SCENARIO 1 — E-Commerce Orders & Customers

You work for a shopping platform. You need to analyze customers, orders, and which customers haven't ordered yet.

CREATE TABLES

```
CREATE TABLE customers (
 customer_id INT PRIMARY KEY,
 customer_name VARCHAR(50),
 city VARCHAR(50)
);

CREATE TABLE orders (
 order_id INT PRIMARY KEY,
 customer_id INT,
 order_date DATE,
 amount DECIMAL(10,2),
 FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

INSERT DATA

```
INSERT INTO customers VALUES
(1, 'Amit', 'Delhi'),
(2, 'Priya', 'Mumbai'),
(3, 'Rohit', 'Bangalore'),
(4, 'Neha', 'Delhi'),
(5, 'Kunal', 'Chennai');
```

```
INSERT INTO orders VALUES
(101, 1, '2025-01-10', 1200.50),
(102, 3, '2025-01-12', 450.00),
(103, 3, '2025-01-14', 780.25),
(104, 1, '2025-01-16', 999.00),
(105, 2, '2025-01-18', 650.00);
```

JOIN QUESTIONS

1. **List all customers with their orders (including customers with no orders).**
2. **Show customers who never placed any order.**
3. **Show total amount spent by each customer.**
4. **Get details of orders placed by customers from Delhi.**
5. **Show customer name, order date, and order amount — sorted by highest order value.**

SCENARIO 2 — Students, Courses & Enrollment

Context

A college wants to analyze course enrollment & students without courses.

CREATE TABLES

```
CREATE TABLE students (
  student_id INT PRIMARY KEY,
  student_name VARCHAR(50),
  branch VARCHAR(50)
);

CREATE TABLE courses (
  course_id INT PRIMARY KEY,
```

```
 course_name VARCHAR(50),
 credits INT
);

CREATE TABLE enrollment (
 student_id INT,
 course_id INT,
 semester VARCHAR(10),
 PRIMARY KEY(student_id, course_id),
 FOREIGN KEY(student_id) REFERENCES students(student_id),
 FOREIGN KEY(course_id) REFERENCES courses(course_id)
);
```

INSERT DATA

```
INSERT INTO students VALUES
(1, 'Dhruv', 'CSE'),
(2, 'Sneha', 'IT'),
(3, 'Karan', 'ECE'),
(4, 'Megha', 'CSE'),
(5, 'Arjun', 'MECH');

INSERT INTO courses VALUES
(101, 'DBMS', 4),
(102, 'OS', 3),
(103, 'Networks', 4),
(104, 'Python', 3);

INSERT INTO enrollment VALUES
```

(1, 101, 'Sem1'),

(1, 102, 'Sem1'),

(2, 103, 'Sem2'),

(3, 101, 'Sem1'),

(3, 104, 'Sem3');

JOIN QUESTIONS

1. **Show all students with enrolled courses. Include students not enrolled.**
2. **Find students who have not enrolled in any course.**
3. **Show course-wise student count.**
4. **List students and course names where credits > 3.**
5. **Show which students enrolled in more than 1 course.**

SCENARIO 3 — Employee, Department & Salary

Context

A company wants to analyze employee salaries, departments, and missing dept mapping.

CREATE TABLES

```
CREATE TABLE departments (
 dept_id INT PRIMARY KEY,
 dept_name VARCHAR(50)
);

CREATE TABLE employees (
 emp_id INT PRIMARY KEY,
 emp_name VARCHAR(50),
 dept_id INT,
```

```
  FOREIGN KEY(dept_id) REFERENCES departments(dept_id)
);

CREATE TABLE salaries (
  salary_id INT PRIMARY KEY,
  emp_id INT,
  salary_amount DECIMAL(10,2),
  month VARCHAR(20),
  FOREIGN KEY(emp_id) REFERENCES employees(emp_id)
);
```

INSERT DATA

```
INSERT INTO departments VALUES
(10, 'HR'),
(20, 'Finance'),
(30, 'IT'),
(40, 'Marketing');

INSERT INTO employees VALUES
(1, 'Anil', 30),
(2, 'Sakshi', 10),
(3, 'Vivek', 30),
(4, 'Manish', 20),
(5, 'Reena', NULL);

INSERT INTO salaries VALUES
(1, 1, 50000, 'January'),
(2, 2, 45000, 'January'),
```

(3, 3, 55000, 'January'),

(4, 1, 52000, 'February'),

(5, 4, 60000, 'January');

JOIN QUESTIONS

1. **Show all employees with their department names — include employees without a department.**
2. **Display employees who haven’t received any salary yet.**
3. **List department-wise total salary paid.**
4. **Find employees working in IT with salary > 50000.**
5. **Show employee name, dept name, month, salary in a single result.**