DataBase Management System and SQL

MYSQL

It is freely available open source Relational Database Management System (RDBMS) that uses **Structured Query Language(SQL).** In MySQL database , information is stored in Tables. A single MySQL database can contain many tables at once and store thousands of individual records.

SQL (Structured Query Language)

SQL is a language that enables you to create and operate on relational databases, which are sets of related information stored in tables.

DIFFERENT DATA MODELS

A **data model refers to a set of concepts to describe the structure of a database**, and certain constraints (restrictions) that the database should obey. The four data model that are used for database management are :

1. **Relational data model** : In this data model, the data is organized into tables (i.e. rows and columns). These tables are called relations.

2. Hierarchical data model 3. Network data model 4. Object Oriented data model

RELATIONAL MODEL TERMINOLOGY

1. **Relation :** A table storing logically related data is called a Relation.
2. **Tuple :** A **row of a relation** is generally referred to as a tuple.
3. **Attribute :** A **column** of a relation is generally referred to as an attribute.
4. **Degree :** This refers to the **number of attributes** in a relation.
5. **Cardinality :** This refers to the **number of tuples** in a relation.
6. **Primary Key :** This refers to a set of one or more attributes that can uniquely identify tuples within the relation.
7. **Candidate Key :** All attribute combinations inside a relation that can serve as primary key are candidate keys as these are candidates for primary key position.
8. **Alternate Key :** A candidate key that is not primary key, is called an alternate key.
9. **Foreign Key :** A non-key attribute, whose values are derived from the primary key of some other table, is known as foreign key in its current table.

REFERENTIAL INTEGRITY

- A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data. This integrity is ensured by foreign key.

CLASSIFICATION OF SQL STATEMENTS

SQL commands can be mainly divided into following categories:

1. Data Definition Language(DDL) Commands

Commands that allow you to perform task, related to data definition e.g;

- Creating, altering and dropping.
- Granting and revoking privileges and roles.
- Maintenance commands.

2. **Data Manipulation Language(DML) Commands**

   Commands that allow you to perform data manipulation e.g., retrieval, insertion, deletion and modification of data stored in a database.

3. **Transaction Control Language(TCL) Commands**

   Commands that allow you to manage and control the transactions e.g.,

   - Making changes to database, permanent
   - Undoing changes to database, permanent
   - Creating savepoints
   - Setting properties for current transactions.

MySQL ELEMENTS

1. Literals 2. Datatypes 3. Nulls 4. Comments

LITERALS

It refer to a fixed data value. This fixed data value may be of character type or numeric type. For example, ‘replay’ , ‘Raj’, ‘8’ , ‘306’ are all character literals.

Numbers not enclosed in quotation marks are numeric literals. E.g. 22 , 18 , 1997 are all numeric literals.

Numeric literals can either be integer literals i.e., without any decimal or be real literals i.e. with a decimal point e.g. 17 is an integer literal but 17.0 and 17.5 are real literals.

DATA TYPES

Data types are means to identify the type of data and associated operations for handling it. MySQL data types are divided into three categories:

- ➢ Numeric
- ➢ Date and time
- ➢ String types

Numeric Data Type

1. int – used for number without decimal.
2. Decimal(m,d) – used for floating/real numbers. m denotes the total length of number and d is number of decimal digits.

Date and Time Data Type

1. date – used to store date in YYYY-MM-DD format.
2. time – used to store time in HH:MM:SS format.

String Data Types

1. char(m) – used to store a fixed length string. **m** denotes max. number of characters.
2. varchar(m) – used to store a variable length string. **m** denotes max. no. of characters.

DIFFERENCE BETWEEN CHAR AND VARCHAR DATA TYPE

S.NO.	Char Datatype	Varchar Datatype
1.	It specifies a **fixed length** character String.	It specifies a **variable length** character string.
2.	When a column is given datatype as CHAR(**n**), then MySQL ensures that all values stored in that column have this length i.e. **n** bytes. If a value is shorter than this length **n** then blanks are added, but the size of value remains **n** bytes.	When a column is given datatype as VARCHAR(**n**), then the maximum size a value in this column can have is **n** bytes. Each value that is stored in this column store exactly as you specify it i.e. no blanks are added if the length is shorter than maximum length **n**.

NULL VALUE

If a column in a row has no value, then column is said to be **null** , or to contain a null. **You should use a null value** when the actual value is not known or when a value would not be meaningful.

DATABASE COMMNADS

1. VIEW EXISTING DATABASE

To view existing database names, the command is : **SHOW DATABASES ;**

2. CREATING DATABASE IN MYSQL

For creating the database in MySQL, we write the following command : **CREATE DATABASE** <databasename> ;

e.g. In order to create a database Student, command is :

CREATE DATABASE Student ;

3. ACCESSING DATABASE

For accessing already existing database , we write :

USE <databasename> ;

e.g. to access a database named Student , we write command as :

USE Student ;

4. DELETING DATABASE

For deleting any existing database , the command is :

DROP DATABASE <databasename> ;

e.g. to delete a database , say student, we write command as ; **DROP DATABASE** Student ;

5. VIEWING TABLE IN DATABASE

In order to view tables present in currently accessed database , command is : **SHOW TABLES ;**

CREATING TABLES IN MYSQL

- Tables are created with the CREATE TABLE command. When a table is created, its columns are named, data types and sizes are supplied for each column.

Syntax of CREATE TABLE command is : CREATE TABLE <table-name>

```
( <column name> <data type> ,
  <column name> <data type> ,
 ……… ) ;
```

E.g. in order to create table EMPLOYEE given below :

ECODE	ENAME	GENDER	GRADE	GROSS

We write the following command :

```
CREATE TABLE employee
( ECODE integer ,
     ENAME varchar(20) ,
     GENDER char(1) ,
     GRADE char(2) ,
     GROSS integer     ) ;
```

INSERTING DATA INTO TABLE

- The rows are added to relations(table) using INSERT command of SQL. Syntax of INSERT is : INSERT INTO <tablename> [<column list>]
VALUE (<value1> , <value2> , …..) ;

e.g. to enter a row into EMPLOYEE table (created above), we write command as :

```
INSERT INTO employee
VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);
```

OR

```
INSERT INTO employee (ECODE , ENAME , GENDER , GRADE , GROSS)
VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);
```

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000

In order to insert another row in EMPLOYEE table , we write again INSERT command :

```
INSERT INTO employee
VALUES(1002 , 'Akash' , 'M' , 'A1' , 35000);
```

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000

INSERTING NULL VALUES

- To insert value NULL in a specific column, we can type NULL without quotes and NULL will be inserted in that column. E.g. in order to insert NULL value in ENAME column of above table, we write INSERT command as :

```
INSERT INTO EMPLOYEE
VALUES (1004 , NULL , 'M' , 'B2' , 38965 ) ;
```

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	NULL	M	B2	38965

SIMPLE QUERY USING SELECT COMMAND

- The SELECT command is used to pull information from a table. Syntax of SELECT command is : SELECT <column name>,<column name>
  FROM <tablename>
  WHERE <condition name> ;

SELECTING ALL DATA

- In order to retrieve everything (all columns) from a table, SELECT command is used as : **SELECT * FROM** <tablename> ;

e.g.

In order to retrieve everything from **Employee** table, we write SELECT command as :

EMPLOYEE

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	NULL	M	B2	38965

SELECT * FROM Employee ;

SELECTING PARTICULAR COLUMNS

EMPLOYEE

ECODE	**ENAME**	**GENDER**	**GRADE**	**GROSS**
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000
1009	Neema	F	A2	52000

- A particular column from a table can be selected by specifying column-names with SELECT command. E.g. in above table, if we want to select ECODE and ENAME column, then command is :

```
SELECT ECODE , ENAME
 FROM EMPLOYEE ;
```

E.g.2 in order to select only ENAME, GRADE and GROSS column, the command is :

```
SELECT ENAME , GRADE ,
 GROSS FROM EMPLOYEE ;
```

SELECTING PARTICULAR ROWS

We can select particular rows from a table by specifying a condition through **WHERE clause** along with SELECT statement. **E.g.** In employee table if we want to select rows where Gender is female, then command is :

```
SELECT * FROM EMPLOYEE
WHERE GENDER = ‘F’ ;
```

E.g.2. in order to select rows where salary is greater than 48000, then command is :

```
SELECT * FROM EMPLOYEE
WHERE GROSS > 48000 ;
```

ELIMINATING REDUNDANT DATA

The **DISTINCT** keyword eliminates duplicate rows from the results of a SELECT statement. For example ,

```
SELECT GENDER FROM EMPLOYEE ;
```

GENDER
M
M
F
M
F
F

```
SELECT DISTINCT(GENDER) FROM EMPLOYEE ;
```

DISTINCT(GENDER)
M
F

VIEWING STRUCTURE OF A TABLE

- If we want to know the structure of a table, we can use DESCRIBE or DESC command, as per following syntax :

```
DESCRIBE | DESC <tablename> ;
```

e.g. to view the structure of table **EMPLOYEE**, command is : **DESCRIBE** EMPLOYEE ; **OR DESC** EMPLOYEE ;

USING COLUMN ALIASES

- The columns that we select in a query can be given a different name, i.e. column alias name for output purpose.

Syntax :

```
SELECT <columnname> AS column alias , <columnname> AS column alias …..
FROM <tablename> ;
```

e.g. In output, suppose we want to display ECODE column as EMPLOYEE_CODE in output , then command is :

```
SELECT ECODE AS "EMPLOYEE_CODE"
FROM EMPLOYEE ;
```

CONDITION BASED ON A RANGE

- The **BETWEEN** operator defines a range of values that the column values must fall in to make the condition true. The range include both lower value and upper value.

  e.g. to display ECODE, ENAME and GRADE of those employees whose salary is between 40000 and 50000, command is:

```
SELECT ECODE , ENAME ,GRADE
FROM EMPLOYEE
WHERE GROSS BETWEEN 40000 AND 50000 ;
```

Output will be :

ECODE	ENAME	GRADE
1001	Ravi	E4
1006	Ruby	A1

CONDITION BASED ON A LIST

- To specify a list of values, IN operator is used. The IN operator selects value that match any value in a given list of values. E.g.

```
SELECT * FROM EMPLOYEE
WHERE GRADE IN ('A1' , 'A2');
```

Output will be :

ECODE	**ENAME**	**GENDER**	**GRADE**	**GROSS**
1002	Akash	M	A1	35000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000
1009	Neema	F	A2	52000

- The **NOT IN** operator finds rows that do not match in the list. E.g.

```
SELECT * FROM EMPLOYEE
WHERE GRADE NOT IN ('A1' , 'A2');
```

Output will be :

ECODE	**ENAME**	**GENDER**	**GRADE**	**GROSS**
1001	Ravi	M	E4	50000
1004	Neela	F	B2	38965

CONDITION BASED ON PATTERN MATCHES

- LIKE operator is used for pattern matching in SQL. Patterns are described using two special wildcard characters:

  1. percent(%) – The % character matches any substring.
  2. underscore(_) – The _ character matches any character.

  e.g. to display names of employee whose name starts with R in EMPLOYEE table, the command is :

```
SELECT ENAME
FROM EMPLOYEE
WHERE ENAME LIKE 'R%' ;
```

Output will be :

ENAME
Ravi
Ruby

e.g. to display details of employee whose second character in name is 'e'.

```
SELECT *
FROM EMPLOYEE
WHERE ENAME LIKE '_e%' ;
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000

e.g. to display details of employee whose name ends with 'y'.

```
SELECT *
FROM EMPLOYEE
WHERE ENAME LIKE '%y' ;
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000

SEARCHING FOR NULL

- The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. E.g. to list employee details whose salary contain NULL, we use the command :

```
SELECT *
FROM EMPLOYEE
WHERE GROSS IS NULL ;
```

e.g.

STUDENT

Roll_No	Name	Marks
1	ARUN	NULL
2	RAVI	56
4	SANJAY	NULL

to display the names of those students whose marks is NULL, we use the command :

```
SELECT Name
FROM EMPLOYEE
WHERE Marks IS NULL ;
```

Output will be :

Name
ARUN
SANJAY

SORTING RESULTS

Whenever the SELECT query is executed , the resulting rows appear in a predecided order.The **ORDER BY clause** allow sorting of query result. The sorting can be done either in ascending or descending order, the default is ascending.

The **ORDER BY clause is used as :**

```
SELECT <column name> , <column name>….
FROM <tablename>
WHERE <condition>
ORDER BY <column name> ;
```

e.g. to display the details of employees in EMPLOYEE table in alphabetical order, we use command :

```
SELECT *
FROM EMPLOYEE
ORDER BY ENAME ;
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000
1001	Ravi	M	E4	50000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000

e.g. display list of employee in descending alphabetical order whose salary is greater than 40000.

```
SELECT ENAME
FROM EMPLOYEE
WHERE GROSS > 40000
ORDER BY ENAME desc ;
```

Output will be :

ENAME
Ravi
Ruby
Neema

MODIFYING DATA IN TABLES

you can modify data in tables using UPDATE command of SQL. The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword. Syntax of update command is :

```
UPDATE <tablename>
SET <columnname>=value , <columnname>=value
WHERE <condition> ;
```

e.g. to change the salary of employee of those in EMPLOYEE table having employee code 1009 to 55000.

```
UPDATE  EMPLOYEE
SET GROSS = 55000
WHERE ECODE = 1009 ;
```

UPDATING MORE THAN ONE COLUMNS

e.g. to update the salary to 58000 and grade to B2 for those employee whose employee code is 1001.

```
UPDATE EMPLOYEE
SET GROSS = 58000, GRADE='B2'
WHERE ECODE = 1009 ;
```

OTHER EXAMPLES

e.g.1. Increase the salary of each employee by 1000 in the EMPLOYEE table.

```
UPDATE EMPLOYEE
SET GROSS = GROSS +100 ;
```

e.g.2. Double the salary of employees having grade as 'A1' or 'A2' .

```
UPDATE EMPLOYEE
SET GROSS = GROSS * 2 ;
WHERE GRADE='A1' OR GRADE='A2' ;
```

e.g.3. Change the grade to 'A2' for those employees whose employee code is 1004 and name is Neela.

```
UPDATE EMPLOYEE
SET GRADE='A2'
WHERE ECODE=1004 AND GRADE='NEELA' ;
```

DELETING DATA FROM TABLES

To delete some data from tables, DELETE command is used. **The DELETE command removes rows from a table.** The syntax of DELETE command is :

```
DELETE FROM <tablename>
WHERE <condition> ;
```

For example, to remove the details of those employee from EMPLOYEE table whose grade is A1.

```
DELETE FROM EMPLOYEE
WHERE GRADE ='A1' ;
```

TO DELETE ALL THE CONTENTS FROM A TABLE

```
DELETE FROM EMPLOYEE ;
```

So if we do not specify any condition with WHERE clause, then all the rows of the table will be deleted. Thus above line will delete all rows from employee table.

DROPPING TABLES

The DROP TABLE command lets you drop a table from the database. The **syntax of DROP TABLE** command is :

```
DROP TABLE <tablename> ;
```

e.g. to drop a table employee, we need to write :

```
DROP TABLE employee ;
```

Once this command is given, the table name is no longer recognized and no more commands can be given on that table.
After this command is executed, all the data in the table along with table structure will be deleted.

S.NO.	DELETE COMMAND	DROP TABLE COMMAND
1	It is a DML command.	It is a DDL Command.
2	This command is used to delete only rows of data from a table	This command is used to delete all the data of the table along with the structure of the table. The table is no longer recognized when this command gets executed.
3	Syntax of DELETE command is: **DELETE FROM** <tablename> **WHERE** <condition> ;	Syntax of DROP command is : **DROP TABLE** <tablename> ;

ALTER TABLE COMMAND

The ALTER TABLE command is used to change definitions of existing tables.(adding columns,deleting columns etc.). The ALTER TABLE command is used for :

1. adding columns to a table

2. Modifying column-definitions of a table.
3. Deleting columns of a table.
4. Adding constraints to table.
5. Enabling/Disabling constraints.

ADDING COLUMNS TO TABLE

To add a column to a table, ALTER TABLE command can be used as per following syntax:

```
ALTER TABLE <tablename>
ADD <Column name> <datatype> <constraint> ;
```

e.g. to add a new column ADDRESS to the EMPLOYEE table, we can write command as :

```
ALTER TABLE EMPLOYEE
ADD ADDRESS VARCHAR(50);
```

A new column by the name ADDRESS will be added to the table, where each row will contain NULL value for the new column.

ECODE	ENAME	GENDER	GRADE	GROSS	ADDRESS
1001	Ravi	M	E4	50000	NULL
1002	Akash	M	A1	35000	NULL
1004	Neela	F	B2	38965	NULL
1005	Sunny	M	A2	30000	NULL
1006	Ruby	F	A1	45000	NULL
1009	Neema	F	A2	52000	NULL

However **if you specify NOT NULL constraint while adding a new column**, MySQL adds the new column with the default value of that datatype e.g. for INT type it will add 0 , for CHAR types, it will add a space, and so on.

e.g. Given a table namely Testt with the following data in it.

Col1	Col2
1	A
2	G

Now following commands are given for the table. Predict the table contents after each of the following statements:

(i) ALTER TABLE testt ADD col3 INT ;
(ii) ALTER TABLE testt ADD col4 INT NOT NULL ;
(iii) ALTER TABLE testt ADD col5 CHAR(3) NOT NULL ;
(iv) ALTER TABLE testt ADD col6 VARCHAR(3);

MODIFYING COLUMNS

Column name and data type of column can be changed as per following syntax :

```
ALTER TABLE <table name>
CHANGE <old column name> <new column name> <new datatype>;
```

If **Only data type of column need to be changed**, then

```
ALTER TABLE <table name>
MODIFY <column name> <new datatype>;
```

e.g.1. In table EMPLOYEE, change the column GROSS to SALARY.

ALTER TABLE EMPLOYEE
CHANGE GROSS SALARY INTEGER;

e.g.2. In table EMPLOYEE , change the column ENAME to EM_NAME and data type from VARCHAR(20) to VARCHAR(30).

ALTER TABLE EMPLOYEE
CHANGE ENAME EM_NAME VARCHAR(30);

e.g.3. In table EMPLOYEE , change the datatype of GRADE column from CHAR(2) to VARCHAR(2).

ALTER TABLE EMPLOYEE
MODIFY GRADE VARCHAR(2);

DELETING COLUMNS

To delete a column from a table, the ALTER TABLE command takes the following form :

ALTER TABLE <table name>
DROP <column name>;

e.g. to delete column GRADE from table EMPLOYEE, we will write :

ALTER TABLE EMPLOYEE
DROP GRADE ;

ADDING/REMOVING CONSTRAINTS TO A TABLE

ALTER TABLE statement can be used to add constraints to your existing table by using it in following manner:

❖ **TO ADD PRIMARY KEY CONSTRAINT**

ALTER TABLE <table name>
ADD PRIMARY KEY (Column name);

e.g. to add PRIMARY KEY constraint on column ECODE of table EMPLOYEE , the command is :

ALTER TABLE EMPLOYEE
ADD PRIMARY KEY (ECODE) ;

❖ **TO ADD FOREIGN KEY CONSTRAINT**

ALTER TABLE <table name>
ADD FOREIGN KEY (Column name) REFERENCES Parent Table (Primary key of Parent Table);

REMOVING CONSTRAINTS

- To remove primary key constraint from a table, we use ALTER TABLE command
  as : **ALTER TABLE** <table name>
  DROP PRIMARY KEY ;
- To remove foreign key constraint from a table, we use ALTER TABLE command
  as : **ALTER TABLE** <table name>
  DROP FOREIGN KEY ;

ENABLING/DISABLING CONSTRAINTS

Only foreign key can be disabled/enabled in MySQL.

To disable foreign keys : **SET** FOREIGN_KEY_CHECKS = 0 ;

To enable foreign keys : **SET** FOREIGN_KEY_CHECKS = 1 ;

INTEGRITY CONSTRAINTS/CONSTRAINTS

- A constraint is a condition or check applicable on a field(column) or set of fields(columns).
- Common types of constraints include :

S.No.	Constraints	Description
1	NOT NULL	Ensures that a column cannot have NULL value
2	DEFAULT	Provides a default value for a column when none is specified
3	UNIQUE	Ensures that all values in a column are different
4	CHECK	Makes sure that all values in a column satisfy certain criteria
5	PRIMARY KEY	Used to uniquely identify a row in the table
6	FOREIGN KEY	Used to ensure referential integrity of the data

NOT NULL CONSTRAINT

By default, a column can hold NULL. It you not want to allow NULL value in a column, then NOT NULL constraint must be applied on that column. E.g.

```
CREATE TABLE Customer
 (      SID integer NOT NULL ,
          Last_Name varchar(30)   NOT NULL ,
          First_Name varchar(30)      ) ;
```

Columns **SID** and **Last_Name** cannot include NULL, while **First_Name** can include NULL.

An attempt to execute the following SQL statement,

```
INSERT INTO Customer
VALUES (NULL , ‘Kumar’ , ‘Ajay’);
```

will result in an error because this will lead to column SID being NULL, which violates the NOT NULL constraint on that column.

DEFAULT CONSTARINT

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value. **E.g.**

```
CREATE TABLE Student
( Student_ID integer ,
   Name varchar(30) ,
   Score integer DEFAULT 80);
```

When following SQL statement is executed on table created above:

```
INSERT INTO Student

VALUES (10 , ‘Ravi’ );
```

no value has been provided for score field.

Then table **Student** looks like the following:

Student_ID	Name	Score
10	Ravi	**80**

score field has got the default value

UNIQUE CONSTRAINT

- The UNIQUE constraint ensures that all values in a column are distinct. In other words, no two rows can hold the same value for a column with UNIQUE constraint.

e.g.

```
CREATE TABLE Customer
(       SID integer Unique ,
         Last_Name varchar(30) ,
         First_Name varchar(30) ) ;
```

Column SID has a unique constraint, and hence cannot include duplicate values. So, if the table already contains the following rows :

SID	Last_Name	First_Name
1	Kumar	Ravi
2	Sharma	Ajay
3	Devi	Raj

The executing the following SQL statement,

```
INSERT INTO Customer
VALUES ('3' , 'Cyrus' , 'Grace') ;
```

will result in an error because the value 3 already exist in the SID column, thus trying to insert another row with that value violates the UNIQUE constraint.

CHECK CONSTRAINT

- The CHECK constraint ensures that all values in a column satisfy certain conditions. Once defined, the table will only insert a new row or update an existing row if the new value satisfies the CHECK constraint.

  e.g.

```
CREATE TABLE Customer
(       SID integer CHECK (SID > 0),
         Last_Name varchar(30) ,
         First_Name varchar(30) ) ;
```

So, attempting to execute the following statement :

```
INSERT INTO Customer
VALUES (-2 , 'Kapoor' , 'Raj');
```

will result in an error because the values for SID must be greater than 0.

PRIMARY KEY CONSTRAINT

- A primary key is used to identify each row in a table. A primary key can consist of one or more fields(column) on a table. When multiple fields are used as a primary key, they are called a **composite key.**

- You can define a primary key in CREATE TABLE command through keywords PRIMARY KEY. e.g.

```
CREATE TABLE Customer
(       SID integer NOT NULL PRIMARY KEY,
         Last_Name varchar(30) ,
         First_Name varchar(30) ) ;
```

Or

```
CREATE TABLE Customer
 ( SID integer,
        Last_Name varchar(30) ,
        First_Name  varchar(30),
        PRIMARY KEY (SID) ) ;
```

- The latter way is useful if you want to specify a composite primary key, **e.g.**

```
CREATE TABLE Customer
 (       Branch integer NOT NULL,
         SID integer NOT NULL ,
         Last_Name varchar(30) ,
         First_Name varchar(30),
         PRIMARY KEY (Branch , SID) ) ;
```

FOREIGN KEY CONSTRAINT

- Foreign key is a non key column of a table (**child table**) that draws its values from **primary key** of another table(**parent table).**
- The table in which a foreign key is defined is called a **referencing table or child table.** A table to which a foreign key points is called **referenced table or parent table.**

e.g.

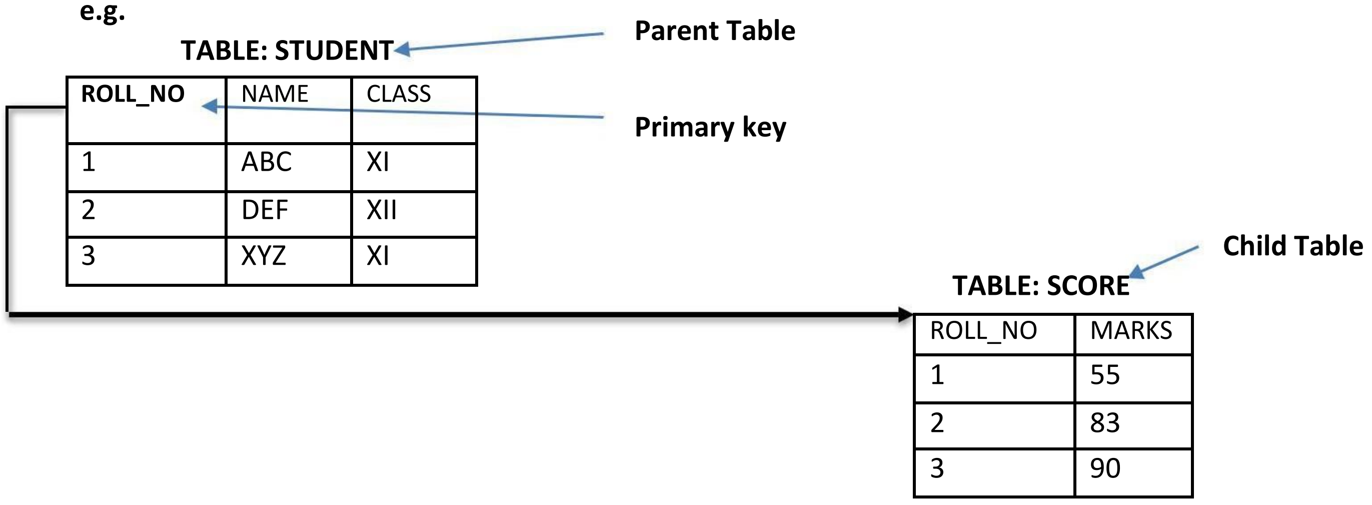


TABLE: STUDENT

ROLL_NO	NAME	CLASS
1	ABC	XI
2	DEF	XII
3	XYZ	XI

TABLE: SCORE

ROLL_NO	MARKS
1	55
2	83
3	90

Here column Roll_No is a foreign key in table SCORE(Child Table) and it is drawing its values from Primary key (ROLL_NO) of STUDENT table.(Parent Key).

```
CREATE TABLE STUDENT
(    ROLL_NO integer NOT NULL PRIMARY KEY ,
     NAME VARCHAR(30) ,
     CLASS VARCHAR(3) );
```

```
CREATE TABLE SCORE
 ( ROLL_NO integer ,
   MARKS integer ,
   FOREIGN KEY(ROLL_NO) REFERNCES STUDENT(ROLL_NO) ) ;
```

**** Foreign key is always defined in the child table.***

Syntax for using foreign key

```
FOREIGN KEY(column name) REFERENCES Parent_Table(PK of Parent Table);
```

REFERENCING ACTIONS

Referencing action with ON DELETE clause determines what to do in case of a DELETE occurs in the parent table.
Referencing action with ON UPDATE clause determines what to do in case of a UPDATE occurs in the parent table.

Actions:

1. **CASCADE :** This action states that if a DELETE or UPDATE operation affects a row from the parent table, then automatically delete or update the matching rows in the child table i.e., cascade the action to child table.
2. **SET NULL :** This action states that if a DELETE or UPDATE operation affects a row from the parent table, then set the foreign key column in the child table to NULL.
3. **NO ACTION :** Any attempt for DELETE or UPDATE in parent table is not allowed.
4. **RESTRICT :** This action rejects the DELETE or UPDATE operation for the parent table.

Q: Create two tables

Customer(customer_id, name)
Customer_sales(transaction_id, amount , **customer_id**)

Underlined columns indicate primary keys and bold column names indicate foreign key.
Make sure that no action should take place in case of a DELETE or UPDATE in the parent table.

Sol :
```
CREATE TABLE Customer (
        customer_id int Not Null Primary Key ,
        name varchar(30) ) ;

    CREATE TABLE Customer_sales (
        transaction_id Not Null Primary Key ,
        amount int ,
        customer_id int ,
        FOREIGN KEY(customer_id) REFERENCES Customer (customer_id)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION );
```

Q: Distinguish between a Primary Key and a Unique key in a table.

S.NO.	PRIMARY KEY	UNIQUE KEY
1.	Column having Primary key can't contain NULL value	Column having Unique Key can contain NULL value
2.	There can be only one primary key in Table.	Many columns can be defined as Unique key

Q: Distinguish between ALTER Command and UPDATE command of SQL.

S.NO.	ALTER COMMAND	UPDATE COMMAND
1.	It is a DDL Command	It is a DML command
2.	It is used to change the definition of existing table, i.e. adding column, deleting column, etc.	It is used to modify the data values present in the rows of the table.

3.	Syntax for adding column in a table:
	ALTER TABLE <tablename>
	ADD <Column name><Datatype> ;
		Syntax for using UPDATE command:
		UPDATE <Tablename>
		SET <Columnname>=value
		WHERE <Condition> ;

AGGREGATE / GROUP FUNCTIONS

Aggregate / Group functions work upon groups of rows , rather than on single row, and return one single output. Different aggregate functions are : COUNT( ) , AVG( ) , MIN( ) , MAX( ) , SUM ( )

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20
8912	SUR	NULL	3000	10

1. **AVG( )**

This function computes the average of given data. e.g. SELECT AVG(SAL)
FROM EMPL ;

Output

AVG(SAL)
6051.6

2. **COUNT( )**

This function counts the number of rows in a given column.

If you specify the COLUMN name in parenthesis of function, then this function returns rows where COLUMN is not null.
If you specify the asterisk (*), this function returns all rows, including duplicates and nulls.

e.g. SELECT COUNT(*)
FROM EMPL ;

Output

COUNT(*)
5

e.g.2 SELECT COUNT(JOB)
FROM EMPL ;

Output

COUNT(JOB)
4

3. **MAX( )**

This function returns the maximum value from a given column or expression.

e.g. SELECT MAX(SAL)
FROM EMPL ;

Output

MAX(SAL)
9870

4. MIN( )

This function returns the minimum value from a given column or expression.

```
e.g. SELECT MIN(SAL)
        FROM EMPL ;
```

Output

MIN(SAL)
2985

5. SUM( )

This function returns the sum of values in given column or expression.

```
e.g. SELECT SUM(SAL)
       FROM EMPL ;
```

Output

SUM(SAL)
30258

GROUPING RESULT – GROUP BY

The GROUP BY clause combines all those records(row) that have identical values in a particular field(column) or a group of fields(columns).

GROUPING can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20

e.g. Calculate the number of employees in each grade.

```
SELECT JOB, COUNT(*)
FROM EMPL
GROUP BY JOB ;
```

Output

JOB	COUNT(*)
CLERK	1
SALESMAN	2
MANAGER	1

e.g.2. Calculate the sum of salary for each department.

```
SELECT DEPTNO , SUM(SAL)
FROM EMPL
GROUP BY DEPTNO ;
```

Output

DEPTNO	SUM(SAL)
10	2985
20	15513
30	8760

e.g.3. find the average salary of each department.
Sol:

*** One thing that you should keep in mind is that while grouping , you should include only those values in the SELECT list that either have the same value for a group or contain a group(aggregate) function. Like in e.g. 2 given above, DEPTNO column has one(same) value for a group and the other expression SUM(SAL) contains a group function.*

NESTED GROUP

- To create a group within a group i.e., nested group, you need to specify multiple fields in the GROUP BY expression. e.g. To group records **job wise** within **Deptno wise**, you need to issue a query statement like :

```
SELECT DEPTNO , JOB , COUNT(EMPNO)
FROM EMPL
GROUP BY DEPTNO , JOB ;
```

Output

DEPTNO	JOB	COUNT(EMPNO)
10	CLERK	1
20	SALESMAN	1
20	MANAGER	1
30	SALESMAN	1

PLACING CONDITION ON GROUPS – HAVING CLAUSE

- The **HAVING clause places conditions on groups** in contrast to WHERE clause that places condition on individual rows. While **WHERE conditions cannot include aggregate functions, HAVING conditions can do so.**
- e.g. To display the jobs where the number of employees is less than 2,

```
SELECT JOB, COUNT(*)
FROM EMPL
GROUP BY JOB
HAVING COUNT(*) < 2 ;
```

Output

JOB	COUNT(*)
CLERK	1
MANAGER	1

STRUCTURED QUERY LANGUAGE

TYPE A : VERY SHORT ANSWER QUESTIONS

1. **Ans.**	**What is SQL? What are the various subdivisions of SQL?** SQL means Structured Query Language. It is the set of commands that is recognized by all RDBMS. Data Definition Language (DDL) Data Manipulation Language (DML) Data Control Language (DCL)
2. **Ans.**	**Give examples of some DDL commands and some DML commands.** DDL Commands f CREATE g ALTER h DROP DML Commands - INSERT INTO - DELETE - UPDATE
3. **Ans.**	**What is the difference between column constraint and table constraint? Name some database integrity constrains.** The difference between column constraint and table constraint is that column constraint applies only to individual columns, whereas table constraints apply to groups of one or more columns. Following are the few of database integrity constrains: 1. Unique constraint 2. Primary Key constraint 3. Default constraint 4. Check constraint
4. **Ans.**	**1. How do following constraint work?** **(i) Unique (ii) Primary Key (iii) Default (iv) Check** Unique: This constraint ensures that no two rows have the same value in the specified columns. For eg , CREATE TABLE employee (ecode integer NOT NULL UNIQUE, ename char(20),Sex char(2)); Primary Key: Primary key does not allow NULL value and Duplicate data in the column which is declared as Primary Key. For eg , CREATE TABLE employee (ecode integer NOT NULL PRIMARY KEY, ename char(20),Sexchar(2)); Default: When a user does not enter a value for the column, automatically the defined default value is inserted in field. A column can have only one default value. For eg , CREATE TABLE employee (ecode integer NOT NULL PRIMARY KEY, ename char(20), Sexchar(2), Grade char(2) DEFAULT = 'E1'); Check: This constraint limits values that can inserted into a column of table. For eg , CREATE TABLE employee (ecode integer NOT NULL PRIMARY KEY, ename char(20),Sex char(2) , Grade char(2) DEFAULT = 'E1', Gross decimal CHECK (gross > 2000);
5. **Ans.**	**Compare DISTINCT and ALL keywords when used with SELECT command.** DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement. ALL keyword retains the duplicate rows, by default ALL keyword is use by SELECT statement.

6.	**What is wrong with the following statement? Write the corrected form of this query :**
	SELECT * FROM employee
	WHERE grade = NULL ;

Ans. IS NULL should be used in place of = NULL. Following is the correct statement :

```
SELECT * FROM employee
WHERE grade IS NULL ;
```

7. **What is the difference between where and having clause ?**

Ans.

WHERE CLAUSE	**HAVING CLAUSE**
Places conditions on individual rows.	Places conditions on groups.
Cannot include aggregate function.	Can include aggregate function.
For eg. SELECT * FROM student WHERE Rno >=10;	**For eg.** SELECT AVG(marks) FROM student GROUP BY grade HAVING grade = 'B1';

8. **What is difference between working of the following functions?**
Count(*),Count (<column-name>), Count (DISTINCT), Count (ALL)

Ans. Count(*):- The COUNT(*) function returns the number of records in a table:

```
SELECT COUNT(*) FROM student;
```

Count (<column-name>):- The COUNT(<column-name>) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(name) FROM student;
```

Count (DISTINCT):- The COUNT(DISTINCT column_name) function returns the number of distinct values of the specified column:

```
SELECT COUNT(DISTINCT city) FROM student;
```

Count (ALL):- to count the number of non-null values in column dept, i.e. counting repetitions too.

```
SELECT COUNT(ALL) FROM student;
```

11. **What is the condition of dropping a table?**

Ans. There is a one condition for dropping a table that is a table must be an empty table which we want to drop.

Note: For the following questions consider the tables EMP given in book.

12. **Insert a record with suitable data in the table EMP, tabing system date as the Hiredate.**

Ans.

```
INSERT INTO emp VALUES
(1101,'ROBIN','CLERK',7902,curdate(),5000.00,500.00,10);
```

13. **To create a table DEPTO30 to hold the employee numbers, names, jobs and salaries of employee in department with DeptNo = 30.**

Ans.

```
CREATE TABLE DEPTP30 AS(SELECT EmpNo, EmpName, Job, Sal FROM EMP WHERE
DeptNo=30);
```

14. **Display names all employees whose names include either of the substring “TH” or “LL”.**

Ans.

```
SELECT empname FROM emp WHERE(empname LIKE '%TH%' OR empname LIKE '%LL%');
```

15. **Display data for all CLERKS who earn between 1000 and 2000.**

Ans.

```
SELECT * FROM emp WHERE((job LIKE 'clerk') AND (sal BETWEEN 1000 AND 2000));
```

16. **Display data for all employees sorted by their department, seniority and salary.**

Ans.

```
SELECT * FROM emp ORDER BY deptno, hiredate, sal;
```

17. **Write a SQL statement to list EmpNo, EmpName, DeptNo, for all the employees. This information is should be sorted on EmpName.**

Ans.

```
SELECT empno, empname,deptno FROM emp ORDER BY empname;
```

18. **Write SQL statement for : Find all the employees who have no manager.**

Ans.

```
SELECT empname FROM emp WHERE mgr IS NULL;
```

19. **Write a SQL statement (s) to list all employees in the following format:**

EMPLOYEE	WORKS IN DEPARTMENT	DeptNo
7369-SMITH	WORKS IN DEPARTMENT	20
7300-SUDHIR	WORKS IN DEPARTMENT	20
7345-RAJ	WORKS IN DEPARTMENT	10
7329-SMITHS	WORKS IN DEPARTMENT	30
7234-SANTOSH	WORKS IN DEPARTMENT	30

Ans.

```
SELECT ename,'WORKS IN DEPARTMENT', deptno FROM emp;
```

20. **To find all those employees whose job does not start with ‘M’.**

Ans.

```
SELECT empname FROM emp WHERE job NOT LIKE 'M%';
```

21. **To display all employees who were hired during 1995.**

Ans.

```
SELECT ename FROM emp WHERE YEAR(hiredate) = '1995';
```

22. **To display DeptNo, Job, EmpName in reverse order of salary from the EMP table.**

Ans.

```
SELECT deptno,job,empname FROM emp ORDER BY sal DESC;
```

23. **List EmpName, Job, Sal for all the employees who have a manager.**

Ans.

```
SELECT empname , job, salary from EMP WHERE mgr IS NOT NULL;
```

24. **List the minimum and maximum salary of each job type.**

Ans.

```
SELECT job,MIN(sal),MAX(sal) FROM emp GROUP BY job;
```

25. **Show the average salary for all departments with more than 3 people for job.**

Ans.

```
SELECT AVG( sal ) FROM emp GROUP BY deptno HAVING COUNT(job)>3;
```

26. **Display only the jobs with maximum salary greater than or equal to 3000.**

Ans.

```
SELECT job FROM emp GROUP BY job HAVING MAX(salary)>=3000;
```

27. **Find out number of employee having ‘Manager’ as job.**

Ans.

```
SELECT COUNT(empname) FROM emp WHERE job LIKE 'Manager';
```

28. **Create view Dept20 with EmpName and the Sal of employees for dept 20.**

Ans.

```
CREATE VIEW dept20 as SELECT empname, sal FROM emp WHERE deptno=20;
```

29. **Find the average salary and average total remuneration for each job type remember salesman earn commission.**

Ans.

```
SELECT AVG( sal ) , AVG( sal + IFNULL( comm, 0 ) ) FROM emp GROUP BY job;
```

30. **What happens if you try to drop a table on which a view exists?**

Ans. If we try to drop a table on which a view exist, then the table is dropped but DBMS invalidates these dependent views but does not drop them. We cannot use these views unless we recreate the table or drop and recreate the objects so that they no longer depend on the table.

31. **Create a view with one of the columns Salary * 12. Try updating columns of this view.**

Ans.

```
CREATE VIEW emp_view (v_empno,v_empname,v_avgsal) AS SELECT empno, empname,
salary*12 FROM emp;
UPDATE emp_view SET empname = 'MOHAN' WHERE empno=8698;
```

32. **Can you create view of view?**

Ans. Yes, We can create view of view.

33. **Write a suitable SQL statement to display ALL employees working in New York in the following format :**

EmpName Salary Location

Ans.
```
SELECT A.empname, A.salary,B.location FROM emp A INNER JOIN dept B on
A.deptno=B.deptno WHERE location LIKE 'NewYork';
```

34. **Write a suitable SQL statement to display employees' name of all the employees of GRADE 3.**

Ans.
```
SELECT empname FROM emp A, salgrade B WHERE grade=3 AND A.empno=B.empno;
```

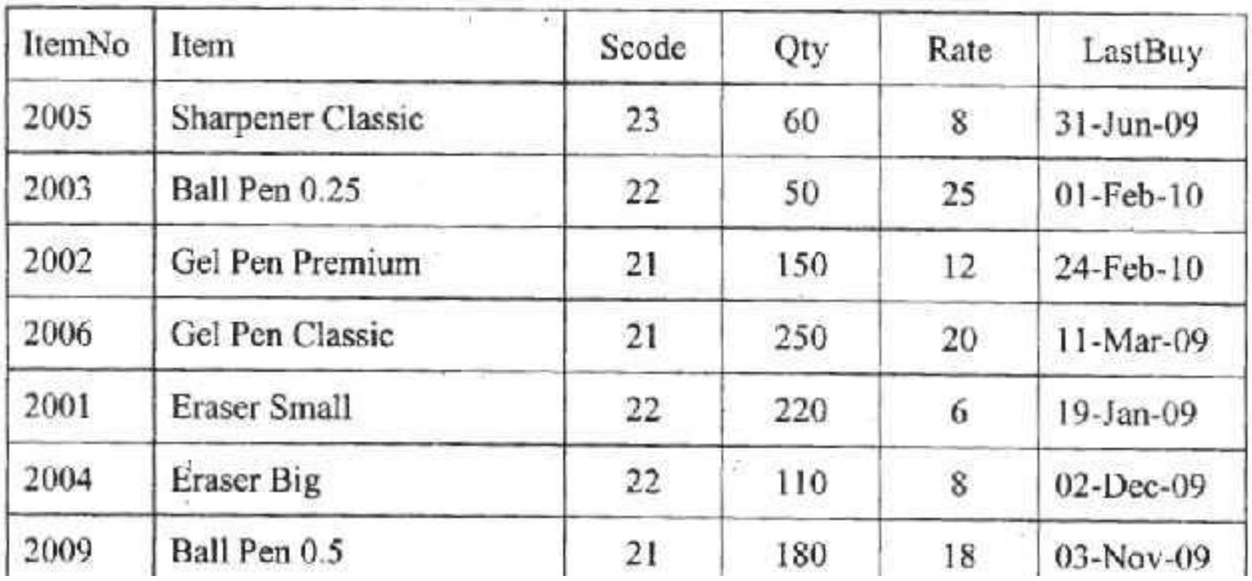
35. **Write a suitable SQL statement to find out the total number of employees from EMP table.**

Ans.
```
SELECT count(empname) from EMP;
```

TYPE B : SHORT ANSWER QUESTIONS

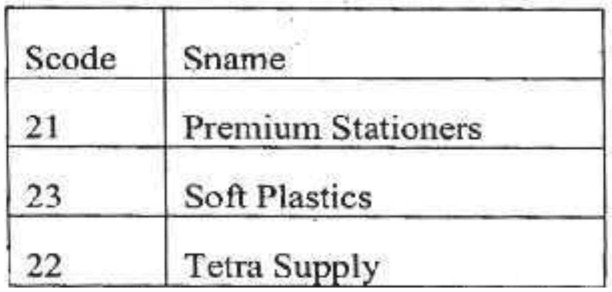
1. **Consider the following tables STORE and SUPPLIERS and answer (a) and (b) parts of this question:**

Table: STORE



ItemNo	Item	Scode	Qty	Rate	LastBuy
2005	Sharpener Classic	23	60	8	31-Jun-09
2003	Ball Pen 0.25	22	50	25	01-Feb-10
2002	Gel Pen Premium	21	150	12	24-Feb-10
2006	Gel Pen Classic	21	250	20	11-Mar-09
2001	Eraser Small	22	220	6	19-Jan-09
2004	Eraser Big	22	110	8	02-Dec-09
2009	Ball Pen 0.5	21	180	18	03-Nov-09

Table: SUPPLIERS



Scode	Sname
21	Premium Stationers
23	Soft Plastics
22	Tetra Supply

(a) **Write SQL commands for the following statements:**

(i) **To display details of all the items in the Store table in ascending order of LastBuy.**

Ans.
```
SELECT * FROM STORE ORDER BY LastBuy;
```

(ii) **To display ItemNo and Item name of those items from Store table, whose Rate is more than 15 Rupees.**

Ans.
```
SELECT ItemNo, Item FROM STORE WHERE Rate >15;
```

(iii) **To display the details of those items whose Supplier code (Scode) is 22 or Quantity in Store (Qty) is more than 110 from the table Store.**

Ans.
```
SELECT * FROM STORE WHERE Scode = 22 OR Qty >110;
```

(iv) **To display Minimum Rate of items for each Supplier individually as per Scode from the table Store.**

Ans.
```
SELECT Scode, MIN(Rate) FROM STORE GROUP BY Scode;
```

(b) Give the output of the following SQL queries:

(i) **SELECT COUNT(DISTINCT Scode) FROM Store;**

Ans.
```
COUNT(DISTINCT Scode)
3
```

(ii) **SELECT Rate*Qty FROM Store WHERE ItemNo=2004;**

Ans.
```
RATE*QTY
880
```

(iii) **SELECT Item,Sname FROM Store S, Suppliers P WHERE S.Scode=P.Scode AND ItemNo=2006;**

Ans.
```
ITEM                                SNAME
Gel Pen Classic          Premium Stationers
```

(iv) **SELECT MAX(LastBuy) FROM Store;**

Ans.
```
MAX (LASTBUY)
24-Feb-10
```

2. **Consider the following table Item and Customer. Write SQL commands for the statement (i) to (iv) and give outputs for SQL queries (v) to (viii).**

Table : ITEM

i_ID	ItemName	Manufacturer	Price
PC01	Personal Computer	ABC	35000
LC05	Laptop	ABC	55000
Pc03	Personal Computer	XYZ	32000
Pc06	Personal Computer	COMP	37000
Lc03	Laptop	PQR	57000

Table : CUSTOMER

C_ID	CustomerName	City	I_ID
01	N Roy	Delhi	LC03
06	H Singh	Mumbai	PC03
12	R Pandey	Delhi	PC06
15	C Sharma	Delhi	LC03
16	K Agarwal	Banglore	PC01

(i) **To display the details of those Customer whose City is Delhi.**

Ans.
```
SELECT * FROM CUSTOMER WHERE CITY = 'DELHI';
```

(ii) **To display the details of Item whose Price is in the range of 3500 to 55000 (Both values included).**

Ans.
```
SELECT * FROM ITEM WHERE PRICE BETWEEN 35000 AND 55000;
```

(iii) **To displa the customerName, City from table Customer, and ItemName and Price from table Item, with their corresponding matching I_ID.**

Ans.
```
SELECT CUSTOMERNAME,CITY,ITEMNAME,PRICE FROM CUSTOMER A INNER    JOIN ITEM
B WHERE A.I_ID=B.I_ID;
```

(iv) **To increase the Price of all Items by 1000 in the table Item.**

Ans.
```
UPDATE ITEM SET PRICE=PRICE+1000;
```

(v) **SELECT DISTINCT City FROM Customer;**

Ans.
```
City
Delhi
Mumbai
Banglore
```

(vi) **SELECT ItemName, Max(Price), Count(*) FROM Item GROUP BY ItemName;**

Ans.

```
Name               Max(Price)   Count(*)
Laptop             58000        2
Personal           38000        3
Computer
```

(vii) **SELECT CustomerName, Manufacturer From Item, Customer WHERE Item.I_Id=Customer.I_Id;**

Ans.

```
Cname        Manufacturer
N Roy        PQR
H Singh      XYZ
R Pandey     COMP
C Sharma     PQR
K Agarwal    ABC
```

(viii) **SELECT ItemName, Price * 100 FROM Item WHERE Manufacturer = 'ABC';**

Ans.

```
Name                   Price*100
Personal  Computer     3600000
Laptop                 5600000
```

3. **Consider the following tables. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).**

TABLE : SENDER

SenderID	SenderName	SenderName	SenderCity
ND01	R Jain	2,ABC Appts	New Delhi
MU02	H Sinha	12, Newtown	Mumbai
MU15	S Jha	27/A, Park Street	Mumbai
ND50	T Prasad	122-K, SDA	New Delhi

TABLE : RECIPIENT

RecID	SenderID	RecName	RecAddress	ReCity
KO05	ND01	R Bajpayee	5, Central Avenue	Kolkata
ND08	MU02	S Mohan	116, A vihar	New Delhi
MU19	ND01	H singh	2a, Andheri east	Mumbai
MU32	MU15	P K Swamy	B5, c S Terminus	Mumbai
ND48	ND50	S Tirupathi	13, B1 d, Mayur vihar	New Delhi

(i) **To display the names of all Senders from Mumbai**

Ans.
```
SELECT SENDERNAME FROM SENDER WHERE SENDERCITY='MUMBAI';
```

(ii) **To display the RecID, SenderName, SenderAddress, RecName, RecAddess for every Recipient**

Ans.
```
SELECT RECID, SENDERNAME,SENDERADDRESS,RECNAME,RECADDRESS FROM RECIPIENT A
INNER JOIN SENDER B ON A.SENDERID=B.SENDERID;
```

(iii) **To display Recipient detail in asending order of RecName**

Ans.
```
SELECT * FROM RECIPIENT ORDER BY RECNAME;
```

(iv) **To display number of Recipients from each city**

Ans.
```
SELECT RECCITY,COUNT(RECNAME)  FROM RECIPIENT GROUP BY RECCITY;
```

(v) **SELECT DISTINCT Sendercity FROM Sender;**

Ans.
```
Sendercity
New Delhi
Mumbai
```

(vi) **SELECT A.SenderName, B.RecName FROM Sender A, Recipient B WHERE A.SenderID=B.SenderID AND B.RecCity='Mumbai';**

Ans.
```
SenderName      RecName
R Jain          H Singh
S Jha           P K Swamy
```

SELECT RecName, RecAddress FROM Recipient WHERE recCity NOT IN('Mumbai', 'Kolkata');

(vii)
Ans.
```
RecName      RecAddress
S Mahajan    116, A Viharl
S Tirupati   13, B1 D, Mayur Vihar
```

(viii) **SELECT RecID, RecName FROM Recipient WHERE SenderID='MU02' OR SenderID='ND50';**

Ans.
```
RecID     RecName
ND08      S Mahajan
ND48      S Tirupati
```

4. (a) **What happens if you drop a table on which a view exists?**

Ans. If we try to drop a table on which a view exist, then the DBMS like Oracle invalidates these dependent views but does not drop them. We cannot use these views unless we recreate the table or drop and recreate the objects so that they no longer depend on the table.

Note: *Write the SQL commands for (b) to (g) and write outputs for SQL commands given in (h) on the basis of table MOV*

Table : MOV

No.	Title	Type	Rating	Stars	Qty	Price
1	Gone with the Wind	Drama	G	Gable	4	39.95
2	Friday the 13th	Horror	R	Jason	2	69.95
3	Top Gun	Drama	PG	Cruise	7	49.95
4	Splash	Comedy	PG13	Hanks	3	29.95
5	Independence Day	Drama	R	Turner	3	19.95
6	Risky Business	Comedy	R	Cruise	2	44.95
7	Cocoon	Scifi	PG	Ameche	2	31.95
8	Crocodile Dundee	Comedy	PG13	Harris	2	69.95
9	101 Dalmatians	Comedy	G		3	59.95
10	Tootsie	Comedy	PG	Hoffman	1	29.95

(b) **Find the total value of the movie cassettes available in the library.**

Ans.
```
SELECT COUNT(TITLE) FROM MOV;
```

(c) **Display a list of all movies with Price over 20 and sorted by Price.**

Ans.
```
SELECT * FROM MOV WHERE PRICE>20 ORDER BY PRICE;
```

(d) **Display all the movies sorted by Qty in decreasing order.**

Ans.
```
SELECT * FROM MOV ORDER BY QTY DESC;
```

(e) **Display a report listing a movie number, current value and replacement value for each movie in the above table. Calculate the replacement value for all movies as**

QTY * Price * 1.15

Ans.
```
SELECT NO,PRICE AS 'CURRENT VALUE',(QTY*PRICE*1.15) AS 'REPLACEMENT VALUE'
FROM MOV;
```

(f) **Count the number of movies where Rating is not "G".**

Ans.
```
SELECT COUNT(TITLE) FROM MOV WHERE RATING<>'G';
```

(g) **Insert a new movie in MOV table. Fill all the columns with values.**

Ans.
```
INSERT IN TO MOV VALUES(11,'Republic Day','Drama','R','Turner',3,38.95);
```

(h) **Give the output of following SQL commands on the basis of table MOV.**

(i) **Select AVG(Price) from MOV where Price < 30;**

Ans.
```
AVG(Price)
19.95
```

(ii) **Select MAX(Price) from MOV where price > 30;**

Ans.
```
MAX(Price)
79.95
```

(iii) **Select SUM(Price * QTY) from MOV where QTY < 4;**

Ans.
```
SUM(Price*QTY)
791.75
```

(iv) **Select COUNT(DISTINCT TYPE) from MOV;**

Ans.
```
COUNT(DISTINCT TYPE)
4
```

5. **Write SQL statement to create EMPLOYEE relation which contains EmpNo, Name, Skill, PayRate.**

Ans.
```
CREATE TABLE EMPLOYEE(EmpNo VARCHAR(10), Name CHAR(20), Skill CHAR(20),
PayRate DECIMAL);
```

6. **Create a table with the under mentioned structures**

Table : EMP		**Table : PROJECT**		**Table : DEPT**	
EmpNo	NUMBER(4)	ProjId	NUMBER(4)	DeptNo	NUMBER(2)
DeptNo	NUMBER(2)	ProjDesig	CHAR(20)	DeptName	CHAR(12)
EmpName	CHAR(10)	ProjStartDT	DATE	Location	CHAR(12)
Job	CHAR(10)	ProjEndDT	DATE		
Manager	NUMBER(4)	BudgetAmount	NUMBER(7)		
Hiredate	DATE	MaxNoStaff	NUMBER(2)		
Salary	NUMBER(7,2)				
Commission	NUMBER(7,2)				

Ans.
```
CREATE TABLE EMP(EmpNo INTEGER(4), DeptNo INTEGER(2), EmpName CHAR(10), Job
CHAR(10), Manager INTEGER(4), HireDate DATE, Salary DECIMAL(7,2), Commission
DECIMAL(7,2));

CREATE TABLE PROJECT(ProjId INTEGER(4), ProjDesign CHAR(20), ProjStartDT
DATE, ProjEndDT DATE, BudgetAmount INTEGER(7), MaxNoStaff INTEGER(2));

CREATE TABLE DEPT(DeptNo INTEGER(2), DeptName CHAR(12), Location CHAR(12));
```

7. **Create a table called SALEGRADE with the columns specified beow :**

LowSal	NUMBER(7,2)
HighSal	NUMBER(7,2)
Grade	NUMBER(2)

Ans.
```
CREATE TABLE SALEGRADE(LowSal DECIMAL(7,2) CHECK(LowSal>=1000.00), HighSal
DECIMAL(7,2) CHECK(HighSal<=10000.00), Grade INTEGER);
```

8. **Write SQL commands for (a) to (f) and write the outputs for (g) on the basis of tables FURNITURE and ARRIVALS:**

TABLE : FURNITURE

NO	ITEMNAME	TYPE	DATEOFSTOCK	PRICE	DISCOUNT
1	White lotus	Double Bed	23/02/02	30000	25
2	Pink feather	Baby cot	20/01/02	7000	20
3	Dolphin	Baby cot	19/02/02	9500	20
4	Decent	Office Table	01/01/02	25000	30

5	Comfort zone	Double Bed	12/01/02	25000	25
6	Donald	Baby cot	24/02/02	6500	15
7	Royal Finish	Office Table	20/02/02	18000	30
8	Royal tiger	Sofa	22/02/02	31000	30
9	Econo sitting	Sofa	13/12/01	9500	25
10	Eating Paradise	Dining Table	19/02/02	11500	25

TABLE : ARRIVALS

NO	ITEMNAME	TYPE	DATEOFSTOCK	PRICE	DISCOUNT
11	Wood Comfort	Double Bed	23/03/03	25000	25
12	Old Fox	Sofa	20/02/03	17000	20
13	Micky	Baby cot	21/02/03	7500	15

(a) **To show all information about the Baby cots from the FURNITURE table.**

Ans.
```
SELECT * FROM FURNITURE WHERE TYPE='Baby cot';
```

(b) **To list the ITEMNAME which are priced at more than 15000 from the FURNITURE table.**

Ans.
```
SELECT ITEMNAME  FROM FURNITURE WHERE PRICE>15000;
```

(c) **To list ITEMNAME and TYPE of those items, in which DATEOFSTOCK is before 22/01/02 from the FURNITURE table in descending order of ITEMNAME.**

Ans.
```
SELECT ITEMNAME, TYPE FROM FURNITURE WHERE DATEOFSTOCK<{22/01/02} ORDER BY
ITEMNAME DESC;
```

(d) **To display ITEMNAME and DATEOFSTOCK of those items, in which the DISCOUNT percentage is more than 25 form FURNITURE table.**

Ans.
```
SELECT ITEMNAME,DATEOFSTOCK FROM FURNITURE WHERE DISCOUNT>25;
```

(e) **To count the number of items, whose TYPE is "Sofa" from FURNITURE table.**

Ans.
```
SELECT COUNT(TYPE) FROM FURNITURE WHERE TYPE='SOFA';
```

(f) **To insert a new row in the ARRIVALS table with the following data:**

14, 'Velvet touch', 'Double bed', {25/03/03}, 25000,30

Ans.
```
INSERT INTO ARRIVALS VALUES(14, 'Velvet touch', 'Double bed', {25/03/03},
25000,30);
```

(g) **Give the output of following SQL statement:**

NOTE : Outputs of the below mentioned queries should be based on original data given in both the tables, i.e., without considering the insertion done in (f) part of this question :

(i) **Select COUNT (distinct TYPE) from FURNITURE;**

Ans.
```
COUNT(distinct TYPE)
5
```

(ii) **Select MAX(DISCOUT) form FURNITURE, ARRIVALS;**

Ans.
```
MAX(DISCOUNT)
30,25
```

(iii) **Select AVG(DISCOUT) form FURNITURE where TYPE = 'Baby cot';**

Ans.
```
AVG(DISCOUT)
15
```

(iv) **Select SUM(PRICE) from FURNITURE where DATEOFSTOCK<{12/02/02};**

Ans.
```
SUM(PRICE)
66500
```

9. **Differentiate between SQL commands DROP TABLE and DROP VIEW.**

Ans. DROP TABLE:- DROP TABLE statement is used to delete the table and all its data from the database entirely. The syntax for DROP TABLE is DROP TABLE ;

DROP VIEW:- Removes an existing view from a database. DROP VIEW statement is used to remove a view or an object view from the database. The syntax for DROP VIEW is DROP VIEW ;

10. **Study the following tables DOCTOR and SALARY and write SQL commands for the questions (i) to (iv) and give outputs for SQL queries (v) to (vi) :**

TABLE : DOCTOR

ID	NAME	DEPT	SEX	EXPERIENCE
101	John	ENT	M	12
104	Smith	ORTHOPEDIC	M	5
107	George	CARDIOLOGY	M	10
114	Lara	SKIN	F	3
109	K George	MEDICINE	F	9
105	Johnson	ORTHOPEDIC	M	10
117	Lucy	ENT	F	3
111	Bill	MEDICINE	F	12
130	Morphy	ORTHOPEDIC	M	15

TABLE : SALARY

1D	BASIC	ALLOWANCE	CONSULTATION
101	12000	1000	300
104	23000	2300	500
107	32000	4000	500
114	12000	5200	100
109	42000	1700	200
105	18900	1690	300
130	21700	2600	300

(i) **Display NAME of all doctors who are in "MEDICINE" having more than 10 year experience from the table DOCTOR.**

Ans.
```
SELECT NAME FROM DOCTOR WHERE DEPT='MEDICINE' AND   EXPERIENCE>10;
```

(ii) **Display the average salary of all doctors working in "ENT" department using the tables DOCTOR and SALARY. Salary=BASIC + ALLOWANCE.**

Ans.
```
SELECT AVG(BASIC+ALLOWANCE) FROM DOCTOR A, SALARY B WHERE DEPT='ENT' AND
A.ID=B.ID;
```

(iii) **Display the minimum ALLOWANCE of female doctors.**

Ans.
```
SELECT MIN(ALLOWANCE) FROM DOCTOR A AND SALARY B WHERE SEX='F' AND
A.ID=B.ID;
```

(iv.) **Display the highest consultation fee among all male doctor.**

Ans.
```
SELECT MAX(CONST) FROM DOCTOR A, SALARY B WHERE SEX='M' AND A.ID=B.ID;
```

(v) **SELECT count(*) from DOCTOR where SEX="F".**

Ans.
```
count(*)
4
```

(vi) **SELECT NAME, DEPT, BASIC from DOCTOR Salary WHERE DEPT="ENT" AND DOCTORID=SALARY.ID**

Ans.
```
NAME          DEPT        BASIC
John          ENT         12000
```

11. (a) **What are DDL and DML commands?**

Ans. DDL is short form of Data Definition Language statements are used to build and modify the structure of database, tables and other objects in the database. When you execute a DDL statement, it takes effect immediately. Some of the commands comprising DDL are CREATE TABLE, DROP TABLE and CREATE INDEX.

DML is abbreviation of Data Manipulation Language. It is used to retrieve, store, modify, delete, insert and update data in database. Examples: SELECT, UPDATE, INSERT statements.

(b) **Study the following tables FLIGHTS and FARES and write SQL commands for the questions (i) to (iv) and give outputs for SQL queries (v) to (vi).**

Table : FLIGHTS

FL_NO	STARTING	ENDING	NO_FLIGHTS	NO_STOPS
IC301	MUMBAI	DELHI	8	0
IC799	BANGALORE	DELHI	2	1
MC101	INDORE	MUMBAI	3	0
IC302	DELHI	MUMBAI	8	0
AM812	KANPUR	BANGALORE	3	1
IC899	MUMBAI	KOCHI	1	4
AM501	DELHI	TRIVANDRUM	1	5
MU499	MUMBAI	MADRAS	3	3
IC701	DELHI	AHMEDABAD	4	0

Table : FARES

FL_NO	AIRLINES	FARE	TAX%
IC701	Indian Airlines	6500	10
MU499	Sahara	9400	5
AM501	Jet Airways	13450	8
IC899	India Airlines	8300	4
IC302	Indian Airlines	4300	10
IC799	Indian Airlines	10500	10
MC101	Deccan Airlines	3500	4

(i) **Display FL_NO and NO_FLIGHTS from “KANPUR” to “BANGLORE” from the table FLIGHTS.**

Ans.
```
SELECT FL_NO,NO_FLIGHTS FROM FLIGHTS WHERE ‘STARTING’ LIKE 'KANPUR' AND
ENDING LIKE 'BANGALORE';
```

(ii) **Arrange the contents of the table FLIGHTS in the ascending order of FL_NO.**

Ans.
```
SELECT * FROM FLIGHTS ORDER BY FL_NO
```

(iii) **Display the FL_NO and fare to be paid for the flights from DELHI to MUMBAI using the tables FLIGHTS and FARES, where the fare to be paid=FARE + FARE*TAX%/100.**

Ans.
```
SELECT `FLIGHTS`.`FL_NO`,(`FARE`+`FARE`*(`TAX%`/100)) FROM `FLIGHTS`,`FARES`
WHERE `STARTING` LIKE 'DELHI' AND `ENDING` LIKE 'MUMBAI' AND
`FLIGHTS`.`FL_NO` = `FARES`.`FL_NO`;
```

(iv) **Display the minimum fare “Indian Airlines” is offering from the table FARES.**

Ans.
```
SELECT MIN(FARE) FROM FARES;
```

(v) **SELECT FL_NO, NO_FLIGHTS, AIRLINES from FLIGHTS, FARES WHERE STARTING = “DELHI” AND FLIGHTS.FL_NO=FARES.FL_NO.**

Ans.
```
ERROR - Column 'FL_NO' in field list is ambiguous
```
Correct Code and Ans.

```
SELECT FLIGHTS.FL_NO,NO_FLIGHTS,FARES.AIRLINES FROM FLIGHTS,FARES
WHERE FLIGHTS.STARTING LIKE 'DELHI' AND FLIGHTS.FL_NO = FARES.FL_NO;
FL_NO   NO_FLIGHTS     AIRLINES
IC302   8              Indian Airlines
AM501   1              Jet Airways
IC701   4              Indian Airlines
```

(vi) **SELECT count(distinct ENDING) from FLIGHTS.**

Ans.

```
count(distinct ENDING)
7
```

12. **Consider the following tables WORKERS and DESIG. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).**

Table : Workers

W_ID	FIRSTNAME	LASTNAME	ADDRESS	CITY
102	Sam	Tones	33 Elm St.	Paris
105	Sarah	Ackerman	440 U. S. 110	New York
144	Manila	Sengupta	24 Friends Street	New Delhi
210	George	Smith	83 First Street	Howard
255	Mary	Jones	842 Vine Ave.	Losantiville
300	Robert	Samuel	9 Fifth Cross	Washington
335	Henry	Williams	12 Moore Street	Boston
403	Ronny	Lee	121 Harrison St.	New York
451	Pat	Thompson	11 Red Road	Paris

Table : DESIG

W_ID	SALARY	BENEFITS	DESIGNATION
102	75000	15000	Manager
105	85000	25000	Director
144	70000	15000	Manager
210	75000	12500	Manager
255	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
400	32000	7500	Salesman
451	28000	7500	Salesman

(i) **To display W_ID Firstname, Address and City of all employees living in New York from the table WORKERS.**

Ans.

```
SELECT W_ID,FIRSTNAME,ADDRESS,CITY FROM WORKERS WHERE CITY='NEW YORK';
```

(ii) **To dislay the content of WORKERS table in ascending order of LASTNAME.**

Ans.

```
SELECT * FROM WORKERS ORDER BY LASTNAME;
```

(iii) **To display the Firstname, Lastname, and Total Salary of all clerk from the tables WORKERS and DESIG, where Total Salary is calculated as Salary + Benefits.**

Ans.

```
SELECT FIRSTNAME,LASTNAME,SALARY+BENEFITS AS 'TOTAL SALARY' FROM
WORKERS,DESIG WHERE WORKERS.W_ID=DESIG.W_ID;
```

(iv) **To display the Minimum salary among Managers and Clerks from the table DESIG.**

Ans.

```
SELECT MIN(SALARY) FROM DESIG WHERE DESIGNATION IN('MANAGER','CLERK');
```

(v) **SELECT FIRSTNAME, SALARY FROM WORKERS, DESIG WHERE DESIGNATION = ‘Manager’ AND WORKERS.W_ID=DESIG.W_ID;**

Ans.

```
FIRSTNAME  SALARY
Sam        75000
Manila     70000
George     75000
```

(vi) **SELECT COUNT(DISTINCT DESIGNATION) FROM DESIG;**

Ans.

```
COUNT(DISTINCT DESIGNATION)
4
```

(vii) **SELECT DESIGNATION, SUM(SALARY) FROM DESIG GROUP BY DESIGNATION HAVING COUNT(*)<3;**

Ans.

```
DESIGNATION      SUM(SALARY)
Director         85000
Salesman         60000
```

(viii) **SELECT SUM(BENEFITS) FROM DESIG WHERE DESIGNATION='Salesman';**

Ans.

```
SUM(BENEFITS)
15000
```

13. **Consider the following tables GARMENT and FABRIC. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).**

Table: GARMENT

GCODE	DESCRIPTION	PRICE	FCODE	READYDATE
10023	PENCIL SKIRT	1150	F03	19-DEC-08
10001	FORMAL SHIRT	1250	F01	12-JAN-08
10012	INFORMAL SHIRT	1550	F02	06-JAN-08
10024	BABY TOP	750	F03	07-APR-07
10090	TULIP SKIRT	850	F02	31-MAR-07
10019	EVENING GOWN	850	F03	06-JUN-08
10009	INFORMAL PANT	1500	F02	20-OCT-08
10007	FORMAL PANT	1350	F01	09-MAR-08
10020	FROCK	850	F04	09-SEP-07
10089	SLACKS	750	F03	20-OCT-08

Table: FABRIC

FCODE	TYPE
F04	POLYSTER
F02	COTTON
F03	SILK
F01	TERELENE

(i) **To display GCODE and DESCRIPTION of each GARMENT in descending order of GCODE**

Ans.

```
SELECT GCODE, DESCRIPTION FROM GARMENT ORDER BY GCODE DESC;
```

(ii) **To display the details of all the GARMENTs, which have READYDATE in between 08-DEC-07 and 16-JUN-08 (inclusive of both the dates).**

Ans.

```
SELECT * FROM GARMENTWHERE READYDATE BETWEEN '08-DEC-07'AND '16-JUN-08';
```

(iii) **To display the average PRICE of all the GARMENTs, which are made up of FABRIC with FCODE as F03.**

Ans.

```
SELECT AVG(PRICE) FROM GARMENTWHERE FCODE = 'F03';
```

(iv) **To display FABRICwise highest and lowest price of GARMENTs from GARMENT table. (Display FCODE of each GARMENT along with highest and lowest price).**

Ans.

```
SELECT FCODE, MAX(PRICE), MIN(PRICE) FROM GARMENT GROUP BY FCODE;
```

(v) **SELECT SUM(PRICE) FROM GARMENT WHERE FCODE='F01';**

Ans.
```
SUM(PRICE)
2600
```

(vi) **SELECT DESCRIPTION, TYPE FROM GARMENT, FABRIC WHERE GARMENT.FCODE =FABRIC.FCODE AND GARMENT.PRICE > = 1260;**

Ans.
```
DESCRIPTION TYPE
INFORMAL SHIRT COTTON
INFORMAL PANT COTTON
FORMAL PANT TERELENE
```

(vii) **SELECT MAX(FCODE) FROM FABRIC;**

Ans.
```
MAX(FCODE)
F04
```

(viii) **SELECT COUNT (DISTINCT PRICE) FROM GARMENT;**

Ans.
```
COUNT(DISTINCT PRICE)
7
```

14. **Consider the following tables DRESS and MATERIAL. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).**

Table : DRESS

DCODE	DESCRIPTION	PRICE	MCODE	LAUNCHDATE
10001	FORMAL SHIRT	1250	M001	12-JAN-08
10020	FROCK	750	M004	09-SEP-07
10012	ONFORMAL SHIRT	1450	M002	06-JUN-08
10019	EVENING GOWN	850	M003	06-JUN-08
10090	TULIP SKIRT	850	M002	31-MAR-07
10023	PENCIL SKIRT	1250	M003	19-DEC-08
10089	SLACKS	850	M003	20-OCT-08
10007	FORMAL PANT	1450	M001	09-MAR-08
10009	INFORMAL PANT	1400	M002	20-OCT-08
10024	BABY TOP	650	M003	07-APR-07

Table : MATERIAL

MCODE	TYPE
M001	TERELENE
M002	COTTON
M004	POLYESTER
M003	SILK

(i) **To display DCODE and DISCRIPTION of each dress in ascending order of DCODE.**

Ans.
```
SELECT DCODE,DESCRIPTION FROM DRESS ORDER BY DCODE;
```

(ii) **To display the details of all the dresses which have LAUNCHDATE in between 05-DEC-07 AND 20-JUN-08 (inclusive of both the dates).**

Ans.
```
SELECT * FROM DRESS WHERE LAUNCHDATE BETWEEN '05-DEC-07' AND '20-JUN-08';
```

(iii) **To display the average PRICE of all the dresses which are made up of material with MCODE as M003.**

Ans.
```
SELECT AVG(PRICE) FROM DRESS WHERE MCODE='M003';
```

(iv) **To display materialwie highest and lowest price of dresses from DRESS table. (Display MCODE of each dress along with highest and lowest price)**

Ans.
```
SELECT B.MCODE,TYPE,MAX(PRICE) AS "HIGHEST",MIN(PRICE) AS "LOWEST" FROM
DRESS A, MATERIAL B WHERE A.MCODE=B.MCODE GROUP BY TYPE;
```

(v) **SELECT SUM(PRICE) FROM DRESS WHERE MCODE = 'M001';**

Ans.
```
SUM(PRICE)
2700
```

(vi) **SELECT DESCRIPTION, TYPE FROM DRESS, MATERIAL WHERE DRESS.MCODE=MATERIAL.MCODE AND DRESS.PRICE >= 1250;**

Ans.
```
DESCRIPTION      TYPE
FORMAL SHIRT     TERELENE
INFORMAL SHIRT COTTON
PENCIL SKIRT     SILK
FORMAL PANT      TERELENE
INFORMAL PANT    COTTON
```

(vii) **SELECT MAX(MCODE) FROM MATERIAL;**

Ans.
```
MAX(MCODE)
M004
```

(viii) **SELECT COUNT(DISTINCT PRICE) FROM DRESS;**

Ans.
```
COUNT(DISTINCT PRICE)
6
```

15. **Consider the following tables Stationery and Consumer. Write SQL commands for the statement (i) to (iv) and give output for SQL queries (v) to (viii).**

Table : STATIONERY

S_ID	StationeryName	Company	Price
DP01	Dot Pen	ABC	10
PL02	Pencil	XYZ	6
ER05	Eraser	XYZ	7
PL01	Pencil	CAM	5
GP02	Gel Pen	ABC	15

Table : CONSUMER

C_ID	ConsumerName	Address	S_ID
01	Good Lerner	Delhi	PL01
06	Write Well	Mumbai	GP02
12	Topper	Delhi	DP01
15	Write & Draw	Delhi	PL02
16	Motivation	Banglore	PL01

(i) **To display the details of those Consumers whose Address is Delhi.**

Ans.
```
SELECT * FROM CONSUMER WHERE ADDRESS="DELHI";
```

(ii) **To display the details of Stationery whose Price is in the range of 8 to 15 (Both value included)**

Ans.
```
SELECT * FROM STATIONERY WHERE PRICE BETWEEN 8 AND 15;
```

(iii) **To display the ConsumerName, Address from Tble Consumer, and Company and Price from table Stationery, with their corresponding matching S_ID**

Ans.
```
SELECT CONSUMERNAME,ADDRESS,COMPANY,PRICE FROM CONSUMER,STATIONERY WHERE
CONSUMER.S_ID=STATIONERY.S_ID;
```

(iv) **To increase the Price of all stationery by 2**

Ans.
```
UPDATE STATIONERY SET PRICE=PRICE+2;
```

(v) **SELECT DISTINCT Address FROM Consumer;**

Ans.

```
Address
Delhi
Mumbai
Banglore
```

(vi) **SELECT Company, MAX(Price),Min(Price),Count(*) FROM Stationery GROUP BY Company;**

Ans.

```
Company    MAX(Price) Min(Price) Count(*)
ABC        17         12         2
CAM        7          7          1
XYZ        9          8          2
```

(vii) **SELECT Consumer.CnsumerName, stationery.stationeryName, Stationery.Price FROM Stationery, Consumer WHERE Consumer.S_Id=Stationery.S_Id**

Ans.

```
     ConsumerName    StationeryName  Price
     Good Lerner     Pencil          7
     Write Well      Gel Pen         17
     Topper          Dot Pen         12
     Write & Drow    Pencil          8
     Motivation      Pencil          7
```

(viii) **SELECT StationeryName, Price * 3 FROM Stationery**

Ans.

```
         StationeryName   Price*3
         Dot Pen          36
         Pencil           24
         Eraser           27
         Pencil           21
         Gel Pen          51
```