**1. Sales Performance Analysis**

**Objective:**

Analyze the sales performance of products, categories, and regions.

**Tables & Schema:**

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE,

phone VARCHAR(15),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Products (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

category VARCHAR(50),

price DECIMAL(10,2) NOT NULL,

stock INT CHECK (stock >= 0)

);

CREATE TABLE Orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

order\_date DATE NOT NULL,

total\_amount DECIMAL(10,2),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id) ON DELETE CASCADE

);

CREATE TABLE OrderDetails (

order\_detail\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

product\_id INT,

quantity INT CHECK (quantity > 0),

subtotal DECIMAL(10,2),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES Products(product\_id) ON DELETE CASCADE

);

**MySQL Queries:**

**1. Total Sales per Product**

SELECT p.name, SUM(od.quantity) AS total\_quantity, SUM(od.subtotal) AS total\_sales

FROM OrderDetails od

JOIN Products p ON od.product\_id = p.product\_id

GROUP BY p.name

ORDER BY total\_sales DESC;

**2. Monthly Sales Trend**

SELECT DATE\_FORMAT(order\_date, '%Y-%m') AS month, SUM(total\_amount) AS total\_sales

FROM Orders

GROUP BY month

ORDER BY month;

**3. Top 5 Customers by Spending**

SELECT c.name, SUM(o.total\_amount) AS total\_spent

FROM Orders o

JOIN Customers c ON o.customer\_id = c.customer\_id

GROUP BY c.name

ORDER BY total\_spent DESC

LIMIT 5;

**2. Employee Performance & HR Analytics**

**Objective:**

Analyze employee performance, attrition rate, and department-wise productivity.

**Tables & Schema:**

CREATE TABLE Employees (

employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

department VARCHAR(50),

salary DECIMAL(10,2) NOT NULL,

hire\_date DATE NOT NULL,

resignation\_date DATE NULL,

status ENUM('Active', 'Resigned') DEFAULT 'Active'

);

CREATE TABLE Performance (

performance\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT,

review\_date DATE NOT NULL,

rating INT CHECK (rating BETWEEN 1 AND 5),

comments TEXT,

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id) ON DELETE CASCADE

);

**MySQL Queries:**

**1. Employee Attrition Rate**

SELECT

(COUNT(CASE WHEN status = 'Resigned' THEN 1 END) / COUNT(\*)) \* 100 AS attrition\_rate

FROM Employees;

**2. Department-wise Average Salary**

SELECT department, AVG(salary) AS avg\_salary

FROM Employees

GROUP BY department;

**3. Top Performing Employees**

SELECT e.name, e.department, AVG(p.rating) AS avg\_rating

FROM Employees e

JOIN Performance p ON e.employee\_id = p.employee\_id

GROUP BY e.name, e.department

ORDER BY avg\_rating DESC

LIMIT 5;

**3. Hospital Patient & Doctor Analytics**

**Objective:**

Monitor patient visits, doctor workload, and hospital revenue.

**Tables & Schema:**

CREATE TABLE Doctors (

doctor\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

specialty VARCHAR(100),

experience\_years INT CHECK (experience\_years >= 0)

);

CREATE TABLE Patients (

patient\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

gender ENUM('Male', 'Female', 'Other'),

birth\_date DATE

);

CREATE TABLE Appointments (

appointment\_id INT PRIMARY KEY AUTO\_INCREMENT,

doctor\_id INT,

patient\_id INT,

appointment\_date DATE NOT NULL,

fee DECIMAL(10,2),

status ENUM('Completed', 'Cancelled', 'Pending') DEFAULT 'Pending',

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id) ON DELETE CASCADE,

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id) ON DELETE CASCADE

);

**MySQL Queries:**

**1. Total Revenue per Doctor**

SELECT d.name, SUM(a.fee) AS total\_revenue

FROM Appointments a

JOIN Doctors d ON a.doctor\_id = d.doctor\_id

WHERE a.status = 'Completed'

GROUP BY d.name

ORDER BY total\_revenue DESC;

**2. Most Visited Doctors**

SELECT d.name, COUNT(a.appointment\_id) AS total\_appointments

FROM Appointments a

JOIN Doctors d ON a.doctor\_id = d.doctor\_id

GROUP BY d.name

ORDER BY total\_appointments DESC

LIMIT 5;

**3. Age Distribution of Patients**

SELECT

CASE

WHEN TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 0 AND 18 THEN '0-18'

WHEN TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 19 AND 35 THEN '19-35'

WHEN TIMESTAMPDIFF(YEAR, birth\_date, CURDATE()) BETWEEN 36 AND 60 THEN '36-60'

ELSE '60+'

END AS age\_group,

COUNT(\*) AS total\_patients

FROM Patients

GROUP BY age\_group;

**4. E-commerce Website User Behavior Analysis**

**Objective:**

Analyze website traffic, user behavior, and product engagement.

**Tables & Schema:**

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

registration\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE UserActivity (

activity\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

activity\_type ENUM('Page Visit', 'Product Click', 'Add to Cart', 'Purchase') NOT NULL,

product\_id INT NULL,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

**MySQL Queries:**

**1. Most Popular Products Based on Clicks**

SELECT product\_id, COUNT(\*) AS total\_clicks

FROM UserActivity

WHERE activity\_type = 'Product Click'

GROUP BY product\_id

ORDER BY total\_clicks DESC

LIMIT 10;

**2. Conversion Rate (Add to Cart to Purchase)**

SELECT

(COUNT(CASE WHEN activity\_type = 'Purchase' THEN 1 END) \* 100.0) /

COUNT(CASE WHEN activity\_type = 'Add to Cart' THEN 1 END) AS conversion\_rate

FROM UserActivity;

**3. Daily Active Users**

SELECT DATE(timestamp) AS activity\_date, COUNT(DISTINCT user\_id) AS active\_users

FROM UserActivity

GROUP BY activity\_date

ORDER BY activity\_date DESC;

**5. Library Management & Book Borrowing Analytics**

**Objective:**

Track book borrowing trends, popular books, and overdue returns.

**Tables & Schema:**

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

author VARCHAR(100),

genre VARCHAR(50),

copies\_available INT CHECK (copies\_available >= 0)

);

CREATE TABLE Borrowers (

borrower\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE BorrowedBooks (

borrow\_id INT PRIMARY KEY AUTO\_INCREMENT,

book\_id INT,

borrower\_id INT,

borrow\_date DATE NOT NULL,

return\_date DATE NULL,

due\_date DATE NOT NULL,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id) ON DELETE CASCADE,

FOREIGN KEY (borrower\_id) REFERENCES Borrowers(borrower\_id) ON DELETE CASCADE

);

**MySQL Queries:**

**1. Most Borrowed Books**

SELECT b.title, COUNT(bb.borrow\_id) AS borrow\_count

FROM BorrowedBooks bb

JOIN Books b ON bb.book\_id = b.book\_id

GROUP BY b.title

ORDER BY borrow\_count DESC;