SQL Code

```sql
CREATE database RetailDB_3;
USE RetailDB_3;
-- ====================================
-- CREATE DATABASE
-- ====================================
CREATE DATABASE IF NOT EXISTS RetailDB_3;
USE RetailDB_3;

-- ====================================
-- 1. Customerq
-- ====================================
CREATE TABLE Customerq (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    city VARCHAR(50),
    signup_date DATE
);

-- ====================================
-- 2. Suppliersq
-- ====================================
CREATE TABLE Suppliersq (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(100),
    contact_email VARCHAR(100),
    city VARCHAR(50)
);

-- ====================================
-- 3. Shippersq
-- ====================================
CREATE TABLE Shippersq (
    shipper_id INT AUTO_INCREMENT PRIMARY KEY,
    shipper_name VARCHAR(100),
    contact VARCHAR(100)
);

-- ====================================
-- 4. Payment_Methodsq
-- ====================================
CREATE TABLE Payment_Methodsq (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    payment_type VARCHAR(50) UNIQUE
);
DROP TABLE Productsq;
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS Productsq;

-- ====================================
-- 5. Productsq
-- ====================================
```

```sql
CREATE TABLE Productsq (
    product_id INT,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10,2),
    stock_qty INT,
    supplier_id INT
);

-- ====================================
-- 6. Orderq
-- ====================================
CREATE TABLE Orderq (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    payment_id INT,
    shipper_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customerq(customer_id),
    FOREIGN KEY (payment_id) REFERENCES Payment_Methodsq(payment_id),
    FOREIGN KEY (shipper_id) REFERENCES Shippersq(shipper_id)
);
SET FOREIGN_KEY_CHECKS = 1;
-- ====================================
-- 7. Order_Itemq
-- ====================================
CREATE TABLE Order_Itemq (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price_each DECIMAL(10,2)
);


-- ========================
-- Q1 Total revenue by shipper
-- ========================
SELECT sh.shipper_name,
       SUM(oi.quantity * oi.price_each) AS total_revenue
FROM Orderq o
JOIN Shippersq sh ON o.shipper_id = sh.shipper_id
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY sh.shipper_name;

-- ========================
-- Q2 Top 5 highest spending customers
-- ========================
SELECT c.name,
       SUM(oi.quantity * oi.price_each) AS total_spent
FROM Customerq c
JOIN Orderq o ON c.customer_id = o.customer_id
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY c.name
ORDER BY total_spent DESC
```

```sql
LIMIT 5;

-- ========================
-- Q3 Categories avg price > 8000
-- ========================
SELECT category, AVG(price) AS avg_price
FROM Productsq
GROUP BY category
HAVING AVG(price) > 8000;

-- ========================
-- Q4 Orders per city
-- ========================
SELECT c.city, COUNT(o.order_id) AS total_orders
FROM Customerq c
JOIN Orderq o ON c.customer_id = o.customer_id
GROUP BY c.city
ORDER BY total_orders DESC;

-- ========================
-- Q5 Suppliers with >1 category
-- ========================
SELECT s.supplier_name
FROM Suppliersq s
JOIN Productsq p ON s.supplier_id = p.supplier_id
GROUP BY s.supplier_name
HAVING COUNT(DISTINCT p.category) > 1;

-- ========================
-- Q6 Items per order
-- ========================
SELECT o.order_id, COUNT(oi.order_item_id) AS item_count
FROM Orderq o
LEFT JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY o.order_id;

-- ========================
-- Q7 Customers spent > average
-- ========================
SELECT c.name
FROM Customerq c
JOIN Orderq o ON c.customer_id = o.customer_id
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY c.name
HAVING SUM(oi.quantity * oi.price_each) >
       (SELECT AVG(total_spent)
        FROM (
          SELECT SUM(quantity * price_each) AS total_spent
          FROM Order_Itemq
          GROUP BY order_id
        ) t);

-- ========================
-- Q8 Products > category avg price
-- ========================
```

```sql
SELECT *
FROM Productsq p
WHERE price >
      (SELECT AVG(price)
       FROM Productsq
       WHERE category = p.category);

-- ========================
-- Q9 Orders > 50000
-- ========================
SELECT DISTINCT c.name
FROM Customerq c
JOIN Orderq o ON c.customer_id = o.customer_id
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY c.name, o.order_id
HAVING SUM(oi.quantity * oi.price_each) > 50000;

-- ========================
-- Q10 Customers > avg orders
-- ========================
SELECT customer_id
FROM Orderq
GROUP BY customer_id
HAVING COUNT(order_id) >
      (SELECT AVG(cnt)
       FROM (
         SELECT COUNT(order_id) AS cnt
         FROM Orderq
         GROUP BY customer_id
       ) t);

-- ========================
-- Q11 Most expensive product
-- ========================
SELECT *
FROM Productsq
WHERE price = (SELECT MAX(price) FROM Productsq);

-- ========================
-- Q12 Rank customers by spending
-- ========================
SELECT c.name,
       SUM(oi.quantity * oi.price_each) AS total_spent,
       RANK() OVER (ORDER BY SUM(oi.quantity * oi.price_each) DESC) AS rank_no
FROM Customerq c
JOIN Orderq o ON c.customer_id = o.customer_id
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY c.name;

-- ========================
-- Q13 Cumulative sales by date
-- ========================
SELECT o.order_date,
       SUM(oi.quantity * oi.price_each) AS daily_sales,
       SUM(SUM(oi.quantity * oi.price_each))
```

```sql
       OVER (ORDER BY o.order_date) AS cumulative_sales
FROM Orderq o
JOIN Order_Itemq oi ON o.order_id = oi.order_id
GROUP BY o.order_date;

-- ========================
-- Q14 Order count + percentage
-- ========================
SELECT customer_id,
       COUNT(order_id) AS order_count,
       ROUND(
         COUNT(order_id) * 100.0 /
         SUM(COUNT(order_id)) OVER (), 2
       ) AS percentage_contribution
FROM Orderq
GROUP BY customer_id;

-- ========================
-- Q15 Most recent order per customer
-- ========================
SELECT *
FROM (
  SELECT o.*,
         ROW_NUMBER() OVER
         (PARTITION BY customer_id ORDER BY order_date DESC) rn
  FROM Orderq o
) t
WHERE rn = 1;

-- ========================
-- Q16 Product sales rank per category
-- ========================
SELECT p.product_name,
       p.category,
       SUM(oi.quantity) AS total_qty,
       RANK() OVER (
         PARTITION BY p.category
         ORDER BY SUM(oi.quantity) DESC
       ) AS rank_in_category
FROM Productsq p
JOIN Order_Itemq oi ON p.product_id = oi.product_id
GROUP BY p.product_name, p.category;

-- ========================
-- Q17 Price category CASE
-- ========================
SELECT product_name,
       CASE
         WHEN price > 60000 THEN 'High'
         WHEN price BETWEEN 10000 AND 60000 THEN 'Medium'
         ELSE 'Low'
       END AS price_category
FROM Productsq;

-- ========================
```

```sql
-- Q18 Top 3 customers (CTE)
-- ========================
WITH cust_spend AS (
  SELECT c.name,
         SUM(oi.quantity * oi.price_each) AS total_spent
  FROM Customerq c
  JOIN Orderq o ON c.customer_id = o.customer_id
  JOIN Order_Itemq oi ON o.order_id = oi.order_id
  GROUP BY c.name
)
SELECT *
FROM cust_spend
ORDER BY total_spent DESC
LIMIT 3;

-- ========================
-- Q19 Customer loyalty
-- ========================
WITH order_count AS (
  SELECT customer_id, COUNT(order_id) AS cnt
  FROM Orderq
  GROUP BY customer_id
)
SELECT customer_id,
       CASE
         WHEN cnt >= 5 THEN 'High Loyalty'
         WHEN cnt >= 2 THEN 'Medium Loyalty'
         ELSE 'Low Loyalty'
       END AS loyalty_status
FROM order_count;

-- ========================
-- Q20 Monthly growth %
-- ========================
WITH monthly_sales AS (
  SELECT DATE_FORMAT(order_date, '%Y-%m') AS month,
         SUM(oi.quantity * oi.price_each) AS revenue
  FROM Orderq o
  JOIN Order_Itemq oi ON o.order_id = oi.order_id
  GROUP BY month
)
SELECT month,
       revenue,
       ROUND(
         (revenue - LAG(revenue) OVER (ORDER BY month)) * 100 /
         LAG(revenue) OVER (ORDER BY month), 2
       ) AS growth_percent
FROM monthly_sales;

-- ========================
-- Q21 Top 2 customers per city
-- ========================
WITH city_sales AS (
  SELECT c.city, c.name,
         SUM(oi.quantity * oi.price_each) AS total_spent
```

```
    FROM Customerq c
    JOIN Orderq o ON c.customer_id = o.customer_id
    JOIN Order_Itemq oi ON o.order_id = oi.order_id
    GROUP BY c.city, c.name
  )
  SELECT *
  FROM (
    SELECT *,
           RANK() OVER (PARTITION BY city ORDER BY total_spent DESC) rnk
    FROM city_sales
  ) t
  WHERE rnk <= 2;

  -- =========================
  -- Q22 Top 3 cities by revenue + shipper
  -- =========================
  SELECT c.city, sh.shipper_name,
         SUM(oi.quantity * oi.price_each) AS revenue
  FROM Customerq c
  JOIN Orderq o ON c.customer_id = o.customer_id
  JOIN Shippersq sh ON o.shipper_id = sh.shipper_id
  JOIN Order_Itemq oi ON o.order_id = oi.order_id
  GROUP BY c.city, sh.shipper_name
  ORDER BY revenue DESC
  LIMIT 3;

  -- =========================
  -- Q23 Order full details
  -- =========================
  SELECT o.order_id, c.name, p.product_name,
         s.supplier_name, sh.shipper_name
  FROM Orderq o
  JOIN Customerq c ON o.customer_id = c.customer_id
  JOIN Order_Itemq oi ON o.order_id = oi.order_id
  JOIN Productsq p ON oi.product_id = p.product_id
  JOIN Suppliersq s ON p.supplier_id = s.supplier_id
  JOIN Shippersq sh ON o.shipper_id = sh.shipper_id;

  -- =========================
  -- Q24 Sales per supplier
  -- =========================
  SELECT s.supplier_name,
         SUM(oi.quantity * oi.price_each) AS total_sales,
         AVG(oi.quantity * oi.price_each) AS avg_order_value
  FROM Suppliersq s
  JOIN Productsq p ON s.supplier_id = p.supplier_id
  JOIN Order_Itemq oi ON p.product_id = oi.product_id
  GROUP BY s.supplier_name;

  -- ================================================================================
================================================================================
  -- Q25 Categories >30% revenue
  -- =========================
  WITH cat_sales AS (
    SELECT p.category,
```

```
382           SUM(oi.quantity * oi.price_each) AS revenue
383     FROM Productsq p
384     JOIN Order_Itemq oi ON p.product_id = oi.product_id
385     GROUP BY p.category
386   ),
387   total_rev AS (
388     SELECT SUM(revenue) AS total FROM cat_sales
389   )
390   SELECT cs.category
391   FROM cat_sales cs, total_rev tr
392   WHERE cs.revenue > 0.3 * tr.total;
```