

SQL Code

```
1  CREATE DATABASE ecommerce;
2
3
4  --Customers table
5  CREATE TABLE Customers (
6      CustomerID INT IDENTITY PRIMARY KEY,
7      Name VARCHAR(100),
8      Email VARCHAR(100),
9      Phone VARCHAR(15),
10     City VARCHAR(50),
11     JoinDate DATE
12 );
13
14
15  --Products table
16  CREATE TABLE Products (
17      ProductID INT IDENTITY PRIMARY KEY,
18      ProductName VARCHAR(100),
19      Category VARCHAR(50),
20      Price DECIMAL(10,2),
21      StockQuantity INT,
22      IsActive BIT
23 );
24
25
26  --Orders table
27  CREATE TABLE Orders (
28      OrderID INT IDENTITY PRIMARY KEY,
29      CustomerID INT,
30      OrderDate DATE,
31      OrderStatus VARCHAR(30),
32      FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
33 );
34
35
36  --OrderItems table
37  CREATE TABLE OrderItems (
38      OrderItemID INT IDENTITY PRIMARY KEY,
39      OrderID INT,
40      ProductID INT,
41      Quantity INT,
42      ItemPrice DECIMAL(10,2),
43      FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
44      FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
45 );
46
47
48  --Payments table
49  CREATE TABLE Payments (
50      PaymentID INT IDENTITY PRIMARY KEY,
51      OrderID INT,
```

```

52     PaymentDate DATE,
53     Amount DECIMAL(10,2),
54     PaymentMode VARCHAR(30),
55     PaymentStatus VARCHAR(30),
56     FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
57 );
58
59
60 INSERT INTO Customers (Name, Email, Phone, City, JoinDate)
61 VALUES
62 ('Rahul Sharma', 'rahul@gmail.com', '9876543210', 'Bangalore', '2024-01-1
0'),
63 ('Anita Rao', 'anita@gmail.com', '9876543211', 'Mysore', '2024-01-12'),
64 ('Suresh Kumar', 'suresh@gmail.com', '9876543212', 'Chennai', '2024-02-01'),
65 ('Priya Singh', 'priya@gmail.com', '9876543213', 'Delhi', '2024-02-05'),
66 ('Amit Verma', 'amit@gmail.com', '9876543214', 'Mumbai', '2024-02-10'),
67 ('Neha Patel', 'neha@gmail.com', '9876543215', 'Ahmedabad', '2024-02-15'),
68 ('Rohit Mehta', 'rohit@gmail.com', '9876543216', 'Pune', '2024-03-01'),
69 ('Kavya N', 'kavya@gmail.com', '9876543217', 'Bangalore', '2024-03-05'),
70 ('Manoj Das', 'manoj@gmail.com', '9876543218', 'Kolkata', '2024-03-10'),
71 ('Sneha Iyer', 'sneha@gmail.com', '9876543219', 'Chennai', '2024-03-15');
72
73
74 INSERT INTO Products (ProductName, Category, Price, StockQuantity, IsActive)
75 VALUES
76 ('Laptop', 'Electronics', 65000, 20, 1),
77 ('Smartphone', 'Electronics', 30000, 30, 1),
78 ('Headphones', 'Electronics', 2500, 50, 1),
79 ('Keyboard', 'Electronics', 1500, 40, 1),
80 ('Office Chair', 'Furniture', 8000, 15, 1),
81 ('Study Table', 'Furniture', 12000, 10, 1),
82 ('Mixer Grinder', 'Home Appliances', 4500, 25, 1),
83 ('Microwave Oven', 'Home Appliances', 15000, 12, 1),
84 ('Running Shoes', 'Fashion', 3500, 60, 1),
85 ('Wrist Watch', 'Fashion', 7000, 35, 1);
86
87
88 INSERT INTO Orders (CustomerID, OrderDate, OrderStatus)
89 VALUES
90 (1, '2024-04-01', 'Pending'),
91 (2, '2024-04-02', 'Delivered'),
92 (3, '2024-04-03', 'Pending'),
93 (4, '2024-04-04', 'Cancelled'),
94 (5, '2024-04-05', 'Delivered'),
95 (6, '2024-04-06', 'Pending'),
96 (7, '2024-04-07', 'Delivered'),
97 (8, '2024-04-08', 'Pending'),
98 (9, '2024-04-09', 'Delivered'),
99 (10, '2024-04-10', 'Pending');
100
101
102 INSERT INTO OrderItems (OrderID, ProductID, Quantity, ItemPrice)
103 VALUES
104 (1, 1, 1, 65000),
105 (2, 2, 1, 30000),

```

```

106    (3, 3, 2, 5000),
107    (4, 4, 1, 1500),
108    (5, 5, 1, 8000),
109    (6, 6, 1, 12000),
110    (7, 7, 2, 9000),
111    (8, 8, 1, 15000),
112    (9, 9, 2, 7000),
113    (10,10,1, 7000);
114
115
116    INSERT INTO Payments (OrderID, PaymentDate, Amount, PaymentMode, PaymentStat
us) VALUES
117    (1, '2024-04-01', 65000, 'UPI', 'Pending'),
118    (2, '2024-04-02', 30000, 'Credit Card', 'Success'),
119    (3, '2024-04-03', 5000, 'Debit Card', 'Failed'),
120    (4, '2024-04-04', 1500, 'UPI', 'Refunded'),
121    (5, '2024-04-05', 8000, 'Net Banking', 'Success'),
122    (6, '2024-04-06', 12000, 'UPI', 'Pending'),
123    (7, '2024-04-07', 9000, 'Credit Card', 'Success'),
124    (8, '2024-04-08', 15000, 'Debit Card', 'Pending'),
125    (9, '2024-04-09', 7000, 'UPI', 'Success'),
126    (10,'2024-04-10', 7000, 'Net Banking', 'Pending');
127
128
129    INSERT INTO Customers (Name, Email, Phone, City, JoinDate)
130    VALUES
131    ('Vikram Singh', 'vikram@gmail.com', '9876543220', 'Bangalore', '2024-03-2
0'),
132    ('Ananya Iyer', 'ananya@gmail.com', '9876543221', 'Hyderabad', '2024-03-2
2');
133
134
135    INSERT INTO Products (ProductName, Category, Price, StockQuantity, IsActive)
136    VALUES
137    ('Desk Lamp', 'Home Appliances', 2000, 25, 1),
138    ('Bluetooth Speaker', 'Electronics', 3500, 15, 1);
139
140
141    -- New Orders for Rahul (CustomerID=1)
142    INSERT INTO Orders (CustomerID, OrderDate, OrderStatus)
143    VALUES
144    (1, '2024-04-11', 'Delivered'),
145    (1, '2024-04-12', 'Delivered');
146
147    -- Add items in different categories
148    INSERT INTO OrderItems (OrderID, ProductID, Quantity, ItemPrice)
149    VALUES
150    (11, 5, 1, 8000), -- Furniture
151    (11, 7, 1, 4500), -- Home Appliances
152    (12, 9, 1, 3500); -- Fashion
153
154    -- Payments for new orders
155    INSERT INTO Payments (OrderID, PaymentDate, Amount, PaymentMode, PaymentStat
us) VALUES
156    VALUES

```

```

157 (11, '2024-04-11', 12500, 'UPI', 'Success'),
158 (12, '2024-04-12', 3500, 'Credit Card', 'Success');
159
160
161 SELECT * FROM Customers;
162 SELECT * FROM Products;
163 SELECT * FROM Orders;
164 SELECT * FROM OrderItems;
165 SELECT * FROM Payments;
166
167
168 --1. Retrieve all customers who have not placed any orders yet.
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211

```

--1. Retrieve all customers who have not placed any orders yet.

```

SELECT
    c.CustomerID,
    c.Name,
    o.OrderID
FROM Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL;

```

--or

```

SELECT
    CustomerID,
    Name
FROM Customers c
WHERE NOT EXISTS (SELECT 1 FROM Orders o WHERE c.CustomerID = o.CustomerID)

```

-- 2. Find the top 3 products by total sales amount

```

SELECT
    TOP 3
    p.ProductID,
    p.ProductName,
    SUM(oi.ItemPrice * oi.Quantity) AS TotalSales
FROM OrderItems oi
JOIN Products p
ON p.ProductID = oi.ProductID
GROUP BY p.ProductName, p.ProductID
ORDER BY TotalSales DESC;

```

--or

```

SELECT
    TOP 3
    p.ProductID,
    p.ProductName,
    (
        SELECT SUM(oi.ItemPrice * oi.Quantity)
        FROM OrderItems oi
        WHERE oi.ProductID = p.ProductID
    ) AS Totalales
FROM Products p
ORDER BY Totalales DESC;

```

```

212
213
214 --3. Retrieve orders with customer name, total order amount, and payment stat
us
215
216 SELECT
217     c.CustomerID,
218     c.Name,
219     SUM(oi.ItemPrice * oi.Quantity) AS OrderTotal,
220     p.PaymentStatus
221 FROM Customers c
222 JOIN Orders o
223     ON c.CustomerID = o.CustomerID
224 JOIN OrderItems oi
225     ON oi.OrderID = o.OrderID
226 JOIN Payments p
227     ON p.OrderID = oi.OrderID
228 WHERE p.PaymentStatus = 'Success'
229 GROUP BY
230     c.CustomerID,
231     c.Name,
232     p.PaymentStatus;
233
234
235     --OR
236 SELECT
237     c.CustomerID,
238     c.Name,
239     p.PaymentStatus,
240     (
241         SELECT SUM(oi.ItemPrice * oi.Quantity)
242         FROM OrderItems oi
243         WHERE oi.OrderID = o.OrderID
244     ) AS ItemTotal
245 FROM Customers c
246 JOIN Orders o
247 ON o.CustomerID = c.CustomerID
248 JOIN Payments p
249 ON p.OrderID = o.OrderID
250 WHERE p.PaymentStatus = 'Success';
251
252
253 --4. Calculate total revenue per month in 2024
254
255 SELECT
256     YEAR(p.PaymentDate) AS Year,
257     MONTH(p.PaymentDate) AS Month,
258     SUM(p.Amount) AS TotalRevenue
259 FROM Payments p
260 JOIN Orders o
261     ON o.OrderID = p.OrderID
262 WHERE YEAR(p.PaymentDate) = 2024
263 AND p.PaymentStatus = 'Success'
264 GROUP BY
265     YEAR(p.PaymentDate),

```

```

266     MONTH(p.PaymentDate);
267
268
269 --5. Find active products that have never been ordered
270
271 SELECT
272     p.ProductID,
273     p.ProductName,
274     oi.OrderID,
275     p.IsActive
276 FROM Products p
277 LEFT JOIN OrderItems oi
278 ON oi.ProductID = p.ProductID
279 WHERE oi.OrderID IS NULL AND p.IsActive = 1;
280
281 --or
282 SELECT
283     p.ProductID,
284     p.Productname,
285     p.IsActive
286 FROM Products p
287 WHERE p.IsActive = 1
288 AND NOT EXISTS ( Select 1
289             FROM OrderItems oi
290             WHERE oi.ProductID = p.ProductID
291         );
292
293
294 --6. Identify customer(s) who spent the most           ---TOP 1 WITH TIES
295 SELECT
296     TOP 1 WITH TIES
297     c.CustomerID,
298     c.Name,
299     SUM(p.Amount) As TotalSpent
300 FROM Customers c
301 JOIN Orders o
302 ON o.CustomerID = c.CustomerID
303 JOIN Payments p
304 ON p.OrderID = o.OrderID
305 GROUP BY
306     c.CustomerID,
307     c.Name
308 ORDER BY TotalSpent DESC;
309
310 --or Using CTE
311 WITH HighestSpentCustomers AS (
312     SELECT
313         c.CustomerID,
314         c.Name,
315         SUM(p.Amount) AS TotalSpent,
316         RANK() OVER(ORDER BY SUM(p.Amount) DESC) AS Rankk
317     FROM Customers c
318     JOIN Orders o
319     ON o.CustomerID = c.CustomerID
320     JOIN Payments p

```

```

321      ON p.OrderID = o.OrderID
322      GROUP BY c.CustomerID, c.Name
323  )
324  SELECT
325      CustomerID,
326      Name,
327      TotalSpent,
328      Rankk
329  FROM HighestSpentCustomers
330  WHERE Rankk = 1;
331
332
333  --* 7. Retrieve orders having at least one item priced above 10,000
334
335  --SELECT
336  --    o.OrderID,
337  --    o.CustomerID,
338  --    SUM(oi.ItemPrice * oi.Quantity) AS ItemPrice
339  --FROM Orders o
340  --JOIN OrderItems oi
341  --ON oi.OrderID = o.OrderID
342  --GROUP BY
343  --    o.CustomerID,
344  --    o.OrderID
345  --HAVING SUM(oi.ItemPrice * oi.Quantity) > 10000
346
347  --or
348  SELECT
349      o.OrderID,
350      o.CustomerID,
351      oi.ItemPrice
352  FROM Orders o
353  JOIN OrderItems oi
354  ON oi.OrderID = o.OrderID
355  WHERE oi.ItemPrice > 10000;
356
357  --or
358  SELECT
359      o.OrderID,
360      o.CustomerID
361  FROM Orders o
362  WHERE EXISTS (
363      SELECT 1,
364      oi.ItemPrice
365      FROM OrderItems oi
366      WHERE oi.OrderID = o.OrderID
367      AND oi.ItemPrice > 10000
368  );
369
370
371
372  --8. Show first order date and total orders per customer
373
374  SELECT
375      c.CustomerID,

```

```

376     MIN(o.OrderDate) AS FirstOrderDate, --3
377     COUNT(o.OrderID) AS NoOfOrders
378 FROM Orders o
379 JOIN Customers c           --1
380 ON c.CustomerID = o.CustomerID
381 GROUP BY                  --2
382     c.CustomerID
383 ORDER BY FirstOrderDate;   --4
384
385
386 --9. Find average quantity ordered per product category
387 SELECT
388     AVG(oi.Quantity) as AvgCount,
389     p.Category
390 FROM Products AS p
391 JOIN OrderItems AS oi
392 ON p.ProductID = oi.ProductID
393 GROUP BY p.Category;
394
395 select * from OrderItems
396
397
398 --10. List customers with number of pending payments
399 SELECT
400     c.CustomerID,
401     p.PaymentStatus,
402     COUNT(p.PaymentID) As NoOfPendingOrders
403 FROM Payments p
404 JOIN Orders oi
405 ON oi.OrderID = p.OrderID
406 JOIN Customers c
407 ON c.CustomerID = oi.CustomerID
408 GROUP BY
409     c.CustomerID,
410     p.PaymentStatus
411 HAVING p.PaymentStatus = 'Pending'
412
413 select * from Payments;
414
415 insert into Payments values(1, '2024-05-22', 300.00, 'CASH', 'Pending');
416
417 -- 11. Show cumulative sales per day
418 SELECT
419     p1.PaymentDate,
420     (
421         SELECT SUM(p2.Amount)
422         FROM Payments p2
423         WHERE p2.PaymentDate <= p1.PaymentDate
424         AND p2.PaymentStatus = 'Success'
425     ) AS CumulativeSales
426 FROM Payments p1
427 WHERE p1.PaymentStatus = 'Success'
428 GROUP BY p1.PaymentDate
429 ORDER BY p1.PaymentDate;
430

```

```

431
432    --12. Retrieve orders with total amount greater than 20,000
433    SELECT
434        oi.OrderID,
435        SUM(oi.ItemPrice * oi.Quantity) AS TotalOrderAmout
436    FROM OrderItems oi
437    GROUP BY oi.OrderID
438    HAVING SUM(oi.ItemPrice * oi.Quantity) > 20000;
439
440
441    --13. Find products with stock < 20 and total quantity sold > 5
442    SELECT
443        p.ProductID,
444        p.ProductName,
445        oi.Quantity,
446        SUM(oi.Quantity) AS TotalQuantitySold
447    FROM Products p
448    JOIN OrderItems oi
449        ON oi.ProductID = p.ProductID
450    WHERE p.StockQuantity < 20
451    GROUP BY p.ProductID,
452            p.ProductName,
453            oi.Quantity
454    HAVING SUM(oi.Quantity) > 5;
455
456    SELECT * FROM Orders;
457
458
459    --14. Retrieve last 2 orders per customer
460    WITH LastTwoOrders AS (
461        SELECT
462            c.CustomerID,
463            c.Name,
464            o.OrderDate,
465            COUNT(o.OrderID) As NoofOrders,
466            DENSE_RANK() OVER(
467                PARTITION BY c.CustomerID
468                ORDER BY o.OrderDate DESC
469            ) AS RecentOrders
470        FROM Orders o
471        JOIN Customers c
472            ON c.CustomerID = o.CustomerID
473        GROUP BY
474            c.CustomerID,
475            c.Name,
476            o.OrderDate
477    )
478    SELECT lo./*
479    FROM LastTwoOrders lo
480    WHERE lo.RecentOrders <=2;
481
482
483    --15. Find customers who ordered more than one product in the same order
484    SELECT
485        c.CustomerID,

```

```
486     c.Name,
487     o.OrderID,
488     COUNT(oi.ProductID) As NoOfProducts
489 FROM Orders o
490 JOIN OrderItems oi
491 ON oi.OrderID = o.OrderID
492 JOIN Customers c
493 ON c.CustomerID = o.CustomerID
494 GROUP BY
495     c.CustomerID,
496     c.Name,
497     o.OrderID
498 HAVING COUNT(oi.ProductID) > 1;
499
500 select * from Products;
501
502 --16. Retrieve orders with late payments (3-day rule)
503
504
505 --17. Calculate total revenue lost due to failed or refunded payments
506
507
508 --18. Find customers who ordered from more than 2 categories
509
510
511
512 --19. Increase price of active electronics products by 10%
513
514
515 --20. Delete cancelled orders with no successful payments
```