DATA BASE MANAGEMENT SYSTEM

LAB PRACTICAL FILE

(DBMS LAB MANUAL)

SCHOOL OF COMPUTER APPLICATIONS

INTRODUCTION TO SQL :

-SQL (Structured Query Language) is a standard programming language specifically for managing and manipulating relational databases. It is used to create, read, update, and delete data in a structured way.

-SQL data types help define the kind of data that can be stored in each column of a table.

: Here are some common SQL data types –

1. Numeric Data Types
   - o INT: Integer numbers, e.g., 1, 100, -20.
   - o DECIMAL (p, s): Fixed precision numbers with specified digits after the decimal.
   - o FLOAT and REAL: For floating-point numbers (decimal numbers with variable precision).
2. Character Data Types
   - o CHAR(n): Fixed-length strings (e.g., CHAR(10) reserves 10 characters).
   - o VARCHAR(n): Variable-length strings (e.g., VARCHAR(50) allows up to 50 characters).
   - o TEXT: Large amounts of text.
3. Date and Time Data Types
   - o DATE: Stores date values (year, month, day).
   - o TIME: Stores time values (hours, minutes, seconds).
   - o DATETIME: Stores both date and time values.
   - o TIMESTAMP: Stores date and time with time zone info.

4 . Boolean Data Types

- BOOLEAN: Stores true/false values.

5. Binary Data Types

- BLOB: Stores binary data, often used for images or files.

SQL Command Categories:

DDL, DML, and DCL SQL commands are organized into categories based on their purpose:

1. DDL (Data Definition Language)
   - Used to define and manage database structures.
   - Common DDL commands:
     - ♣ CREATE: Creates a new database, table, or other objects.
     - ♣ ALTER: Modifies an existing database object, such as adding a column.
     - ♣ DROP: Deletes a database object like a table or view.
     - ♣ TRUNCATE: Removes all rows from a table without logging individual row deletions.

2. DML (Data Manipulation Language)
   - Used to interact with data within tables.
   - Common DML commands:
     - ♣ SELECT: Retrieves data from one or more tables.
     - ♣ INSERT: Adds new rows to a table.
     - ♣ UPDATE: Modifies existing data within a table.
     - ♣ DELETE: Removes rows from a table.
3. DCL (Data Control Language)
   - Used to manage permissions and control access to data.
   - Common DCL commands:
     - ♣ GRANT: Gives a user access privileges to a database or table.
     - ♣ REVOKE: Removes access privileges from a user.

Experiment : 01

1 : Create the following tables :

Student_table :

Column_name	Data type	size	constraint
StudentId	Number	4	Primary key
studentname	Varchar2	40	Null
Address1	Varchar2	300	
Gender	Varchar2	15	
Course	Varchar2	8	

Course_table :

Dept No	Number	2	constraint
Dname	varchar	20	Primary key
Location	varchar	10	

1: Insert five records for each table :

```
CREATE TABLE student_0 (std_id int(4) , std_name char(20) , std_address varchar(20)  ,
Std_course char(10) , std_emailid varchar(30)  ) ;

insert into student_0 values ('11' , 'deny' , 'fbd-sec21'  , 'BCA' , 'deny@gmail.com'
);
insert into student_0 values ('12' , 'john' , 'fbd-sec25'  , 'BCA' , 'john@gmail.com'
);
insert into student_0 values ('13' , 'james' , 'fbd-sec22'  , 'MCA' , 'james@gmail.com
);
insert into student_0 values ('14' , 'daisy' , 'fbd-sec29'  , 'BCA', 'daisy@gmail.com'
 );
insert into student_0 values ('15' , 'preety' , 'fbd-sec29'  , 'MCA',
'preety@gmail.com');
```

Student_0

std_id	std_name	std_address	Std_course	std_emailid
11	deny	fbd-sec21	BCA	deny@gmail.com
12	john	fbd-sec25	BCA	john@gmail.com
13	james	fbd-sec22	MCA	james@gmail.com
14	daisy	fbd-sec29	BCA	daisy@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

```
CREATE TABLE coursee(dept_no int(4) , dept_name char(20) , dept_location varchar(20)
) ;

insert into coursee values ('11' , 'SCA' , 'C-BLOCK'     );
insert into coursee values ('12' , 'BBA' , 'T-BLOCK'   );
insert into  coursee values ('13' , 'SCA' , 'C-BLOCK'  );
insert into coursee values ('14' , 'LAW' , 'G-BLOCK'  );
insert into coursee values ('15' , 'SCA' , 'C-BLOCK' );
```

Coursee

dept_no	dept_name	dept_location
11	SCA	C-BLOCK
12	BBA	T-BLOCK
13	SCA	C-BLOCK
14	LAW	G-BLOCK
15	SCA	C-BLOCK

(Navya)

2. List all information about all students from student table:

Input

```
select * from student_0
```

Output

std_id	std_name	std_address	Std_course	std_emailid
11	deny	fbd-sec21	BCA	deny@gmail.com
12	john	fbd-sec25	BCA	john@gmail.com
13	james	fbd-sec22	MCA	james@gmail.com
14	daisy	fbd-sec29	BCA	daisy@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

3. List all student numbers along with their Courses :

Input

```
select std_id , std_course from student_0
```

Output

std_id	Std_course
11	BCA
12	BCA
13	MCA
14	BCA
15	MCA

4. List Course names and locations from the Course table :

Input

```
select dept_name , dept_location from coursee
```

(Navya)

Output

dept_name	dept_location
SCA	C-BLOCK
BBA	T-BLOCK
SCA	C-BLOCK
LAW	G-BLOCK
SCA	C-BLOCK

5. List the details of the Students in MCA Course :

Input

```
select * from student_0
where std_course = 'MCA' ;
```

std_id	std_name	std_address	Std_course	std_emailid
13	james	fbd-sec22	MCA	james@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

6. List the students details in ascending order of course :

Input

```
select * from student_0 order by std_course asc ;
```

Output

std_id	std_name	std_address	Std_course	std_emailid
11	deny	fbd-sec21	BCA	deny@gmail.com
12	john	fbd-sec25	BCA	john@gmail.com
14	daisy	fbd-sec29	BCA	daisy@gmail.com
13	james	fbd-sec22	MCA	james@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

(Navya)

7. List the number of Students in BCA course :

```
SELECT COUNT(*) AS BCA_Students FROM Student1 WHERE Course = 'BCA';
```

BCA_Students
4

8. List the number of students available in student table .

```
SELECT COUNT(*) AS Total_Students FROM Student1;
```

Total_Students
5

9. Create a table with a primary key constraint.

Input

Run SQL

```
create table myy_subject (sub_name char(10) , subject_code varchar(20) );
```

Myy_subject

sub_name	subject_code
empty	

(Navya)

10. Create a table with all column having not null constraints .

Input

Run SQL

```
create table my_carsss (d_name char(10) not null , d_code varchar(20) not null , d_std
char(20) not null );
```

My_carsss

d_name	d_code	d_std
empty		

11. Create a foreign key constraint in a table .

```
CREATE TABLE Enrollment (
    EnrollmentID NUMBER(4) PRIMARY KEY,
    StudentId NUMBER(4),
    CourseID NUMBER(2),
    FOREIGN KEY (StudentId) REFERENCES Student(StudentId),
    FOREIGN KEY (CourseID) REFERENCES Course(DeptNo)
);
```

Enrollment

EnrollmentID	StudentId	CourseID
empty		

12. Create a Table with a unique key constraint .

Input

Run SQL

```
create table studnt (st_id int(10) primary key , st_name char(20) not null ,std_email
varchar(20) unique  );
```

Studnt

st_id	st_name	std_email
empty		

13. Display list of student ordered by course .

Input

```
select * from student_0 order by std_course ;
```

Output

std_id	std_name	std_address	Std_course	std_emailid
11	deny	fbd-sec21	BCA	deny@gmail.com
12	john	fbd-sec25	BCA	john@gmail.com
14	daisy	fbd-sec29	BCA	daisy@gmail.com
13	james	fbd-sec22	MCA	james@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

14. Display alphabetically sorted list of students .

Input

```
select * from student_0 order by std_name ASC ;
```

Output

std_id	std_name	std_address	Std_course	std_emailid
14	daisy	fbd-sec29	BCA	daisy@gmail.com
11	deny	fbd-sec21	BCA	deny@gmail.com
13	james	fbd-sec22	MCA	james@gmail.com
12	john	fbd-sec25	BCA	john@gmail.com
15	preety	fbd-sec29	MCA	preety@gmail.com

(Navya

)

Experiment : 2

Q1: Create the following tables :

Customer :

SID	**Primary key**
Last_Name	
First_Name	

Orders :

Order_ID	**Primary key**
Order_Date	
Customer_sid	**Foreign key**
Amount	**Check > 20000**

1: Insert five records for each table .

```
CREATE TABLE Persons (
    PersonID int, LastName varchar(255), FirstName varchar(255));
```

```
insert into persons values  ('1' , 'priya' , 'kumari' );
insert into persons values    ('2' , 'priyanshi' , 'gill' ) ;
insert into persons values   ('3' , 'priyanka' , 'thakur ' ) ;
insert into persons values   ('4' , 'prisha' , 'singh' ) ;
insert into persons values   ('5' , 'priyamvada' , 'rathi' ) ;
```

Persons

PersonID	LastName	FirstName
1	priya	kumari
2	priyanshi	gill
3	priyanka	thakur
4	prisha	singh
5	priyamvada	rathi

Input

```
CREATE TABLE ORDERS (
    Order_ID INT PRIMARY KEY,
    Order_Date DATE,
    Customer_SID INT,
    Amount DECIMAL(10, 2) CHECK (Amount > 20000),
    FOREIGN KEY (Customer_SID) REFERENCES CUSTOMER(SID)
);

INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES
(101, '2023-01-10', 1, 25000),
(102, '2023-02-15', 2, 30000),
(103, '2023-03-20', 3, 27000),
(104, '2023-04-25', 4, 32000),
(105, '2023-05-30', 5, 29000);
```

ORDERS

Order_ID	Order_Date	Customer_SID	Amount
101	2023-01-10	1	25000
102	2023-02-15	2	30000
103	2023-03-20	3	27000
104	2023-04-25	4	32000
105	2023-05-30	5	29000

2. List Customer Details Along with the Order Amount .

```
SELECT CUSTOMER.SID, CUSTOMER.Last_Name, CUSTOMER.First_Name, ORDERS.Amount
FROM CUSTOMER
JOIN ORDERS ON CUSTOMER.SID = ORDERS.Customer_SID;
```

SID	Last_Name	First_Name	Amount
1	Smith	John	25000
2	Jones	Alex	30000
3	Roberts	Sarah	27000
4	Evans	James	32000
5	Stevens	Emma	29000

(NavyA)

3.List Customers Whose Names End with "s" :

```
SELECT * FROM CUSTOMER
WHERE Last_Name LIKE '%s';
```

SID	Last_Name	First_Name
2	Jones	Alex
3	Roberts	Sarah
4	Evans	James
5	Stevens	Emma

4. List Orders Where Amount is Between 21000 and 30000 :

```
SELECT * FROM ORDERS
WHERE Amount BETWEEN 21000 AND 30000;
```

Order_ID	Order_Date	Customer_SID	Amount
101	2023-01-10	1	25000
102	2023-02-15	2	30000
103	2023-03-20	3	27000
105	2023-05-30	5	29000

5. List the orders where amount is increased by 500 and replace with name “new amount”.

```
SELECT Order_ID, Amount + 500 AS "New Amount"
FROM ORDERS;
```

Order_ID	New Amount
101	25500
102	30500
103	27500
104	32500
105	29500

(navya)

6. Display the order_id and total amount of orders :

```
SELECT Order_ID, Amount AS Total_Amount
FROM ORDERS;
```

Order_ID	Total_Amount
101	25000
102	30000
103	27000
104	32000
105	29000

7. Calculate the total amount of orders that has more than 15000 .

```
SELECT SUM(Amount) AS Total_Amount
FROM ORDERS
WHERE Amount > 15000;
```

Total_Amount
143000

8. Display all the contents of s4 and s5 using union clause.

```
SELECT * FROM s4
UNION
SELECT * FROM s5;
```

9. Find out the intersection of s4 and s5 tables.

```
SELECT * FROM s4
INTERSECT
SELECT * FROM s5;
```

(Navya)

10. Display the names of s4 and s5 tables using left, right, inner and full join:

```
SELECT s4.*, s5.*
FROM s4
LEFT JOIN s5 ON s4.ID = s5.ID;
```

```
SELECT s4.*, s5.*
FROM s4
INNER JOIN s5 ON s4.ID = s5.ID;
```

```
SELECT s4.*, s5.*
FROM s4
FULL OUTER JOIN s5 ON s4.ID = s5.ID;
```

11. Find out the names of s4 which are distinct :

```
SELECT DISTINCT Name FROM s4;
```

12. Write a query to Grant access and modification rights to customer table to user :

```
GRANT SELECT, INSERT, UPDATE, DELETE ON CUSTOMER TO user;
```

13. Write a query to revoke access rights to customer table to user :

```
REVOKE SELECT, INSERT, UPDATE, DELETE ON CUSTOMER FROM user;
```

14. Write a query to take backup of a database:

```
BACKUP DATABASE dbname TO DISK = 'path_to_backup_file';
```

15. Write a query to restore a database :

```
RESTORE DATABASE dbname FROM DISK = 'path_to_backup_file';
```

(Navya)