REPORT FILE:

SQL Queries - Minimum 5 sets using one table / two tables.

Set 1: Basic Database Concepts & Table Creation

1. Create a Database

- Create a new database named SchoolDB.
- Command: CREATE DATABASE SchoolDB

2. Create a Table

- Create a table Students with fields StudentID (primary key), Name (varchar), Age (int), Class (varchar), and DOB (date).

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT,
    Class VARCHAR(5),
    DOB DATE
);
```

3. Describe Table

- Display structure of Students table.
- Command: DESCRIBE Students;

Set 2: Data Insertion & Constraints

1. Insert Data with Constraints

- Insert a new student record ensuring Age is not null.

```
INSERT INTO Students (StudentID, Name, Age, Class, DOB)
VALUES (1, 'Amit', 15, '10A', '2009-05-10');
```

2.Adding a Unique Constraint

- Alter Students to make Name a unique attribute.
- Command :

```
ALTER TABLE Students ADD CONSTRAINT Unique_Name UNIQUE(Name);
```

3. Inserting Multiple Rows

- Insert two more records into Students.

```
INSERT INTO Students (StudentID, Name, Age, Class, DOB)
VALUES (2, 'Ravi', 14, '9B', '2010-07-23'),
       (3, 'Sita', 15, '10A', '2009-01-19');
```

Set 3: Data Retrieval & Filtering

1. Select with WHERE

- Retrieve students from class 10A.
- Command:

```
SELECT * FROM Students WHERE Class = '10A';
```

2. Using BETWEEN and IN

- Find students aged between 14 and 15.
- Command:

```
SELECT * FROM Students WHERE Age BETWEEN 14 AND 15;
```

3. Using BETWEEN and IN

- List students ordered by Name.
- Command:

```
SELECT * FROM Students ORDER BY Name;
```

Set 4: Aggregate Functions & Grouping

1. Count

- Count the total number of students.
- Command:

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

2. Average Age

- Calculate the average age of students.
- Command:

```
SELECT AVG(Age) AS Average_Age FROM Students;
```

3. Group By and Having

- Count students in each class, showing only if count > 1.

```
SELECT Class, COUNT(*) AS Num_Students
FROM Students
GROUP BY Class
HAVING COUNT(*) > 1;
```

Set 5: Table Modification & Deletion

1. Alter Table – Add Column

- Add Gender column to Students.
- Command:

```
ALTER TABLE Students ADD Gender CHAR(1);
```

2. Delete Record

- Delete a student by StudentID.
- Command:

```
DELETE FROM Students WHERE StudentID = 1;
```

3. Drop Table

- Drop the Students table.
- Command:

```
DROP TABLE Students;
```

Set 6: Joins with Two Tables

1. Create Second Table

- Create Classes with ClassID, ClassName, and Teacher.

```
CREATE TABLE Classes (
    ClassID INT PRIMARY KEY,
    ClassName VARCHAR(5),
    Teacher VARCHAR(50)
);
```

2. Insert Data into Classes

- Insert a few records into Classes.

```
INSERT INTO Classes (ClassID, ClassName, Teacher)
VALUES (1, '10A', 'Mr. Sharma'), (2, '9B', 'Ms. Gupta');
```

3. Join Query

- Retrieve students with their class teacher.

```
SELECT Students.Name, Classes.Teacher
FROM Students
INNER JOIN Classes ON Students.Class = Classes.ClassName;
```

Set 7: Advanced Filtering & Joins

1. Using IS NULL and IS NOT NULL

- Select students without Gender information.
- Command:

```
SELECT * FROM Students WHERE Gender IS NULL;
```

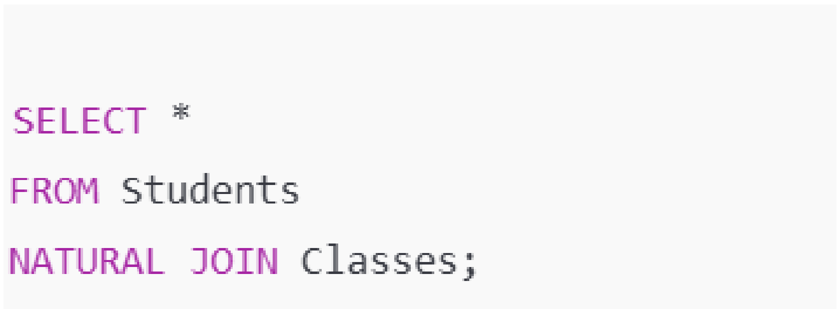
2. LIKE Clause

- Find students whose name starts with 'A'.
- Command:

```
SELECT * FROM Students WHERE Name LIKE 'A%';
```

3. Natural Join

- Display student and class data using a natural join.



```
SELECT *
FROM Students
NATURAL JOIN Classes;
```