MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES

FACULTY OF COMPUTER SCIENCE

Practical File

FOR

1 Semester

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM

Submitted By:-

Student Name: Parth bhatnagar

Roll No.:24/SCA/BCA/032

Branch: BCA

Section: 1A

SUBMITTED TO:- SHILPA

SCHOOL OF COMPUTER SCIENCE

Introduction to SQL

Introduction to SQL and Its Data Types

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)** or **NUMERIC**: 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**
    - o **BOOLEAN**: Stores true/false values.
5. **Binary Data Types**
    - o **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)**
    - o Used to define and manage database structures.
    - o 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 1

Q1: Create the following tables

Student:

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:			

Dept No

Dname	Number	2	constraint
Location	varchar	20	Primary key
	varchar	10	

1. Insert five records for each table.

```
create table Student1(StudentId integer(4) primary key,Student_name varchar(40),Address1
varchar(300),Gender varchar(15),Course varchar(8));
insert into Student1 values(101,"Anishka sharma","Sec81","F","BCA");
insert into Student1 values(102,"Arpit yadav","Sec8","M","BCA");
insert into Student1 values(103,"Sheetal tanwar","Sec86","F","BCA");
insert into Student1 values(104,"Chirag singh","Sec75","M","MCA");
insert into Student1 values(105,"Shashwat thakur","Sec15","M","BCA");
```

Student1

StudentId	Student_name	Address1	Gender	Course
101	Anishka sharma	Sec81	F	BCA
102	Arpit yadav	Sec8	M	BCA
103	Sheetal tanwar	Sec86	F	BCA
104	Chirag singh	Sec75	M	MCA
105	Shashwat thakur	Sec15	M	BCA

```
create table Course(DeptNo integer(2) primary key, Dname varchar(20),Location
varchar(10));
insert into Course values(1,"SCA","C block");
insert into Course values(2,"SCA","C block");
insert into Course values(3,"SCA","C block");
insert into Course values(4,"SCA","C block");
insert into Course values(5,"SCA","C block");
```

Course

DeptNo	Dname	Location
1	SCA	C block
2	SCA	C block
3	SCA	C block
4	SCA	C block
5	SCA	C block

2. List all information about all students from student table

```
SELECT * FROM Student1;
```

StudentId	Student_name	Address1	Gender	Course
101	Anishka sharma	Sec81	F	BCA
102	Arpit yadav	Sec8	M	BCA
103	Sheetal tanwar	Sec86	F	BCA
104	Chirag singh	Sec75	M	MCA
105	Shashwat thakur	Sec15	M	BCA

3. List all student numbers along with their Courses.

```
SELECT StudentId, Course FROM Student1;
```

StudentId	Course
101	BCA
102	BCA
103	BCA
104	MCA
105	BCA

4. List Course names and locations from the Course table

```
SELECT Dname, Location FROM Course;
```

Dname	Location
SCA	C block
SCA	C block
SCA	C block
SCA	C block
SCA	C block

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

```
SELECT * FROM Student1 WHERE Course = 'MCA';
```

StudentId	Student_name	Address1	Gender	Course
104	Chirag singh	Sec75	M	MCA

6. List the students details in ascending order of course

```
SELECT * FROM Student1 ORDER BY Course ASC;
```

StudentId	Student_name	Address1	Gender	Course
101	Anishka sharma	Sec81	F	BCA
102	Arpit yadav	Sec8	M	BCA
103	Sheetal tanwar	Sec86	F	BCA
105	Shashwat thakur	Sec15	M	BCA
104	Chirag singh	Sec75	M	MCA

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.

```
CREATE TABLE Department (
    DeptNo NUMBER(2) PRIMARY KEY,
    DeptName VARCHAR2(20)
);
```

Department

DeptNo	DeptName
empty	

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

```
CREATE TABLE Project (
    ProjectID NUMBER(4) NOT NULL,
    ProjectName VARCHAR2(30) NOT NULL,
    StartDate DATE NOT NULL
);
```

Project

ProjectID	ProjectName	StartDate
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

```
CREATE TABLE Faculty (
    FacultyID NUMBER(4) PRIMARY KEY,
    FacultyName VARCHAR2(40),
    Email VARCHAR2(50) UNIQUE
);
```

Faculty

FacultyID	FacultyName	Email
empty		

13. Display list of student ordered by course

```
SELECT * FROM Student1 ORDER BY Course;
```

StudentId	Student_name	Address1	Gender	Course
101	Anishka sharma	Sec81	F	BCA
102	Arpit yadav	Sec8	M	BCA
103	Sheetal tanwar	Sec86	F	BCA
105	Shashwat thakur	Sec15	M	BCA
104	Chirag singh	Sec75	M	MCA

14. Display alphabetically sorted list of students

```
SELECT * FROM Student1 ORDER BY Student_name ASC;
```

StudentId	Student_name	Address1	Gender	Course
101	Anishka sharma	Sec81	F	BCA
102	Arpit yadav	Sec8	M	BCA
104	Chirag singh	Sec75	M	MCA
105	Shashwat thakur	Sec15	M	BCA
103	Sheetal tanwar	Sec86	F	BCA

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

Input

```
CREATE TABLE CUSTOMER (
    SID INT PRIMARY KEY,
    Last_Name VARCHAR(50),
    First_Name VARCHAR(50)
);

INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES
(1, 'Smith', 'John'),
(2, 'Jones', 'Alex'),
(3, 'Roberts', 'Sarah'),
(4, 'Evans', 'James'),
(5, 'Stevens', 'Emma');
```

CUSTOMER

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

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

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

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;
```

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';
```