

SQL Code

```
1  CREATE DATABASE RETAILDB_2;
2  USE RETAILDB_2;
3  CREATE TABLE Customersss (
4      customer_id INT,
5      name VARCHAR(100),
6      email VARCHAR(100),
7      city VARCHAR(50),
8      signup_date DATE
9  );
10
11 CREATE TABLE Supplierss (
12     supplier_id INT,
13     supplier_name VARCHAR(100),
14     contact_email VARCHAR(100),
15     city VARCHAR(50)
16 );
17
18 CREATE TABLE Productss (
19     product_id INT,
20     product_name VARCHAR(100),
21     category VARCHAR(50),
22     price DECIMAL(10,2),
23     stock_qty INT,
24     supplier_id INT
25 );
26
27 CREATE TABLE Orderrss (
28     order_id INT,
29     customer_id INT,
30     order_date DATE,
31     total_amount DECIMAL(10,2),
32     payment_mode VARCHAR(50)
33 );
34
35 CREATE TABLE Order_Itemss (
36     order_item_id INT,
37     order_id INT,
38     product_id INT,
39     quantity INT,
40     price_each DECIMAL(10,2)
41 );
42 SELECT * FROM Customersss;
43 SELECT * FROM Supplierss;
44 SELECT * FROM Productss;
45 SELECT * FROM Orderrss;
46 SELECT * FROM Order_Itemss;
47
48 -- =====
=====
49 -- == QUERIES
50
51
```

```

52
53 -- Q1. Fetch all products along with their supplier name
54 SELECT p.product_name, s.supplier_name
55 FROM Productss p
56 INNER JOIN Suppliers s
57 ON p.supplier_id = s.supplier_id;
58
59 -- Q2. Find all customers and their orders (even if no order)
60 SELECT c.customer_id, c.name, o.order_id, o.order_date
61 FROM Customers c
62 LEFT JOIN Orders o
63 ON c.customer_id = o.customer_id;
64
65 -- Q3. Get all suppliers and the products they supply (even if no product)
66 SELECT s.supplier_id, s.supplier_name, p.product_name
67 FROM Productss p
68 RIGHT JOIN Suppliers s
69 ON p.supplier_id = s.supplier_id;
70
71 -- Q4. Show all customers and all orders (FULL OUTER JOIN simulation)
72 SELECT c.customer_id, c.name, o.order_id
73 FROM Customers c
74 LEFT JOIN Orders o
75 ON c.customer_id = o.customer_id
76 UNION
77 SELECT c.customer_id, c.name, o.order_id
78 FROM Customers c
79 RIGHT JOIN Orders o
80 ON c.customer_id = o.customer_id;
81
82 -- Q5. Products priced between 5000 and 50000 supplied from Mumbai
83 SELECT p.product_name, p.price
84 FROM Productss p
85 JOIN Suppliers s
86 ON p.supplier_id = s.supplier_id
87 WHERE p.price BETWEEN 5000 AND 50000
88 AND s.city = 'Mumbai';
89
90 -- Q6. Customers who placed more than 2 orders
91 SELECT c.customer_id, c.name, COUNT(o.order_id) AS total_orders
92 FROM Customers c
93 JOIN Orders o
94 ON c.customer_id = o.customer_id
95 GROUP BY c.customer_id, c.name
96 HAVING COUNT(o.order_id) > 2;
97
98 -- Q7. Each supplier's total sales value
99 SELECT s.supplier_id, s.supplier_name,
100        SUM(oi.quantity * oi.price_each) AS total_sales
101 FROM Suppliers s
102 JOIN Productss p ON s.supplier_id = p.supplier_id
103 JOIN Order_Items oi ON p.product_id = oi.product_id
104 GROUP BY s.supplier_id, s.supplier_name;
105
106 -- Q8. Average, highest, and lowest price of products in each category

```

```

107 SELECT category,
108         AVG(price) AS avg_price,
109         MAX(price) AS max_price,
110         MIN(price) AS min_price
111 FROM Productsss
112 GROUP BY category;
113
114 -- Q9. Top 5 customers by total spending
115 SELECT c.customer_id, c.name,
116        SUM(o.total_amount) AS total_spent
117 FROM Customerssss c
118 JOIN Orderrss o
119 ON c.customer_id = o.customer_id
120 GROUP BY c.customer_id, c.name
121 ORDER BY total_spent DESC
122 LIMIT 5;
123
124 -- Q10. Number of unique products ordered by each customer
125 SELECT c.customer_id, c.name,
126        COUNT(DISTINCT oi.product_id) AS unique_products
127 FROM Customerssss c
128 JOIN Orderrss o ON c.customer_id = o.customer_id
129 JOIN Order_Itemss oi ON o.order_id = oi.order_id
130 GROUP BY c.customer_id, c.name;
131
132 -- Q11. Customers who placed an order greater than average order amount
133 SELECT *
134 FROM Orderrss
135 WHERE total_amount >
136        (SELECT AVG(total_amount) FROM Orderrss);
137
138 -- Q12. Products that have never been ordered
139 SELECT product_name
140 FROM Productsss
141 WHERE product_id NOT IN
142        (SELECT product_id FROM Order_Itemss);
143
144 -- Q13. Customers who ordered at least one Electronics product
145 SELECT DISTINCT c.customer_id, c.name
146 FROM Customerssss c
147 JOIN Orderrss o ON c.customer_id = o.customer_id
148 JOIN Order_Itemss oi ON o.order_id = oi.order_id
149 JOIN Productsss p ON oi.product_id = p.product_id
150 WHERE p.category = 'Electronics';
151
152 -- Q14. Suppliers whose products have been ordered more than 100 times
153 SELECT s.supplier_id, s.supplier_name,
154        SUM(oi.quantity) AS total_quantity
155 FROM Suppliersss s
156 JOIN Productsss p ON s.supplier_id = p.supplier_id
157 JOIN Order_Itemss oi ON p.product_id = oi.product_id
158 GROUP BY s.supplier_id, s.supplier_name
159 HAVING SUM(oi.quantity) > 100;
160
161 -- Q15. Most expensive product(s)

```

```

162 SELECT product_name, price
163 FROM Productsss
164 WHERE price = (SELECT MAX(price) FROM Productsss);
165
166 -- Q16. Orders placed by customers from Mumbai, Delhi, or Bengaluru
167 SELECT o.*
168 FROM Orderrss o
169 JOIN Customersss c
170 ON o.customer_id = c.customer_id
171 WHERE c.city IN ('Mumbai', 'Delhi', 'Bengaluru');
172
173 -- Q17. Orders where payment mode is NOT UPI or Credit Card
174 SELECT *
175 FROM Orderrss
176 WHERE payment_mode NOT IN ('UPI', 'Credit Card');
177
178 -- Q18. Customers who have no email address
179 SELECT *
180 FROM Customersss
181 WHERE email IS NULL;
182
183 -- Q19. Suppliers not from the same city as any customer
184 SELECT *
185 FROM Supplierss
186 WHERE city NOT IN
187     (SELECT DISTINCT city FROM Customersss);
188
189 -- Q20. Latest 3 orders, skipping first 2
190 SELECT *
191 FROM Orderrss
192 ORDER BY order_date DESC
193 LIMIT 3 OFFSET 2;
194
195
196
197
198
199
200
201

```