Objective:

Practice creating and dropping database views on complex data sets that span across multiple related tables. These views should help simplify data retrieval from multiple joined tables, apply filters, and perform aggregations for meaningful insights. The focus is on understanding how to create, modify, and delete views effectively

Simple views:

Problem statement: The HR department needs access to employee contact information, such as name, email, and phone number, but should not see sensitive information like salary or personal identification numbers.

Query: Create a simple view that includes only the fields HR staff needs, hiding sensitive columns.

```
mysql> CREATE VIEW EmployeeContact AS
    -> SELECT name, email, phone_number
    -> FROM Employee;
Query OK, 0 rows affected (0.03 sec)

mysql> Select * from EmployeeContactrr
    -> ;
ERROR 1146 (42S02): Table 'week9.employeecontactrr' doesn't exist
mysql> Select * from EmployeeContact;
+--------------+-------------------------+--------------+
| name         | email                   | phone_number |
+--------------+-------------------------+--------------+
| Ravi Kumar   | ravi.kumar@example.com   | 9876543210   |
| Asha Singh   | asha.singh@example.com   | 9123456789   |
| Vikram Mehta | vikram.mehta@example.com | 9998887776   |
+--------------+-------------------------+--------------+
3 rows in set (0.01 sec)
```

scenario 2: Creating a Custom View for a Specific Department

Problem statement: The Sales department only needs access to information about customers, specifically their IDs, names, and cities. The rest of the data (like customer credit scores or billing addresses) is irrelevant for their daily tasks.

Problem statement: Create a view with only the necessary columns, focusing on the data that the Sales team needs.

```
mysql> CREATE VIEW SalesCustomerView AS
    -> SELECT customer_id, name, city
    -> FROM Customers;
Query OK, 0 rows affected (0.02 sec)

mysql> Select * from SalesCustomerView;
+-------------+-------------+-----------+
| customer_id | name        | city      |
+-------------+-------------+-----------+
|           1 | Neha Sharma | Delhi     |
|           2 | Rahul Verma | Mumbai    |
|           3 | Pooja Nair  | Bangalore |
+-------------+-------------+-----------+
3 rows in set (0.00 sec)
```

Scenario 3: Standardized View of Product Information

Problem statement: The Product Management team needs a consistent view of product details to analyse offerings. They only need the product name, description, and category, but not inventory or supplier details.

Query: Create a simple view that presents product details in a standardized format.

```
mysql> CREATE VIEW ProductDetailsView AS
    -> SELECT product_name, description, category
    -> FROM Products;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from ProductDetailsView;
+--------------+----------------+-------------+
| product_name | description    | category    |
+--------------+----------------+-------------+
| Laptop       | 15-inch screen | Electronics |
| Smartphone   | 64GB storage   | Electronics |
| Tablet       | 10-inch screen | Electronics |
+--------------+----------------+-------------+
3 rows in set (0.00 sec)
```

Scenario 4: Presenting Simplified Customer Orders for Customer Support

Problem statement: Customer Support agents need to see the latest orders placed by customers to assist with inquiries. However, they should only have access to customer names, order dates, and status—not financial details like order amounts or payment info.

Query: Create a view that shows only the essential fields for customer support tasks.

```
mysql> CREATE VIEW CustomerSupportOrders AS
    -> SELECT Customers.name, Orders.order_date, Orders.status
    -> FROM Customers
    -> JOIN Orders ON Customers.customer_id = Orders.customer_id;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from CustomerSupportOrders;
+-------------+------------+-----------+
| name        | order_date | status    |
+-------------+------------+-----------+
| Neha Sharma | 2024-11-01 | Delivered |
| Rahul Verma | 2024-11-05 | Pending   |
| Pooja Nair  | 2024-11-10 | Cancelled |
+-------------+------------+-----------+
3 rows in set (0.01 sec)

mysql>
```

Scenario 5: Creating a View for Recent Activity

Problem statement: A team wants a view of recent user activity from a `UserActivity` table, showing only the user ID and last login date for users who logged in within the last 30 days.

Query: Create a view that filters data to show only recent activity.

```
mysql> CREATE VIEW RecentUserActivity AS
    -> SELECT user_id, last_login_date
    -> FROM UserActivity
    -> WHERE last_login_date >= CURRENT_DATE - INTERVAL 30 DAY;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from RecentUserActivity;
+---------+-----------------+
| user_id | last_login_date |
+---------+-----------------+
|       1 | 2024-10-28      |
|       2 | 2024-11-20      |
|       3 | 2024-11-25      |
+---------+-----------------+
3 rows in set (0.01 sec)
```

Scenario 6: View for Product Pricing Information

Problem statement: A marketing analyst needs access to product pricing information, specifically the product name and price, but should not see inventory levels or internal cost details.

```
mysql> CREATE VIEW ProductPricing AS
    -> SELECT product_name, price
    -> FROM Products;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from ProductPricing;
+--------------+----------+
| product_name | price    |
+--------------+----------+
| Laptop       | 50000.00 |
| Smartphone   | 15000.00 |
| Tablet       | 20000.00 |
+--------------+----------+
3 rows in set (0.00 sec)
```

Scenario 7: Simplified Customer Profile View

Problem statement: The Customer Relations team needs quick access to customer profiles, focusing only on fields like `customer_id`, `name`, and `membership_level` to manage loyalty programs.

Query Approach: Create a view to show only essential profile fields.

```
mysql> CREATE VIEW CustomerProfile AS
    -> SELECT customer_id, name, membership_level
    -> FROM Customers;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from CustomerProfile;
+-------------+-------------+------------------+
| customer_id | name        | membership_level |
+-------------+-------------+------------------+
|           1 | Neha Sharma | Gold             |
|           2 | Rahul Verma | Silver           |
|           3 | Pooja Nair  | Bronze           |
+-------------+-------------+------------------+
3 rows in set (0.00 sec)
```

Scenario 8: Summarized Employee Directory

Scenario: A general employee directory is needed to display employee names and departments to all users without showing sensitive personal details.

Query: Create a view with only the name and department fields from the employee data.

```
mysql> CREATE VIEW EmployeeDirectory AS
    -> SELECT name, department
    -> FROM Employee;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from EmployeeDirectory;
+--------------+------------+
| name         | department |
+--------------+------------+
| Ravi Kumar   | HR         |
| Asha Singh   | Sales      |
| Vikram Mehta | IT         |
+--------------+------------+
3 rows in set (0.00 sec)
```

Complex views:

Scenario 1: Frequently Used Boats

Scenario: The marina management wants to see which boats are most frequently used, including the total number of reservations per boat**, the Frequently Used Boats contains the tables: Sailors**, **Boats**, **Reservations**:

Query: Create a view that joins Boats and Reservations, counting reservations grouped by boat_id.

```
mysql> CREATE VIEW FrequentlyUsedBoats AS
    -> SELECT Boats.boat_id, Boats.boat_name, COUNT(Reservations.reservation_id) AS total_reservations
    -> FROM Boats
    -> JOIN Reservations ON Boats.boat_id = Reservations.boat_id
    -> GROUP BY Boats.boat_id, Boats.boat_name;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from FrequentlyUsedBoats;
+---------+--------------+--------------------+
| boat_id | boat_name    | total_reservations |
+---------+--------------+--------------------+
|       1 | Seafarer     |                  2 |
|       2 | Ocean Breeze |                  1 |
|       3 | Wave Rider   |                  1 |
+---------+--------------+--------------------+
3 rows in set (0.01 sec)
```

Scenario: E-commerce Database

You are a database administrator for an e-commerce platform. The database consists of the following tables

1. Products

2.Categories

3. Orders

4. Order Details

5. Customers

Question 1:

Create a view `ProductSalesSummary` to show the total sales revenue for each product, including product name, total quantity sold, and total revenue.

```
mysql> CREATE VIEW ProductSalesSummary AS
    -> SELECT
    ->     Products.product_name,
    ->     SUM(OrderDetails.quantity) AS total_quantity_sold,
    ->     SUM(OrderDetails.quantity * OrderDetails.price) AS total_revenue
    -> FROM Products
    -> JOIN OrderDetails ON Products.product_id = OrderDetails.product_id
    -> GROUP BY Products.product_name;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from ProductSalesSummary;
+--------------+---------------------+---------------+
| product_name | total_quantity_sold | total_revenue |
+--------------+---------------------+---------------+
| Laptop       |                   1 |      50000.00 |
| Smartphone   |                   2 |      60000.00 |
| Tablet       |                   1 |      20000.00 |
+--------------+---------------------+---------------+
3 rows in set (0.00 sec)
```

Question 2:

Write a query to fetch the top 3 products with the highest sales revenue from the `ProductSalesSummary` view.

```
mysql> SELECT *
    -> FROM ProductSalesSummary
    -> ORDER BY total_revenue DESC
    -> LIMIT 3;
+--------------+----------------------+---------------+
| product_name | total_quantity_sold | total_revenue |
+--------------+----------------------+---------------+
| Smartphone   |                    2 |      60000.00 |
| Laptop       |                    1 |      50000.00 |
| Tablet       |                    1 |      20000.00 |
+--------------+----------------------+---------------+
3 rows in set (0.00 sec)
```

Question 3:

You notice that the `Price` of a specific product (`ProductID = 101`) is incorrect. Update the price in the `Products` table via a view.

```
mysql> CREATE VIEW UpdateProductPrice AS
    -> SELECT product_id, product_name, price
    -> FROM Products;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> UPDATE UpdateProductPrice
    -> SET price = 99.99
    -> WHERE product_id = 101;
Query OK, 0 rows affected (0.01 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> select * from UpdateProductPrice;
+------------+--------------+----------+
| product_id | product_name | price    |
+------------+--------------+----------+
|          1 | Laptop       | 50000.00 |
|          2 | Smartphone   | 15000.00 |
|          3 | Tablet       | 20000.00 |
+------------+--------------+----------+
3 rows in set (0.00 sec)
```

Question 4:

Create a view `CustomerOrderSummary` to display each customer's name, their country, and the total amount spent across all orders.

```
mysql> CREATE VIEW CustomerOrderSummary AS
    -> SELECT
    ->     Customers.name,
    ->     Customers.country,
    ->     SUM(OrderDetails.quantity * OrderDetails.price) AS total_amount_spent
    -> FROM Customers
    -> JOIN Orders ON Customers.customer_id = Orders.customer_id
    -> JOIN OrderDetails ON Orders.order_id = OrderDetails.order_id
    -> GROUP BY Customers.name, Customers.country;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from CustomerOrderSummary;
+-------------+---------+--------------------+
| name        | country | total_amount_spent |
+-------------+---------+--------------------+
| Neha Sharma | India   |           50000.00 |
| Rahul Verma | India   |           60000.00 |
| Pooja Nair  | India   |           20000.00 |
+-------------+---------+--------------------+
3 rows in set (0.00 sec)
```

Question 5:

Create a view `LowStockProducts` to display products with stock below 50, including product name, category, and stock level.

```
mysql> CREATE VIEW LowStockProducts AS
    -> SELECT Products.product_name, Categories.category_name, Products.stock
    -> FROM Products
    -> JOIN Categories ON Products.category_id = Categories.category_id
    -> WHERE Products.stock < 50;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from LowStockProducts;
+--------------+---------------+-------+
| product_name | category_name | stock |
+--------------+---------------+-------+
| Laptop       | Electronics   |    25 |
| Tablet       | Electronics   |    10 |
+--------------+---------------+-------+
2 rows in set (0.00 sec)
```

Question 6:

Dealing with Non-Updatable Views

Why might the view `CustomerOrderSummary` not be updatable? Provide a solution to handle updates.?

```
mysql> CREATE VIEW OrderDetailsView AS SELECT OrderDetailID, OrderID, ProductID, Quantity F
ROM OrderDetails;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from OrderDetailsView;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|             1 |       1 |       101 |        1 |
|             2 |       1 |       103 |        2 |
|             3 |       2 |       102 |        1 |
|             4 |       3 |       104 |        3 |
+---------------+---------+-----------+----------+
4 rows in set (0.00 sec)

mysql> UPDATE OrderDetails SET Quantity = 5 WHERE OrderDetailID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from OrderDetailsView;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|             1 |       1 |       101 |        5 |
|             2 |       1 |       103 |        2 |
|             3 |       2 |       102 |        1 |
|             4 |       3 |       104 |        3 |
+---------------+---------+-----------+----------+
4 rows in set (0.00 sec)
```