```
-- Create & use the Database --
CREATE DATABASE ecommerce_analysis;
USE ecommerce_analysis;

-- Customer Table --
CREATE TABLE customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    city VARCHAR(50),
    country VARCHAR(50),
    signup_date DATE
);

-- Category Table --
CREATE TABLE categories (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    category_name VARCHAR(50) NOT NULL
);

-- Products Table --
CREATE TABLE products (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100) NOT NULL,
    category_id INT,
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
);

-- Orders Table --
CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATE NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    status VARCHAR(20) CHECK (status IN ('completed', 'cancelled',
'pending')),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

-- Orderitem Table --
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

-- Inserting customers Data --
INSERT INTO customers (name, email, city, country, signup_date) VALUES
('John Doe', 'john@example.com', 'New York', 'USA', '2022-01-15'),
('Alice Smith', 'alice@example.com', 'London', 'UK', '2022-03-10'),
('Bob Johnson', 'bob@example.com', 'Berlin', 'Germany', '2022-05-20'),
('Emma Brown', 'emma@example.com', 'Paris', 'France', '2022-07-05'),
('Michael Lee', 'michael@example.com', 'Tokyo', 'Japan', '2022-09-12');
```

```sql
-- Inserting the Category --
INSERT INTO categories (category_name) VALUES
('Electronics'), ('Clothing'), ('Books'), ('Home & Kitchen');

-- Inserting the Products Data --
INSERT INTO products (product_name, category_id, price) VALUES
('Laptop', 1, 999.99),
('Smartphone', 1, 699.99),
('T-Shirt', 2, 19.99),
('Jeans', 2, 49.99),
('Novel', 3, 12.99),
('Cookware Set', 4, 89.99);

-- Inserting the Orders Data --
INSERT INTO orders (customer_id, order_date, total_amount, status)
VALUES
(1, '2023-01-10', 1019.98, 'completed'),
(2, '2023-01-15', 69.98, 'completed'),
(3, '2023-02-05', 129.97, 'completed'),
(1, '2023-03-20', 49.99, 'completed'),
(4, '2023-04-12', 89.99, 'completed'),
(5, '2023-05-18', 699.99, 'completed'),
(2, '2023-06-22', 19.99, 'completed'),
(3, '2023-07-30', 999.99, 'completed'),
(1, '2023-08-05', 12.99, 'completed'),
(4, '2023-09-10', 49.99, 'completed');

-- Inserting the Oredreditem Data --
INSERT INTO order_items (order_id, product_id, quantity, price) VALUES
(1, 1, 1, 999.99), (1, 3, 1, 19.99),
(2, 4, 1, 49.99), (2, 5, 1, 12.99),
(3, 2, 1, 699.99), (3, 6, 1, 89.99),
(4, 4, 1, 49.99), (5, 6, 1, 89.99),
(6, 2, 1, 699.99), (7, 3, 1, 19.99),
(8, 1, 1, 999.99), (9, 5, 1, 12.99),
(10, 4, 1, 49.99);

/*Queries for Customer Analysis*/
-- Customer Purchase History --
SELECT c.customer_id, c.name, c.email,
    COUNT(o.order_id) AS total_orders,
    SUM(o.total_amount) AS total_spent,
    MAX(o.order_date) AS last_order_date
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
WHERE o.status = 'completed'
GROUP BY c.customer_id, c.name, c.email
ORDER BY total_spent DESC;

-- Customer Lifetime Value --
SELECT c.customer_id, c.name, c.signup_date,
    SUM(o.total_amount) AS lifetime_value,
    COUNT(o.order_id) AS order_count,
    ROUND(SUM(o.total_amount) / COUNT(o.order_id),2) AS avg_order_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.status = 'completed'
```

```
GROUP BY c.customer_id, c.name, c.signup_date
ORDER BY lifetime_value DESC;

--  Shopping Patterns & Seasonality --
SELECT
    EXTRACT(MONTH FROM o.order_date) AS month,
    EXTRACT(YEAR FROM o.order_date) AS year,
    COUNT(o.order_id) AS total_orders,
    SUM(o.total_amount) AS total_revenue,
    ROUND(AVG(o.total_amount),2) AS avg_order_value
FROM orders o
WHERE o.status = 'completed'
GROUP BY year, month
ORDER BY year, month;

/* Create & use of the View Concepts */
-- High-Value Customer Segmentation --
CREATE OR REPLACE VIEW customer_segmentation_view AS
SELECT c.customer_id,c.name,c.email,
    SUM(o.total_amount) AS total_spent,
    COUNT(o.order_id) AS order_count,
    DATEDIFF(CURRENT_DATE, c.signup_date) AS days_since_signup,
    SUM(o.total_amount) / DATEDIFF(CURRENT_DATE, c.signup_date) * 365
AS annual_spending,
    CASE
        WHEN SUM(o.total_amount) > 1000 THEN 'Platinum'
        WHEN SUM(o.total_amount) > 500 THEN 'Gold'
        WHEN SUM(o.total_amount) > 100 THEN 'Silver'
        ELSE 'Bronze'
    END AS customer_segment
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.status = 'completed'
GROUP BY c.customer_id, c.name, c.email, c.signup_date;

-- Result of the view --
SELECT c.customer_id,c.name,cat.category_name,
    COUNT(oi.order_item_id) AS items_purchased,
    SUM(oi.price * oi.quantity) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
JOIN categories cat ON p.category_id = cat.category_id
WHERE o.status = 'completed'
GROUP BY c.customer_id, c.name, cat.category_name
ORDER BY c.customer_id, total_spent DESC;

-- RFM Analysis View --
CREATE OR REPLACE VIEW customer_rfm_view AS
SELECT
customer_id,name,last_order_date,recency,frequency,monetary,r_score,f_s
core,m_score,
    CONCAT(r_score, f_score, m_score) AS rfm_cell,
    CASE
        WHEN recency <= 30 THEN 'Active'
        WHEN recency <= 90 THEN 'Warming'
        WHEN recency <= 180 THEN 'Cooling'
```

```
        ELSE 'Dormant'
    END AS customer_status
FROM (
    SELECT c.customer_id,c.name,
        MAX(o.order_date) AS last_order_date,
        COUNT(o.order_id) AS frequency,
        SUM(o.total_amount) AS monetary,
        DATEDIFF(CURRENT_DATE, MAX(o.order_date)) AS recency,
        NTILE(5) OVER (ORDER BY DATEDIFF(CURRENT_DATE,
MAX(o.order_date)) DESC) AS r_score,
        NTILE(5) OVER (ORDER BY COUNT(o.order_id)) AS f_score,
        NTILE(5) OVER (ORDER BY SUM(o.total_amount)) AS m_score
    FROM customers c
    JOIN orders o ON c.customer_id = o.customer_id
    WHERE o.status = 'completed'
    GROUP BY c.customer_id, c.name
) AS rfm_data;

-- Result of the View --
SELECT * FROM customer_rfm_view
WHERE rfm_cell IN ('555', '554', '545')
ORDER BY monetary DESC;
SELECT rfm_cell,customer_status,
    COUNT(*) AS customer_count,
    ROUND(AVG(monetary),2) AS avg_spend
FROM customer_rfm_view
GROUP BY rfm_cell, customer_status
ORDER BY rfm_cell;
```