Experiment No. 2 — SQL Joins, Subqueries & Views

Annotated SQL script (created from your MySQL Workbench screenshot).

Aim:

To understand and execute SQL queries using Joins, Sub-queries and Views.

Objective:

- To understand the use of SQL joins
- To understand the use of sub queries in SQL
- To learn creating and using views

Note:

SQL comments in the script use '--' for single-line comments. Copy the SQL blocks into MySQL Workbench and run step-by-step.

Annotated SQL Script

```
-- Step 1: Create Table student_info
-- This table stores basic student details. Columns:
-- Student_id : unique id (here stored as text)
-- Addmission_no : admission number used to join with fee table
-- First_name, Last_name, Age, City
CREATE TABLE student_info (
    Student_id VARCHAR(20),
    Addmission_no VARCHAR(50),
    First_name CHAR(50),
    Last_name CHAR(80),
    Age INT(10),
    City VARCHAR(50)
);


-- Step 2: Insert sample records into student_info
-- These are sample rows so you can test joins, subqueries and views.
INSERT INTO student_info VALUES
('1', 'ADM001', 'Rahul', 'Patil', 20, 'Pune'),
('2', 'ADM002', 'Sneha', 'Deshmukh', 19, 'Mumbai'),
('3', 'ADM003', 'Amit', 'Joshi', 21, 'Nashik'),
('4', 'ADM004', 'Priya', 'Kulkarni', 22, 'Nagpur'),
('5', 'ADM005', 'Neha', 'Shinde', 20, 'Satara'),
('6', 'ADM006', 'Riya', 'More', 19, 'Kolhapur'),
('7', 'ADM007', 'Akshay', 'Jadhav', 20, 'Solapur'),
('8', 'ADM008', 'Sagar', 'Naik', 20, 'Thane'),
('9', 'ADM009', 'Deepak', 'Kale', 21, 'Aurangabad'),
('10', 'ADM010', 'Komal', 'Pawar', 19, 'Nanded');


-- Step 3: View student_info contents
SELECT * FROM student_info;


-- Step 4: Create Table fee
-- This stores fee / course details keyed by Addmission_no
CREATE TABLE fee (
    Addmission_no VARCHAR(50),
    Course_name VARCHAR(30),
    Amount_paid FLOAT(10)
);


-- Step 5: Insert sample records into fee
INSERT INTO fee VALUES
('ADM001', 'BCA', 20000),
('ADM002', 'BBA', 25000),
('ADM003', 'BCA', 18000),
('ADM004', 'BSc', 15000),
('ADM005', 'BBA', 22000),
('ADM006', 'BCA', 19000),
('ADM007', 'BCom', 16000),
```

```sql
('ADM008', 'MBA', 30000),
('ADM009', 'MCA', 28000),
('ADM010', 'BSc', 14000);


-- Step 6: View fee contents
SELECT * FROM fee;


-- INNER JOIN example
-- Shows rows present in both student_info and fee (matching Addmission_no)
SELECT
    s.Student_id,
    s.First_name,
    s.Last_name,
    s.City,
    f.Course_name,
    f.Amount_paid
FROM student_info s
INNER JOIN fee f
ON s.Addmission_no = f.Addmission_no;


-- LEFT JOIN example
-- Shows all students; fee columns will be NULL if there is no fee record
SELECT
    s.Student_id,
    s.First_name,
    s.Last_name,
    f.Course_name,
    f.Amount_paid
FROM student_info s
LEFT JOIN fee f
ON s.Addmission_no = f.Addmission_no;


-- RIGHT JOIN example
-- Shows all fee records; student columns will be NULL if student missing
SELECT
    s.Student_id,
    s.First_name,
    s.Last_name,
    f.Course_name,
    f.Amount_paid
FROM student_info s
RIGHT JOIN fee f
ON s.Addmission_no = f.Addmission_no;


-- SUBQUERY example
-- Find students who paid more than the average fee amount
SELECT
    s.First_name,
    s.Last_name,
    f.Amount_paid
FROM student_info s
JOIN fee f
ON s.Addmission_no = f.Addmission_no
WHERE f.Amount_paid > (SELECT AVG(Amount_paid) FROM fee);


-- VIEW example - create a view combining student and fee info
CREATE VIEW student_fee_view AS
SELECT
    s.Student_id,
    s.First_name,
    s.Last_name,
    s.City,
    f.Course_name,
    f.Amount_paid
FROM student_info s
JOIN fee f
ON s.Addmission_no = f.Addmission_no;

-- Check the view
SELECT * FROM student_fee_view;


-- UPDATE (recreate) view to include Age column
CREATE OR REPLACE VIEW student_fee_view AS
SELECT
    s.Student_id,
    s.First_name,
    s.Last_name,
    s.Age,
    s.City,
    f.Course_name,
    f.Amount_paid
FROM student_info s
JOIN fee f
ON s.Addmission_no = f.Addmission_no;


-- DELETE example (delete a fee record)
DELETE FROM fee
WHERE Addmission_no = 'ADM010';
```