status
Database Engine Services	Success
Client Tools Connectivity	Success
Management Tools - Complete	Success
Management Tools - Basic	Success
SQL Server Books Online	Success

Click on the Next button again...

STEP 18: Installation Complete. The following screen appears:



SQL Server 2008 Setup

Complete

Your SQL Server 2008 installation completed successfully.

Summary log file has been saved to the following location:

C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\20090317_103055\Summary_4GBL03269_20090317_103055.txt

Information about the Setup operation or possible next steps:

Your SQL Server 2008 installation completed successfully.

Supplemental Information:

The following notes apply to this release of SQL Server only.

Microsoft Update

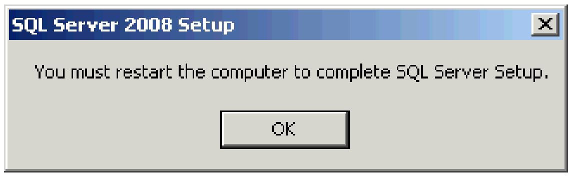
For information about how to use Microsoft Update to identify updates for SQL Server 2008, see the Microsoft Update Web site <http://go.microsoft.com/fwlink/?LinkId=108409> at http://go.microsoft.com/fwlink/?LinkId=108409.

Reporting Services

The Reporting Services installation options that you specified in Setup determine whether additional configuration is required before you can access the report server. If you installed the default configuration, the report server

It may be worth clicking on the installation log at the top of the screen to check everything's gone as expected. Not that this is MUCH smaller than the usual SQL Server installation log files of old.

Finally, **click on the Close button**. The following dialog will appear:

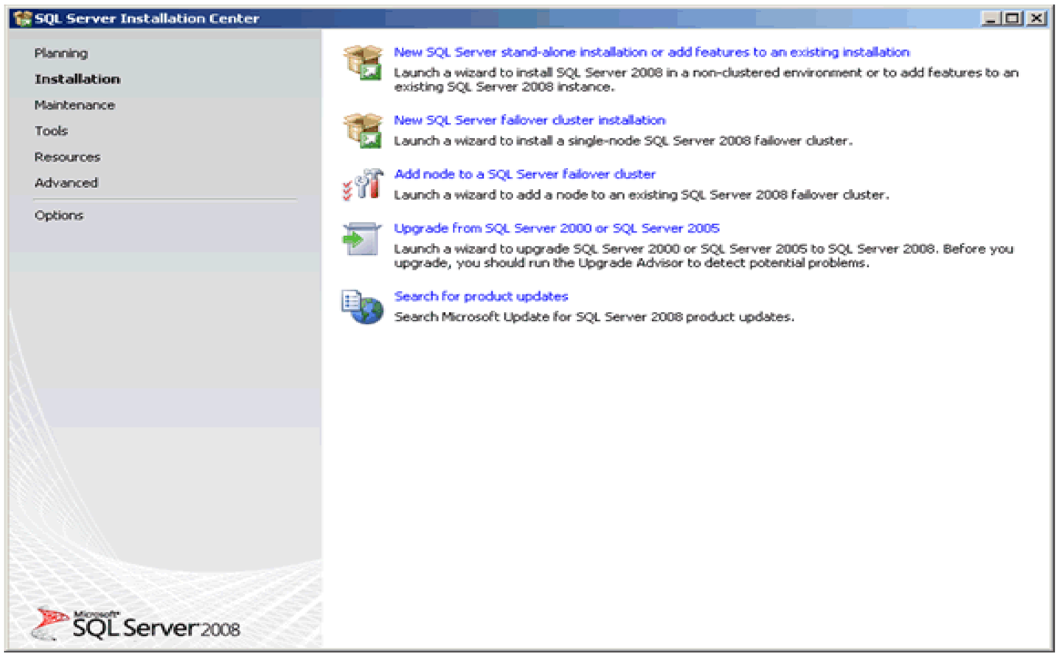


SQL Server 2008 Setup

You must restart the computer to complete SQL Server Setup.

Click on OK – your server will NOT re-boot at this point.

The dialog box will disappear and you will be returned to the Installation Center:


SQL Server Installation Center

New SQL Server stand-alone installation or add features to an existing installation
Launch a wizard to install SQL Server 2008 in a non-clustered environment or to add features to an existing SQL Server 2008 instance.

New SQL Server failover cluster installation
Launch a wizard to install a single-node SQL Server 2008 failover cluster.

Add node to a SQL Server failover cluster
Launch a wizard to add a node to an existing SQL Server 2008 failover cluster.

Upgrade from SQL Server 2000 or SQL Server 2005
Launch a wizard to upgrade SQL Server 2000 or SQL Server 2005 to SQL Server 2008. Before you upgrade, you should run the Upgrade Advisor to detect potential problems.

Search for product updates
Search Microsoft Update for SQL Server 2008 product updates.

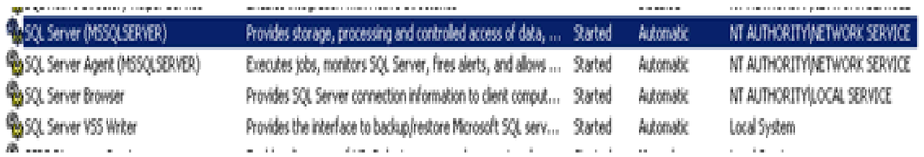
Check SQL Server 2008 has started. **Click on the Close button** (the "x") in the top right of the screen.

Finally, manually reboot your machine to complete the SQL Server 2008 installation.

Top Tips:How to check that SQL Server 2008 has installed correctly

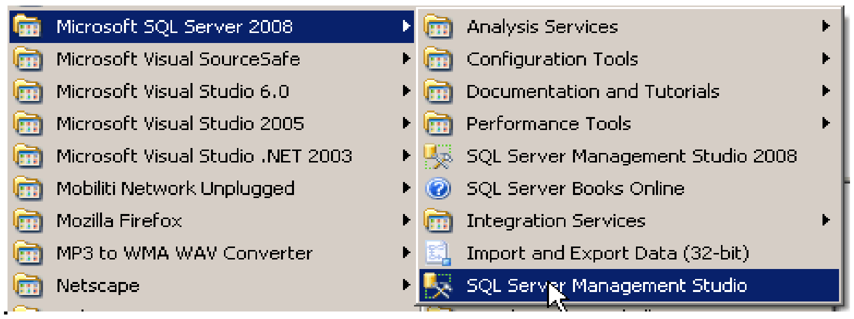
Here are a short number of post-installation checks which are useful to perform after re-booting your new SQL Server.

Check 1: Has the SQL Server Service Started

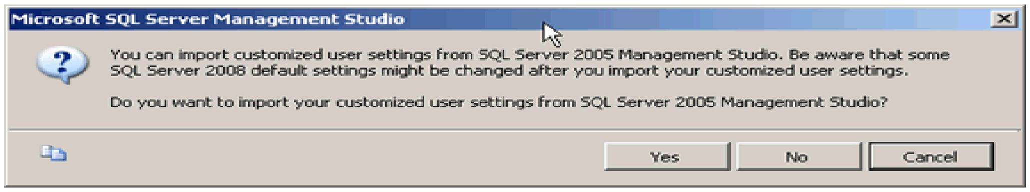


SQL Server (MSSQLSERVER)	Provides storage, processing and controlled access of data, ...	Started	Automatic	NT AUTHORITY\NETWORK SERVICE
SQL Server Agent (MSSQLSERVER)	Executes jobs, monitors SQL Server, fires alerts, and allows ...	Started	Automatic	NT AUTHORITY\NETWORK SERVICE
SQL Server Browser	Provides SQL Server connection information to client comput...	Started	Automatic	NT AUTHORITY\LOCAL SERVICE
SQL Server VSS Writer	Provides the interface to backup/restore Microsoft SQL serv...	Started	Automatic	Local System

Check 2: Does Management Studio Work? Check Management Studio works by firing it up.


Microsoft SQL Server 2008
Microsoft Visual SourceSafe
Microsoft Visual Studio 6.0
Microsoft Visual Studio 2005
Microsoft Visual Studio .NET 2003
Mobiliti Network Unplugged
Mozilla Firefox
MP3 to WMA WAV Converter
Netscape
Analysis Services
Configuration Tools
Documentation and Tutorials
Performance Tools
SQL Server Management Studio 2008
SQL Server Books Online
Integration Services
Import and Export Data (32-bit)
SQL Server Management Studio

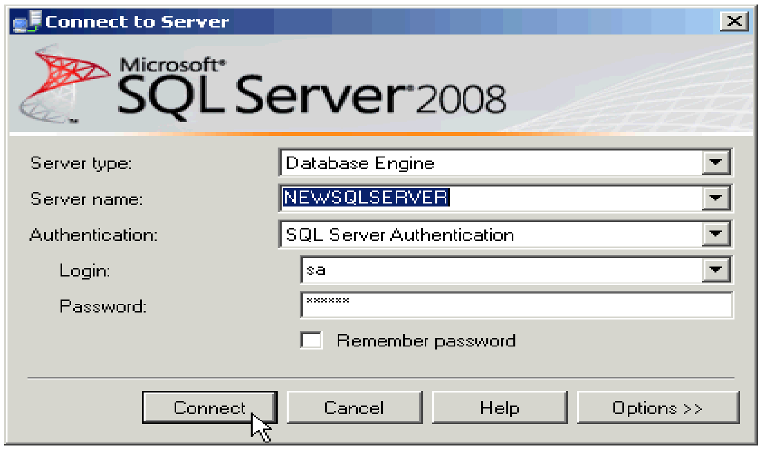
Click on NO when you see this dialog box:



Microsoft SQL Server Management Studio

You can import customized user settings from SQL Server 2005 Management Studio. Be aware that some SQL Server 2008 default settings might be changed after you import your customized user settings.

Do you want to import your customized user settings from SQL Server 2005 Management Studio?



Connect to Server

Microsoft® SQL Server® 2008

Server type: Database Engine

Server name: NEWSQLSERVER

Authentication: SQL Server Authentication

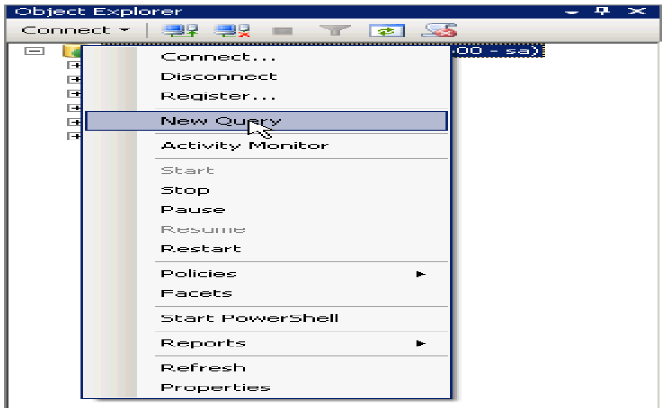
Login: sa

Password: *******

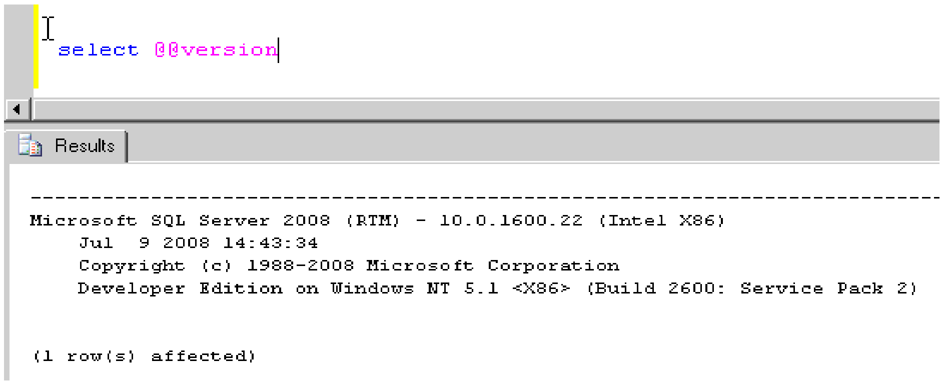
Remember password

Check 3: Can you run a basic query against the new SQL Server?

Check SQL Server works by running a simple query from Management Studio:



Enter the query shown below and hit F5 to run it:



```
select @@version
```

```
--------------------------------------------------------------------------------
Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 (Intel X86)
    Jul  9 2008 14:43:34
    Copyright (c) 1988-2008 Microsoft Corporation
    Developer Edition on Windows NT 5.1 <X86> (Build 2600: Service Pack 2)


(1 row(s) affected)
```

L #34	LO #2- Write SQL statements that use functions

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Using arithmetical operators with the correct precedence
- Using string functions and operators
- Using mathematical functions
- Using date functions
- Using SQL aggregate functions

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Use arithmetical operators with the correct precedence
- Use string functions and operators
- Use mathematical functions
- Use date functions
- Use SQL aggregate functions

Learning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If you earned a satisfactory evaluation proceed to “next information sheets

Information Sheet: 1 Using arithmetical operators with the correct precedence

2.1 Using arithmetical operators with the correct precedence

Arithmetical operators

Arithmetic operators perform mathematical operations on two expressions of one or more of the data types of the numeric data type category.

Arithmetic Operators

Arithmetic operators can perform arithmetical operations on numeric operands involved. Arithmetic operators are addition(+), subtraction(-), multiplication(*) and division(/). The + and - operators can also be used in date arithmetic.

Table 1

Operator	**Meaning**	**Operates on**
+ (Add)	Addition	Numeric value
- (Subtract)	Subtraction	Numeric value
* (Multiply)	Multiplication	Numeric value
/ (Divide)	Division	Numeric value
% (Modulo)	Returns the integer remainder of a division. For example, 17 % 5 = 2 because the remainder of 17 divided by 5 is 2.	Numeric value

Syntax:

SELECT<Expression>[arithmetic operator]< expression >FROM [table_name] WHERE]< [expression]:

Table 2

Parameter	**Description**
Expression	Expression made up of a single constant, variable, scalar function, or column name and can also be the pieces of a SQL query that compare values against other values or perform arithmetic calculations.
arithmetic operator	Plus(+), minus(-), multiply(*), and divide(/).

table_name	Name of the table.

Example: SQL Arithmetic Operators

Table 3

This is a simple example of using SQL arithmetic operators:

```
SELECT15+10-5*5/5FROM dual;
```

SQL minus (-) operator

The SQL minus (-) operator is used to subtract one expression or number from another expression or number.

SQL multiply (*)

The SQL multiply (*) operator is used to multiply two or more expressions or numbers.

SQL divide (/) operator

Parameter	**Description**
Expression	Expression made up of a single constant, variable, scalar function, or column name and can also be the pieces of a SQL query that compare values against other values or perform arithmetic calculations.
arithmetic operator	Plus(+), minus(-), multiply(*), and divide(/).
table_name	Name of the table.

The SQL divide (/) operator is used to divide one expressions or numbers by another.

SQL MODULO (%)operator

The SQL MODULO operator returns the remainder (an integer) of the division.

Example:

SELECT150%7;

Output 150%7;=3

Self-Check -1	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: II multiple choices

1. Which Operators have higher precedence?

Arithmetic Operators	LogicalOperators
Comparison Operators	BooleanOperators

2. What will be the outcome of the query given below?

```
SELECT 100+NULL+999 FROM dual;
```

A. 100 B. 999 C. NULL D. 1099

3. What is the maximum level of sub-queries allowed in Oracle in a single SQL statement?

A. 20 B. 50 C. Unlimited D. 255

4. The decrement operator is denoted by

A. ++ B. -- C. %% D. //

5. From the below operators, which one of them holds the highest precedence level?

A. Division (/)
B. Multiplication (*)
C. Brackets (())
D. Subtraction

Score = ____________

Rating: _____________

Name: ____________________________ Date: ________________

Short Answer Questions

Note: **Satisfactory rating –5 bpoints Unsatisfactory - below 5 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet: 2 Using string functions and operators

2.2 Using string functions and operators

- String functions perform manipulation of text that normally are contained in a table column.
- There is a differentiated range of built-in string functions in the various RDBMS, yet are there some functions which have the same name and execute the same.
- There are also functions that perform the same but do not have the same name in different RDBMS.

Here is a collection of the most important built-in string functions

Table 1

Function	RDBMS	Description
CONCAT()	SQL server, MySQL, Oracle, PostgreSQL	Returns a string that is the result of concatenating two or more string values. Oracle takes only two string values.
LEN()	SQL server	Returns the length of a string, measured in bytes.
LENGTH()	MySQL, Oracle, PostgreSQL	
LOWER()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character expression after converting uppercase character data to lowercase.
LTRIM()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character expression after it removes leading blanks. Oracle and PostgreSQL has an additional parameter that specifies what you want to remove (the default is leading blanks).
REPLACE()	SQL server, MySQL, Oracle, PostgreSQL	Replaces all occurrences of a specified string value with another string value.
RTRIM()	SQL server, MySQL, Oracle, PostgreSQL	Returns a character string after truncating all trailing blanks. Oracle and PostgreSQL has an additional

		parameter that specifies what you want to remove (the default is trailing blanks).
SUBSTR()	MySQL, Oracle, PostgreSQL	Returns a part of a string expression.
SUBSTRING()	SQL server, MySQL, PostgreSQL	
TRIM()	MySQL, Oracle	Returns a character string after truncating all leading blanks and trailing blanks. Oracle and PostgreSQL has additionally specifying what you want to remove leading, trailing or both (default is both and blanks). With SQL server you have to use a combination of LTRIM() and RTRIM().
UPPER()	SQL server, MySQL, Oracle	Returns a character expression with lowercase character data converted to uppercase.

Self-Check -2	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part:II multiple choice

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. ___________is used to sort the data in ascending or descending order, based on one or more columns.
   A. The SQL ORDER BY clause
   B. The SQL GROUP BY clause
   C. The SQL DISTINCT keyword
   D. All
   E. None
2. ____________ is used in collaboration with the SELECT statement to arrange identical data into groups.
   A. The SQL ORDER BY clause
   B. The SQL GROUP BY clause
   C. The SQL DISTINCT keyword
   D. All
   E. None
3. ____________is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.
   A. The SQL ORDER BY clause
   B. The SQL GROUP BY clause
   C. The SQL DISTINCT keyword
   D. All
4. What among the following is a type of single-row function?

A. VARCHAR2
B. Character
C. LONG
D. NULLIF

Score = ___________

Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

***Note:* Satisfactory rating - 4 points Unsatisfactory - below 4 points**

You can ask you teacher for the copy of the correct answers.

Information Sheet 3: Using mathematical functions

2.3 Using mathematical functions

Mathematical operators are provided for many PostgreSQL types. For types without common mathematical conventions for all possible permutations (e.g., date/time types) we describe the actual behavior in subsequent sections.

Table below shows the available mathematical functions. In the table, dp indicates double precision. Many of these functions are provided in multiple forms with different argument types. Except where noted, any given form of a function returns the same data type as its argument. The functions working with double precision data are mostly implemented on top of the host system's C library; accuracy and behavior in boundary cases may therefore vary depending on the host system.

Table Mathematical Functions

Table 1

Function	**Return Type**	**Description**	**Example**	**Result**
`abs(x)`	(same as `x`)	absolute value	`abs(-17.4)`	`17.4`
`cbrt(dp)`	`Dp`	cube root	`cbrt(27.0)`	`3`
`ceil(dp or numeric)`	(same as input)	smallest integer not less than argument	`ceil(-42.8)`	`-42`
`ceiling(dp or numeric)`	(same as input)	smallest integer not less than argument (alias for `ceil`)	`ceiling(-95.3)`	`-95`
`degrees(dp)`	`Dp`	radians to degrees	`degrees(0.5)`	`28.6478897565412`
`exp(dp or numeric)`	(same as input)	Exponential	`exp(1.0)`	`2.71828182845905`
`floor(dp or numeric)`	(same as input)	largest integer not greater than argument	`floor(-42.8)`	`-43`
`ln(dp or numeric)`	(same as input)	natural logarithm	`ln(2.0)`	`0.693147180559945`
`log(dp or numeric)`	(same as input)	base 10 logarithm	`log(100.0)`	`2`
`log(b numeric, x`	`Numeric`	logarithm to base `b`	`log(2.0, 64.0)`	`6.0000000000`

numeric)				
mod(y, x)	(same as argument types)	remainder of y/x	mod(9,4)	1
pi()	Dp	"π" constant	pi()	3.14159265358979
power(a dp, b dp)	Dp	a raised to the power of b	power(9.0, 3.0)	729
power(a numeric, b numeric)	Numeric	a raised to the power of b	power(9.0, 3.0)	729
radians(dp)	Dp	degrees to radians	radians(45.0)	0.785398163397448
random()	Dp	random value between 0.0 and 1.0	random()	
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(v numeric, s int)	Numeric	round to s decimal places	round(42.4382, 2)	42.44
setseed(dp)	Int	set seed for subsequent random() calls	setseed(0.54823)	1177314959
sign(dp or numeric)	(same as input)	sign of the argument (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(v numeric, s int)	Numeric	truncate to s decimal places	trunc(42.4382, 2)	42.43
width_bucket(op numeric, b1 numeric, b2 numeric, count	Int	return the bucket to which operand would be assigned in an equidepth	width_bucket(5.35, 0.024, 10.06, 5)	3

`int)`		histogram with `count` buckets, an upper bound of `b1`, and a lower bound of `b2`		

Assigning names to result columns

When the result of a SELECT statement is determined, you can specify your own names for the result table columns.

- The AS clause can be used to assign a different name, or alias, to the result set column. This can be done to increase readability.
- This capability is particularly useful for a column that is derived from an expression or a function.

Example: - SELECT SALARY+BONUS+COMM AS TOTAL_SAL FROM EMPLOYEE;
- select ID 'identifier',fname 'first name',Lname as last_name from EMPLOYEE;

Self-Check -3	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: II MULTIPLE CHOICES

1. A simple mathematical function in sql that takes a single argument and returns the nearest integer,is a

   A. Round() B. Abs() C. Max() D. floor()

2. A simple mathematical function in sql that takes a single argument and returns the largest that is less than or equal to that argumentr,is a

   A. ceil() B. Abs() C. Max() D. floor()

3. Abs() stands for

A. Arithmetic value	C. Assignment value
B. Absolute value	D. None of them

4. The Ceiling function is denoted by

A. Cel	B. Ceiling	C. Ceil	D. Cell

Score = ___________

Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

Note: **Satisfactory rating - 4 points Unsatisfactory - below 4 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet 4: Using date functions

2.4 Using date functions

Date and Time Functions in SQL Server 2008

GETDATE ()

The GETDATE () function returns the current date and time from the SQL Server.

Syntax: select GETDATE ()

Example: The following SELECT statement:

SELECT GETDATE () AS CurrentDateTime

Will result in something like this:

CurrentDateTime
2:45:34.243

Note: The time part above goes all the way to milliseconds.

Example: - The following SQL creates an "Orders" table with a datetime column (OrderDate):

CREATE TABLE Orders(OrderIdint NOT NULL PRIMARY KEY,ProductNamevarchar(50) NOT NULL,OrderDatedatetime NOT NULL DEFAULT GETDATE())

Notice that the OrderDate column specifies GETDATE() as the default value. As a result, when you insert a row into the table, the current date and time are automatically inserted into the column.

Now we want to insert a record into the "Orders" table:

INSERT INTO Orders (ProductName) VALUES ('Jarlsberg Cheese')\

The "Orders" table will now look something like this:

Table 1

OrderId	**ProductName**	**OrderDate**
1	Jarlsberg Cheese	2008-11-11 13:23:44.657

SYSDATETIME()

This function works the same as GETDATE(); it returns date and time. The difference in both functions is that SYSDATETIME returns a higher level of precision and also returns the newer **datetime2** data type.

select SYSDATETIME()

DAY()

This function returns an integer representing the day part of the specified date.

select DAY(<date>)

MONTH()

This function returns an integer representing the month part of the specified date.

select MONTH(<date>)

YEAR()

This function returns an integer representing the year part of the specified date.

select YEAR(<date>)

DATEADD()

The DATEADD() function adds or subtracts a specified time interval from a date.

Syntax: - Select DATEADD(datepart,number,date)

Where date is a valid date expression and number is the number of interval you want to add. The number can either be positive, for dates in the future, or negative, for dates in the past.

datepart can be one of the following:

Example:-**Assume we have the following "Orders" table:**

Table 2

OrderId	ProductName	OrderDate
1	Jarlsberg Cheese	2008-11-11 13:23:44.657

Now we want to add 45 days to the "OrderDate", to find the payment date.

We use the following SELECT statement:

SELECT OrderId,DATEADD(day,45,OrderDate) AS OrderPayDateFROM Orders

Result:

Table 3

OrderId	OrderPayDate
1	2008-12-26 13:23:44.657

DATEDIFF()

This function returns the difference between two specified dates in a specified unit of time.

DATEDIFF(<datepart>, <startdate>, <enddate>)

Syntax:- select DATEDIFF(datepart,startdate,enddate)

Where startdate and enddate are valid date expressions and datepart can be one of the following:

Example: -**Now we want to get the number of days between two dates.**

We use the following SELECT statement:

SELECT DATEDIFF(day,'2008-06-05','2008-08-05') AS DiffDate

Result:

Table 3

DiffDate
61

Example:-Now we want to get the number of days between two dates (notice that the second date is "earlier" than the first date, and will result in a negative number).

We use the following SELECT statement:

SELECT DATEDIFF(day,'2008-08-05','2008-06-05') AS DiffDate

Result: **Table 4**

DiffDate
-61

DATENAME()

This function returns a string representing the name of the specified datepart of the specified date.

select DATENAME(<datepart>, <date>)

DATEPART()

The DATEPART() function is used to return a single part of a date/time, such as year, month, day, hour, minute, etc.

Syntax: - select DATEPART(datepart,date)

Where date is a valid date expression and datepart can be one of the following:

Table 5

Datepart	Abbreviation
Year	yy, yyyy
Quarter	qq, q
Month	mm, m
Dayofyear	dy, y
Day	dd, d
Week	wk, ww
Weekday	dw, w
Hour	Hh
Minute	mi, n
Second	ss, s
Millisecond	Ms
Microsecond	Mcs
Nanosecond	Ns

Self-Check -4	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: II multiple choice

1. What is the default date format in Oracle?
   A. DD-MON-YY
   B. DD-MON-YYYY
   C. DD-MM-RR
   D. DD-MON-RR
2. Which of the following data type is assigned to Substitution variables?
   A. VARCHAR2
   B. DATE
   C. NO DATA TYPE
   D. NUMBER
3. Which of the following WHERE conditions will list employees who were hired on current date?
   A. WHERE sysdate-hiredate=0
   B. WHERE sysdate=hiredate
   C. WHERE sysdate-hiredate<1
   D. WHERE to_date (sysdate,'DD-MON-YYYY') = to_date (hiredate='DD-MON-YYYY')
4. What of the following are the valid formats of date literals which can be used in WHERE clause?
   A. 24/Mar/95
   B. 02-12-1983
   C. 19-JUN-2001
   D. 31.04.2010

Score = ___________
Rating: ____________

Name: _________________________ Date: _______________

***Note:* Satisfactory rating - 4 points Unsatisfactory - below 4 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet 5 Using SQL aggregate functions

2.5 Using SQL aggregate functions

Aggregate functions perform a calculation on a set of values and return a single value.

There is a differentiated range of built-in aggregate functions in the various RDBMS, yet are there some functions which have the same name and execute the same.

There are also functions that perform the same but do not have the same name in different RDBMS.

Here is a collection of the most important built-in aggregate functions:

Aggregate Functions are:

MIN() returns the smallest value in a given column

MAX() returns the maximum value in a given column.

SUM() returns the sum of the numeric values in a given column

AVG() returns the average value of a given column

COUNT() returns the total number of values in a given column

COUNT(*) returns the number of rows in a table

AVG () FUNCTION

Aggregates the average of a value in a column.

Function syntax:

AVG ([distinct | all] <expression>)

/* all is the default */

AVG Example:

Select avg(item_qty*PRICE) "Average Order sum" from order_books;

COUNT Example:

Select count (telephone_number), count (*) from telephones

Two types of count () function:

COUNT can take an expression as an argument and discovers all non-null occurrences of that argument in a table.

COUNT (*) has an asterisk as an argument and selects all rows, even if some columns contain a NULL value.

COUNT Example:

Select count(telephone_number), count(*) from telephones

MAX () FUNCTION

Aggregates the largest value in a column

Function syntax:

MAX ([distinct | all] <expression>)

/* all is the default */

MAX Example:

Select MAX (item_qty*PRICE) "Max. Order sum"fromorder_books;

MIN () FUNCTION

Aggregates the smallest value in a column

Function syntax:

MIN ([distinct | all] <expression>)

/* all is the default */

MIN Example:

Select MIN (item_qty*PRICE) "Min. Order sum"fromorder_books;

SUM () FUNCTION

Aggregates the sum of the values in a column

Function syntax:

SUM ([distinct | all] <expression>)

/* all is the default */

SUM Example:

Select SUM (item_qty*PRICE) "Total Ordered" from order_books;

Self-Check -5	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part II: write the following queries/script of the statement given below:

ord_no	purch_amt	ord_datesalesman_id	customer_id
70001	150.5	2012-10-05 3005	5002
70009	270.65	2012-09-10 3001	5005
70002	65.26	2012-10-05 3002	5001
70004	110.5	2012-08-17 3009	5003
70007	948.5	2012-09-10 3005	5002
70005	2400.6	2012-07-27 3007	5001
70008	5760	2012-09-10 3002	5001
70010	1983.43	2012-10-10 3004	5006
70003	2480.4	2012-10-10 3009	5003
70012	250.45	2012-06-27 3008	5002
70011	75.29	2012-08-17 3003	5007
70013	3045.6	2012-04-25 3002	5001

1. Write a SQL statement to find the total purchase amount of all orders.
2. Write a SQL statement to find the average purchase amount of all orders.
3. Write a SQL statement to find the number of salesmen currently listing for all of their customers.

Score = ____________
Rating: _____________

Name: ___________________________ Date: _________________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

L #35	LO #3- Write SQL statements that use aggregation and filtering

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Using 'Group by' to aggregate data by multiple columns
- Sorting aggregated data
- Filtering aggregated data using the 'having' clause

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Use 'Group by' to aggregate data by multiple columns
- Sort aggregated data
- Filter aggregated data using the 'having' clause

Learning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If you earned a satisfactory evaluation proceed to “next information sheets

Information Sheet: 1 Using 'Group by' to aggregate data by multiple columns

3.1 Using 'Group by' to aggregate data by multiple columns

When an aggregate function is executed, SQL Server summarizes values for an entire table or for groups of columns within the table, producing a single value for each set of rows for the specified columns.

- You can use aggregate functions with the SELECT statement or in combination with the **GROUP BY** clause
- Use the **GROUP BY** clause on columns or expression to organize rows into groups and to summarize those groups. The **GROUP BY** clause groups rows on the basis of similarities between them.

When you use the **GROUP BY** clause, consider the following facts and guidelines:

- SQL Server returns only single rows for each group that you specify; it does not return detail information.
- All columns that are specified in the GROUP BY clause must be included in the select list.
- If you include a WHERE clause, SQL Server groups only the rows that satisfy the search conditions.
- Do not use the GROUP BY clause on columns that contain multiple null values.

Example: For each department, retrieve the department name, the number of employees in the department, and their average salary.

```
SELECT DEPARTEMENT.Dname , COUNT (*)'number of employee', AVG
(EMPLOYEE.salary)'average salary' FROM EMPLOYEE,DEPARTEMENT
where EMPLOYEE.dnum =DEPARTEMENT.Dnumber
GROUP BY DEPARTEMENT.Dname
```

Displaying Data from Multiple Tables

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Consider the following two tables −

Table 1 − CUSTOMERS Table

ID NAME	AGE	ADDRESS	SALARY
1 Ramesh	32	Ahmedabad	2000.00
2 Khilan	25	Delhi	1500.00
3 kaushik	23	Kota	2000.00
4 Chaitali	25	Mumbai	6500.00
5 Hardik	27	Bhopal	8500.00
6 Komal	22	MP	4500.00
7 Muffy	24	Indore	10000.00

Table 2 − ORDERS Table

OID	DATE	CUSTOMER_ID AMOUNT
102	2009-10-08 00:00:00	3 3000
100	2009-10-08 00:00:00	3 1500
101	2009-11-20 00:00:00	2 1560
103	2008-05-20 00:00:00	42060

Now, let us join these two tables in our SELECT statement as shown below.

```
SQL> SELECT ID, NAME, AGE, AMOUNT
```

```
  FROM          CUSTOMERS,                    ORDERS
WHERE  CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result.

ID	NAME	AGE	AMOUNT
3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol.

There are different types of joins available in SQL −

- INNER JOIN − returns rows when there is a match in both tables.
- LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN − returns rows when there is a match in one of the tables.
- SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

2. Self-Check -1	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part III:multiple choices

1. Choose the correct statements about the GROUP BY clause.

1. Column alias can be used in the GROUP BY clause.
2. GROUP BY column must be in the SELECT clause.
3. GROUP BY clause must appear together with HAVING clause a SELECT query.
4. GROUP BY clause must appear after WHERE clause in a SELECT query.

2. Which of the following is NOT a GROUP BY function?

1. MAX 2. MIN 3. NVL 4. AVG

3. Which of the following functions can be used without GROUP BY clause in SELECT query?

1. COUNT 3. MIN 5. All
2. MAX 4. AVG

4. What are the appropriate data types accepted by GROUP BY functions?

1. Nested Tables 3. CLOB
2. NUMBER 4. DATE

5. Which of the following is NOT a GROUP BY extensions in SQL?

A. GROUP BY C. CUBE
B. GROUPING SETS D. ROLLUP

Score = ____________
Rating: _____________

Name: ___________________________ Date: ________________

Short Answer Questions

***Note:* Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet: 2 Sorting aggregated data

3.2 Sorting aggregated data

SQL allows the user to sort rows in the result set in ascending (ASC) or descending (DESC) order using

ORDER BY clause. Sort is in ascending order by default

- You can sort by column names, computed values, or expressions

Example: Retrieve a list of employees and ordered them alphabetically by their department name, last name, first name.

SELECT EMPLOYEE.ID, EMPLOYEE.Fname, EMPLOYEE.Lname, EMPLOYEE.salary, DEPARTEMENT.Dname FROM DEPARTEMENT,EMPLOYEE WHERE EMPLOYEE.dnum=DEPARTEMENT .Dnumber ORDER BY DNAME, LNAME, FNAME;

Table 1

AGENT_CODE	AGENT_NAME	WORKING_AREA	COMMISSION	PHONE_NO	COUNTRY
A007	Ramasundar	Bangalore	0.15	077-25814763	
A003	Alex	London	0.13	075-12458969	
A008	Alford	New York	0.12	044-25874365	
A011	Ravi Kumar	Bangalore	0.15	077-45625874	
A010	Santakumar	Chennai	0.14	007-22388644	
A012	Lucida	San Jose	0.12	044-52981425	
A005	Anderson	Brisban	0.13	045-21447739	
A001	Subbarao	Bangalore	0.14	077-12346674	
A002	Mukesh	Mumbai	0.11	029-12358964	
A006	McDen	London	0.15	078-22255588	
A004	Ivan	Torento	0.15	008-22544166	
A009	Benjamin	Hampshair	0.11	008-22536178	

SELECTworking_area,AVG(commission),COUNT(agent_name)FROM AGENTS HAVINGCOUNT(agent_name)<3GROUPBYworking_areaORDERBYAVG(commission),COUNT(agent_name)DESC;

SELECTagent_name,working_area,commission FROM AGENTS WHERE commission<=.13 ORDERBY2DESC;

Self-Check -2	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: III Multiple choice

1. One of the following is not true about The GROUP BY Statement
   A. Aggregate functions often need an added GROUP BY statement.
   B. Some database sorts query results in ascending order by default.
   C. The GROUP BY statement is used in conjunction with the aggregate functions to group
   D. All
   E. None

Score = ___________
Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet3: Filtering aggregated data using the 'having' clause

3.3 Filtering aggregated data using the 'having' clause

Use the **HAVING** clause on columns or expressions to set conditions on the groups included in a result set.

When you use the HAVING clause, consider the following facts and guidelines:

- Use the HAVING clause only with the GROUP BY clause to restrict the grouping.
- Using the HAVING clause without the GROUP BY clause is not meaningful.

Example: For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.

```
SELECT DNAME, Fname, COUNT (*) FROM DEPARTMENT, EMPLOYEE
WHERE DNUMBER=DNO AND SALARY>40000
GROUP BY DNAME DESC, Fname ASC
 HAVING COUNT (*) > 5;
```

Essentially,the HAVING and WHERE clauses do the same thing, that is filter rows from inclusion in a result table based on a condition. While it may appear that a HAVING clause filters out groups, it does not.Rather,a HAVING clause filters rows.

When all rows for a group are eliminated so is the group.To summarize, the important differences between the WHERE and HAVING clauses are:

A WHERE clause is used to filter rows BEFORE the GROUPING action (i.e., before the calculation of the aggregate functions).

A HAVING clause filters rows AFTER the GROUPING action (i.e., after the calculation of the aggregate functions).

```
SELECT JOB_ID,SUM (SALARY)FROM employeesGROUP BY JOB_IDHAVING SUM (SALARY) >
10000;
```

Self-Check -3	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part III: Multiple choices

1. Which statements are true regarding the WHERE and HAVING clauses in a SELECT statement?
   A. The HAVING clause can be used with group functions in subqueries.
   B. The WHERE clause can be used to exclude rows after dividing them into groups.
   C. The WHERE clause can be used to exclude rows before dividing them into groups.
   D. The WHERE and HAVING clauses can be used in the same statement only if they are applied to different columns in the table.
2. Choose the correct statements about the HAVING clause.
   A. The HAVING clause is an optional clause in SELECT statement.
   B. The HAVING clause is a mandatory clause if SELECT statement uses a GROUP BY clause.
   C. The HAVING clause can appear in a SELECT statement only if it uses a GROUP BY clause.
   D. The HAVING clause is a mandatory clause if SELECT statement uses a GROUP BY clause.
3. which of the following clauses represent valid uses of group functions?
   A. All
   B. ORDER BY AVG(salary)
   C. HAVING MAX(salary) > 10000
   D. SELECT AVG(NVL(salary, 0))

   A. Subquery must be placed in the outer query's HAVING clause if:
   B. The inner query needs to reference the value returned to the outer query.
   C. The value returned by the inner query is to be compared to grouped data in the outer query.
   D. The subquery returns more than one value to the outer query.
   E. None of the above. Subqueries can't be used in the outer query's HAVING clause.

Name: _________________________Date: _______________

Score = ____________
Rating: _____________

Short Answer Questions

Note: **Satisfactory rating - 6 points Unsatisfactory - below 6 points**
You can ask you teacher for the copy of the correct answers.

L #36	LO #4- Write and execute SQL sub-queries

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Constructing single and nested sub-queries
- Constructing sub-queries that return a single row, and multiple rows
- Using correlated sub-queries
- Writing sub-queries that use aggregates

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Constructing single and nested sub-queries
- Constructing sub-queries that return a single row, and multiple rows
- Using correlated sub-queries
- Writing sub-queries that use aggregates

Learning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If you earned a satisfactory evaluation proceed to “proceed to next information sheer

Information Sheet1: constructing single and nested sub-queries

4.1 Constructing single and nested sub-queries

A **query** is a request for information from a database (Queries are Questions).

Single query is a Single Block query.

Example: SELECT distinct salary FROM Employee where Gender ='female'

A SQL **nested query** is a SELECT query that is nested inside a SELECT, UPDATE, INSERT, or DELETE SQL query.

Nested Queries are queries that contain another complete SELECT statements nested within it, that is, in the WHERE clause.

- The nested SELECT statement is called an “inner query” or an “inner SELECT.”
- The main query is called “outer SELECT” or “outer query.”
- The use of nested query in this case is to avoid explicit coding of JOIN which is a very expensive database operation and to improve query performance.

Example: SELECT ID,LNAME, FNAME FROM EMPLOYEE
WHERE (SELECT COUNT (*) FROM DEPENDENTED WHERE DEPENDENTED.EmpID =EMPLOYEE.ID) >= 2

Subqueries

A subquery is a query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement, or inside another subquery.

- Subquery is an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.
- You use subqueries to break down a complex query into a series of logical steps and, as a result, to solve a problem with single statements.
- Each select statement in the subquery has its own:
    - ✓ select list
    - ✓ where clause

Example: For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

```
SELECT DNUMBER, COUNT (*) FROM DEPARTMENT, EMPLOYEE
  WHERE DNUMBER=DNO AND SALARY>40000 AND
       DNO IN (SELECT DNO FROM EMPLOYEE
                  GROUP BY DNO
                  HAVING COUNT (*) > 5)
       GROUP BY DNUMBER;
```

Self-Check -1	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: IV

1. Which of the following operators cannot be used in a sub-query?

   A. AND
   B. <
   C. >
   D. <>

2. Which of the following are the types of sub-queries?

A. Ordered sub-queries
B. Grouped sub-queries
C. Single row sub-queries
D. None of the above

3. Which of the following is true about sub-queries?
   A. They execute after the main query executes
   B. They execute in parallel to the main query
   C. The user can execute the main query and then, if wanted, execute the sub-query
   D. They execute before the main query executes.
4. Which of the following is true about the result of a sub-query?
   A. The result of a sub-query is generally ignored when executed.
   B. The result of a sub-query doesn't give a result, it is just helpful in speeding up the main query execution
   C. The result of a sub-query is used by the main query.
   D. The result of a sub-query is always NULL

Name: ______________________Date: _______________

Score = ___________

Rating: ____________

Short Answer Questions

Note: **Satisfactory rating - 3 points** **Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet 2: constructing sub-queries that return a single row, and multiple rows

4.2 Constructing sub-queries that return a single row, and multiple rows

A sub-query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

- **Single Row Sub Query**

A single-row sub-query is used when the outer query's results are based on a single, unknown value. Although this query type is formally called "single-row," the name implies that the query returns multiple columns-but only one row of results. However, a single-row subquery can return only one row of results consisting of only one column to the outer query.

In the below SELECT query, inner SQL returns only one row i.e. the minimum salary for the company. It in turn uses this value to compare salary of all the employees and displays only those, whose salary is equal to minimum salary.

```
SELECT first_name, salary, department_id FROM employees
WHERE salary = (SELECT MIN (salary)  FROM employees);
```

A HAVING clause is used when the group results of a query need to be restricted based on some condition. If a sub-query's result must be compared with a group function, you must nest the inner query in the outer query's HAVING clause.

```
SELECT department_id, MIN (salary)FROM employeesGROUP BY department_id
HAVING MIN (salary) < (SELECT AVG (salary)FROM employees)
```

- **Multiple Row Sub Query**

Multiple-row sub-queries are nested queries that can return more than one row of results to the parent query. Multiple-row sub-queries are used most commonly in WHERE and HAVING clauses. Since it returns multiple rows, it must be handled by set comparison operators (IN, ALL, ANY).While IN operator holds the same meaning as discussed in earlier chapter, ANY operator compares a specified value to each value returned by the sub query while ALL compares a value to every value returned by a sub query.

Below query shows the error when single row sub query returns multiple rows.

```
SELECT  first_name, department_id
FROM employees
WHERE department_id = (SELECT department_id
                        FROM employees
                        WHERE LOCATION_ID = 100)
```

department_id = (select * ERROR at line 4: ORA-01427: single-row subquery returns more than one row

Operators in sub queries

The IN, ALL, and ANY operators in subquery.

- **IN operator**

The IN operator is an operator that allows you to specify multiple values in a WHERE clause.

A row from a table satisfies a condition with the IN operator if the value of a particular column occurs in a set of expressions. The expressions in such a set entered one by one by a user.

Syntax to use **IN** operator**:** SELECT column_name(s)

```
FROM table_name
WHERE column_name IN (value1,value2,...)
```

Examle: SELECT * FROM Persons WHERE LastName IN ('Aster','Maru')

```
SELECT * FROM employee
        WHERE Dnum IN(SELECT Dnum FROM DEPARTEMENT
                      WHERE Dname = 'Research');
```

- **Any operator**

ANY operator is an operator that compares a value to each value in a list or results from a query and evaluates to true if the result of an inner query contains at least one row. **ANY** must be preceded by comparison operators.

Syntax to use **Any** operator**:** SELECT column_name(s)

```
FROM table_name
WHERE column_name =any(value1,value2,...)
```

Example: SELECT ID ,fname,lname FROM EMPLOYEE

```
where Dnum =any(select COUNT(*) from DEPARTEMENT
        where Dnumber='ict001')
```

- **All operators**

ALL operator is used to select all records of a SELECT STATEMENT. It compares a value to every value in a list or results from a query. The **ALL** must be preceded by the comparison operators and evaluates to **TRUE** if the query returns no rows. For example, **ALL** means greater than every value, means greater than the maximum value. Suppose **ALL** (1, 2, 3) means greater than 3.

Syntax to use **All** operator**:** SELECT column_name(s)

```
FROM table_name
WHERE column_name =all(value1,value2,...)
```

Enample: SELECT LNAME, FNAME FROM EMPLOYEE

```
WHERE SALARY> ALL (SELECT SALARY FROM EMPLOYEE
          WHERE Dnum=5);
```

- **SOME operators**

SOME operator is the same as ANY operator (SOME and ANY perform the same function).

SOME compares a value to each value in a list or results from a query and evaluates to true if the result of an inner query contains at least one row.

SOME must match at least one row in the subquery and must be preceded by comparison operators.

Example: SELECT ID,Fname,Lname FROM EMPLOYEE

```
WHERE ID =SOME(SELECT EmpID FROM DEPENDENTED
        WHERE Gender='Female');
```

- **EXISTS operator**

The **EXIST operator** checks the existence of a result of a subquery. The **EXISTS**subquery tests whether a subquery fetches at least one row. When no data is returned then this operator returns **'FALSE'**.

- A valid **EXISTS**subquery must contain an outer reference and it must be a correlated subquery.

- You can use the EXISTS and NON EXISTS operators to determine whether data exists in a list of values.
- Use the EXISTS and NOT EXISTS operators with correlated subqueries to restrict the result set of an outer query to rows that satisfy the subquery.

Example: SELECT DepID ,Fname ,Gender FROM DEPENDENTED

```
        WHERE exists(SELECT ID FROM EMPLOYEE
                    WHERE   EMPLOYEE.ID  =DEPENDENTED.EmpID  and  DEPENDENTED  .Gender
='Female')
```

You can use the EXISTS operator with the group by as wel as order by clause to determine whether data exists in a list of values.

Example: SELECT dnum,COUNT (*)'Number of employee who has dependent' FROM EMPLOYEE

```
        WHERE salary<3000 and exists (SELECT EmpID FROM DEPENDENTED
                                    WHERE EMPLOYEE.ID =DEPENDENTED.EmpID
                                    group by EmpID)
          group by Dnum
          order by Dnum
```

- **having clause**

Use the HAVING clause on columns or expressions to set conditions on the groups included in a result set.

The HAVING clause sets conditions on the GROUP BY clause in much the same way that the WHERE clauses interacts with the SELECT statement.

Example: SELECT dnum ,count(*)'number of employee' FROM EMPLOYEE

```
        WHERE  exists (SELECT dname FROM DEPARTEMENT
                    WHERE EMPLOYEE.Dnum =DEPARTEMENT.Dnumber)
        group by Dnum
        having COUNT(*)>2
```

Self-Check -2	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part: IV

4. Which of the following methods is used for writing a query with columns from multiple tables?
   - A. SELECT
   - B. GROUP BY
   - C. ORDER BY
   - D. JOINS
5. Which of the following is one of the most common methods to join multiple tables?
   - A. Hash Join
   - B. Equijoin
   - C. Self-Join
   - D. Cross Join
6. Which of the following can be used to fetch rows from multiple tables in a single SQL query?
   - A. SELECT
   - B. WHERE
   - C. FROM
   - D. Equi-joins
7. Which of the following are valid multi row operators used for sub-queries?
   - A. <=
   - B. ANY >=
   - C. !=
   - D. >=

Score = ___________

Rating: ____________

Name: _________________________ Date: _______________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet 3: Using correlated sub-queries

4.3: Using correlated sub-queries

The name of correlated sub-queries means that a sub-query is correlated with the outer query. The correlation comes from the fact that the sub-query uses information from the outer query and the sub-query executes once for every row in the outer query.

A correlated sub-query can usually be rewritten as a join query. Using joins enables the database engine to use the most efficient execution plan. The query optimizer is more mature for joins than for sub-queries, so in many cases a statement that uses a sub-query should normally be rephrased as a join to gain the extra speed in performance.

Note that alias must be used to distinguish table names in the SQL query that contains correlated sub-queries.

Practice #1: Using correlated sub-query.

By examining the query in this practice, we can sum up the following steps that the database engine takes to evaluate the correlated sub-query. It demonstrates that the sub-query uses data from the outer query and the sub-query executes once for every row in the outer query.

- The outer query passes a value for ProductID to the sub-query. It takes place in the WHERE clause in the sub-query [where a.ProductID = b.ProductID]
- The sub-query uses this passed-in ProductID value to look up the max unit price for this product
  [select max(UnitPrice) from order_details]
- When the max unit price for the product is found in the sub-query, it's returned to the outer query.
- The outer query then uses this max unit price in its WHERE clause to match unit price in order_details table for this product [where a.UnitPrice =]
- When the row is found, query engine temporarily holds the row in memory. It's guaranteed that a row will be found because both outer query and sub-query use the same table - order_details.

- The query engine then moves onto next row in the order_details table and repeat Step 1 to 3 again for the next product.
- When all products in order_details have been evaluated, it does a sorting and then returns the query result.

select distinct a.ProductID,a.UnitPrice as Max_unit_price_sold from order_details as a where a.UnitPrice = (select max(UnitPrice) from order_details as b where a.ProductID = b.ProductID) order by a.ProductID;

Self-Check -3	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part:IV

1. One of the following is not true about The GROUP BY Statement
   A. Aggregate functions often need an added GROUP BY statement.
   B. Some database sorts query results in ascending order by default.
   C. The GROUP BY statement is used in conjunction with the aggregate functions to group
   D. All
2. What is true about co-related sub-queries?
   A. The tables used in the main query are also used in a co-related sub-query
   B. The sub-queries which reference a column used in the main query are called co-related sub-queries
   C. The sub-queries which are written without parenthesis are called co-related sub-queries
   D. The sub-queries which mandatorily use different tables than those used in the main query are called co-related sub-queries
3. What is true about a co-related sub-query?
   A. It is evaluated only once for the parent query
   B. It is evaluated only thrice for the parent query
   C. It is evaluated once for each row processed by the parent sub-query
   D. All of the above

Name: ____________________ Date: _______________

Score = ___________

Rating: ____________

Short Answer Questions

Note: **Satisfactory rating - 3 points Unsatisfactory - below 3 points**
You can ask you teacher for the copy of the correct answers.

Information Sheet4: writing sub-queries that use aggregates

4.4 Writing sub-queries that use aggregates

Aggregate functions in DBMS take multiple rows from the table and return a value according to the query.

All the aggregate functions are used in Select statement.

Syntax:

```
SELECT <FUNCTION NAME> (<PARAMETER>) FROM <TABLE NAME>
```

AVG Function

This function returns the average value of the numeric column that is supplied as a parameter.

Example: Write a query to select average salary from employee table.

```
Select AVG(salary) from Employee
```

COUNT Function

The count function returns the number of rows in the result. It does not count the null values.

Example: Write a query to return number of rows where salary > 20000.

```
Select COUNT(*) from Employee where Salary > 20000;
```

Types

- COUNT (*): Counts all the number of rows of the table including null.
- COUNT (COLUMN_NAME): count number of non-null values in column.
- COUNT (DISTINCT COLUMN_NAME): count number of distinct values in a column.

MAX Function

The MAX function is used to find maximum value in the column that is supplied as a parameter. It can be used on any type of data.

Example: Write a query to find the maximum salary in employee table.

```
Select MAX (salary) from Employee
```

SUM Function

This function sums up the values in the column supplied as a parameter.

Example: Write a query to get the total salary of employees.

```
Select SUM (salary) from Employee
```

Views

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views:

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables, or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in very similar way as you use them in normal SQL SELECT query.

Example: Consider the CUSTOMERS table having the following records:

Table 1

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6				4500.00
7	Muffy	24	Indore	10000.00

Now, following is the example to create a view from CUSTOMERS table. This view would be used to have customer name and age from CUSTOMERS table:

SQL > CREATE VIEW CUSTOMERS_VIEW AS

SELECT name, age

FROM CUSTOMERS;

Now, you can query CUSTOMERS_VIEW in similar way as you query an actual table. Following is the example:

SQL > SELECT * FROM CUSTOMERS_VIEW;

This would produce the following result:

Table 2

name	age
Ramesh	32
Khilan	25
kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

The WITH CHECK OPTION:

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following is an example of creating same view CUSTOMERS_VIEW with the WITH CHECK OPTION:

CREATE VIEW CUSTOMERS_VIEW AS

SELECT name, age

FROM CUSTOMERS

WHERE age IS NOT NULL

WITH CHECK OPTION;

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View:

A view can be updated under certain conditions:

The SELECT clause may not contain the keyword DISTINCT.

The SELECT clause may not contain summary functions.

The SELECT clause may not contain set functions.

The SELECT clause may not contain set operators.

The SELECT clause may not contain an ORDER BY clause.

The FROM clause may not contain multiple tables.

The WHERE clause may not contain subqueries.

The query may not contain GROUP BY or HAVING.

Calculated columns may not be updated.

All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So if a view satisfies all the above-mentioned rules then you can update a view. Following is an example to update the age of Ramesh:

```
SQL > UPDATE CUSTOMERS_VIEW
    SET AGE = 35
    WHERE name='Ramesh';
```

This would ultimately update the base table CUSTOMERS and same would reflect in the view itself. Now, try to query base table, and SELECT statement would produce the following result:

Table 3

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00

5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Inserting Rows into a View:

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here we cannot insert rows in CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in similar way as you insert them in a table.

Deleting Rows into a View:

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE= 22.

SQL > DELETE FROM CUSTOMERS_VIEWWHERE age = 22;

This would ultimately delete a row from the base table CUSTOMERS and same would reflect in the view itself. Now, try to query base table, and SELECT statement would produce the following result:

Table 4

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Dropping Views:

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple as given below:

DROP VIEW view_name;

Following is an example to drop CUSTOMERS_VIEW from CUSTOMERS table:

DROP VIEW CUSTOMERS_VIEW;

Self-Check 4	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page

Part IV multiple choice

1. Why are views useful? (Choose the most appropriate answer)
   A. Because they have shorter names than tables
   B. To prevent users from accessing the columns of tables
   C. To simplify user SQL
   D. All of the above
2. What is true about creating a view? (Choose the most appropriate answer)
   A. A view can only be created from a table
   B. A view can only be created from one table
   C. A view can be created from one or many tables or views
   D. None of the above
3. Which of the following privileges are required to create views in someone else's schema?
   A. CREATE ANY VIEW
   B. CREATE VIEW
   C. Both A and B
   D. None of the above
4. Which of the following privileges are required to create views in one's own schema?
   A. CREATE TABLE system privilege
   B. CREATE VIEW system privilege
   C. ALTER VIEW system privilege
   D. CREATE ANY VIEW system privilege
5. Which of the following privileges are required to create views in someone else's schema?
   A. CREATE ANY VIEW
   B. CREATE VIEW
   C. Both A and B
   D. None of the above

Score = ____________

Rating: _____________

Name: ___________________________ Date: ________________

Short Answer Questions

Note: **Satisfactory rating - 5 points Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Learning activity performance (LAP)/Job

Name_______________ ID______________

Started-time________________

Finished-time_______________

Instruction

Given information:-sunshine_construction has own project and this project has system object such like Employee, Service and Project. Employee has properties such like EmployeeID,Efirstname,Efathername,Gender,Age and Salary. Customer has properties such like Customer ID,Cfirstname,Cfathername, Gender,Address,kebeleID, Birthdate and rentaldate. Project has properties such like Projno, projname,projlocation and projfund

Employee vendor customer, Employee worksforProject and CustomerworksonProject table.

EmployeeID and Customer ID are referring to the Project table.

Depending on the given information create the Database name sunshine_construction

Given Table1:-Employee

Column name	Data type	Length of value	Default value	constraint
EmployeeID	Char	20	-	Primary key
Efirstname	Varchar	50	Dinadig	Not null
Efathername	Varchar	50	-	Not null
Gender	Char	6	Male	Check “male”or“female”
Age	Int	-		Check age<=45
Salary	Int	-	5000	-

Given Table2:-Customer

Column name	Data type	Length of value	Default value	constraint
CustomerID	Char	25	-	Primary key
Cfirstname	Varchar	50		Not null
Cfathername	Varchar	50	Merertu	Not null
Gender	Char	6	-	-
Address	Char	40	-	-

kebeleID	Char	20	-	-
Birthdate	datetime	-	-	
Rentaldate	datetime	-	-	
EmployeeID	Char	15	-	Foreign key

Given Table3:-Service

Column name	Data type	Length of value	Default value	constraint
Servicenumber	Char	10	-	Primary key
Servicetype	Varchar	50	Adugna	Not null
EmployeeID	Char	20	-	Foreign key
CustomerID	Char	25	-	Foreign key

Task1:-show the Diagram relationship of the three Tables.

Task2:- Add Address varchar (45) column to Employee Table.

Add to **Employee** the given below records

Table: 4

EmployeeID	Efirstname	Efathername	Gender	Age	Salary	Birthdate	Address
EM01	Abdi	Tolera	Male	24	6000	12/11/1983	Ghimbi
EM02	Sena	Ebisa	Female	21	5000	12/12/1985	Aira
EM03	Bontu	Jirata	Female	20	8000	12/10/1986	Nejo
EM04	Geda	Gudina	Male	26	4000	12/11/1980	Ghimbi

Add to Customer the given below records

Table: 5

CustomerID	Cfirstname	Cfathername	Gender	Address	kebeleID	Birthdate	Rentaldate
Cust01	Bekam	Gudina	Male	Ghimbi	G03	12/11/1983	07/09/1980
Cust02	Amenu	Bonsa	Male	Nekemte	N04	12/12/1985	12/12/1990
Cust03	Bikiltu	Jiregna	Female	Bako	B02	12/10/1986	10/10/1980
Cust04	Dame	Darara	Female	Duke	D03	12/11/1980	11/05/1999

Add to Service the given below records

Table: 6

Serviceno	**Servicetype**	**EmployeeID**	**CustomerID**
Serv01	Simple	EM01	Cust01
Serv02	Complex	EM02	Cust02
Serv03	Simple	EM03	Cust03
Serv04	Complex	EM04	Cust04

Task4:- Develop the below queries

- Write query which display EmployeeID, Gender,age and salary from Employee Table.
- Write query which change on the Customer table Cfirstname from ‘Amenu’ to the ‘Gifti’.
- Write query which display from Employee table who EmployeeID equal to EM03 and gender equal to female.
- Write query which delete from Employee table who EfirstnameAbdi
- Write query which display on the Employee table Efirstname equal to Sena and servicetype equal to ‘complex’.
- Write query which display all the information from Employee table and sort Z-A Efirstname

References

Database SQL Developer Supplementary Information for Microsoft SQL Server Migrations.(n.d.). Retrieved 01 11, 2016, from

Database SQL Developer Supplementary Information for Microsoft SQL Server Migrations.(n.d.). Retrieved 01 11, 2016, fromhttps://docs.oracle.com/cd/E10405_01/appdev.120/e10379/ss_oracle_compared.htm

Intro to SQL: Querying and managing data. (n.d.). Retrieved 12 21, 2015, from https://www.khanacademy.org/computing/computer-programming/sql

Selecting from the DUAL Table.(n.d.). Retrieved 7 12, 2016, from http://www.ntchosting.com/encyclopedia/databases/structured-query-language/

SQL Basic Query Structure.(n.d.). Retrieved 11 27, 2015, from

SQL Join.(n.d.). Retrieved 12 11, 2015, from http://www.dofactory.com/sql/join

SQL Tutorial.(n.d.). Retrieved 11 18, 2015, from http://www.w3schools.com/sql/

What is SQL? (n.d.). Retrieved 01 10, 2016, from http://www.sqlcourse.com/intro.html

https://www.w3schools.com/

https://www.w3processing.com/

https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries009.htm

SQL (Structured Query Language).(n.d.). Retrieved 01 10, 2016, from

AKNOWLEDGEMENT

We wish to extend thanks and appreciation to the many representatives of TVET instructors and respective industry experts who donated their time and expertise to the development of this TTLM.

We would like also to express our appreciation to the TVET instructors and respective industry experts of Regional TVET Bureaus, TVET College/ Institutes, BEAR II Project, Bishoftu Management Institute Center, UNESCO and Federal Technical and Vocational Education and Training Agency (FTVET) who made the development of this curriculum with required standards and quality possible.

This TTLM developed on December 2020 at Bishoftu,Ethiopia.

The trainers who developed the TTLM

No	Name	Organization.	Edu. Level	Profession /Job title	E-mail	Mobile
1	AyansaErgiba	Ambo TVETC	Msc	Instructor	aergiba@gmail.com	0917851343
3	EjiguBirhanu	Nekemte TVETC	Msc	Instructor	ejigubiranu2011@gmail.com	0906566892
2	EndaleLema	Adama PTC	Msc	Instructor	endhiywet@gmai.com	0913292212
4	KeresaGadisa	Nekemte TVETC	Msc	Instructor	keresag2010@gmail.com	0920420664
5	MeseretTezera	Atletkenenisa PTC	Msc	Instructor	YafetMes@gmail.com	0911751285

Self-Check Answer Key

Answer key PART_I

Self-check-1	1. **D** **2. A** **3.B** **4. A** **5. A**

Self-check-2	1. **B** 2**. A** 3**. A**

Self-check-3	1. **A** 2. **A**

Self-check-4	1. **C** **2.B**

Self-check-5	1**. A, D**2**.B** 3**.A** 4**. C, D** 5**. A**

Self-check-6	1 **. C** **2.C3.C** 4**. C** **5. A**

Self-check-7	1 **. A, B, C** 2**.A**3**.B** 4**. B** 5**. B**

Self-check-8	1. SELECT*FROM customer WHERE city ='New York'ORNOT grade>100; 2. SELECT*FROM customer WHERE city ='New York'AND grade>100; 3. SELECT*FROM customer WHERE city ='New York'OR grade>100; 4. SELECT*FROM customer WHERE grade >100;

Self-check-9	1. D 2. D 3. D 4. A 5. C

Self-check-10	1. C, D 2. C 3. A,B

Self-check-1	4. A2. C3. A4.B5. .A

Answer key PART-II

Self-check-2	1. A2. B 3. C4.B

Self-check-3	1. D2. D3. B4.B 5.C 6.A

Self-check-4	1. D2. D3. B4.B 5.C 6.A

Self-check-1	1. A 2. C 3.C 4.B,D

Self-check-2	1. A 2. B 3.C 4.B,D

Self-check-3	1. A 2. D 3.B 4.C

Self-check-4	1. A 2. D 3.B 4.C

Self-check-5	2. SELECTSUM(purch_amt)FROM orders; 3. SELECTAVG(purch_amt)FROM orders; 4. SELECTCOUNT(DISTINCTsalesman_id)FROM orders;

Answer key PART-III

Self-check-1	1. C 2. D

Self-check-3	2. A,B 2. A 3. C 4. B

Self-check-2	1. A2. B

Self-check-1	1. A 2. A 3. D 4. C

Answer key PART-IV

Self-check-2	3. D2. B3. D

Self-check-3	4. A2. B3. C

Self-check-4	5. B2. C3. A4.B 5. .A