

# SQL Coding Evaluation

**Name:** Ruthravarshan S

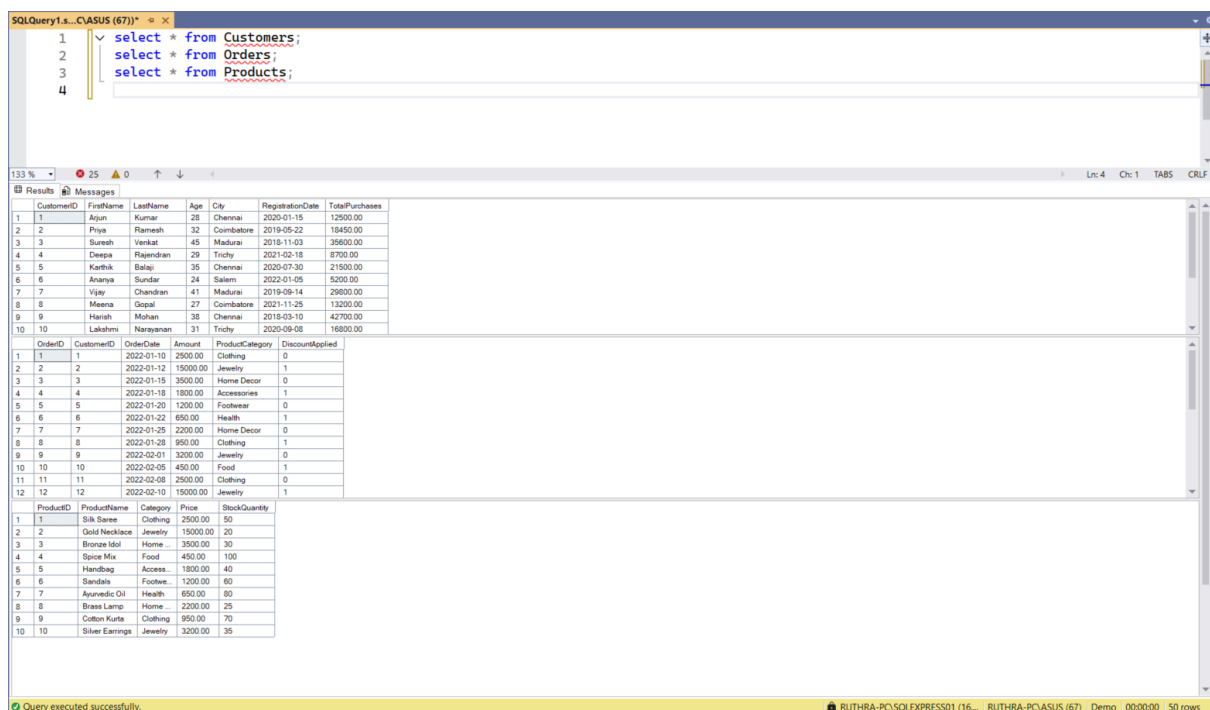
**Date:** 21-07-2025

## Objective

To demonstrate the ability to:

- Create and manipulate relational database tables
- Perform SQL operations involving JOIN, GROUP BY, WHERE, MIN, MAX, and aggregation functions
- Execute queries that extract summarized and filtered information

## Using the previously existing DB for the Evaluation



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the following SQL code:

```
1 select * from Customers;
2 select * from Orders;
3 select * from Products;
```

The bottom pane shows three result grids. The first grid displays customer information, the second grid displays order information, and the third grid displays product information.

CustomerID	FirstName	LastName	Age	City	RegistrationDate	TotalPurchases
1	Arun	Kumar	28	Chennai	2020-01-15	12000.00
2	Priya	Ramesh	32	Coimbatore	2019-05-22	18450.00
3	Suresh	Venkat	45	Madurai	2018-11-03	35600.00
4	Deepa	Rajendran	29	Trichy	2021-02-18	8700.00
5	Karthik	Bojay	35	Chennai	2020-07-30	21000.00
6	Ananya	Sundar	24	Salem	2022-01-05	5200.00
7	Visay	Chandran	41	Madurai	2019-08-14	29800.00
8	Meena	Gopal	27	Coimbatore	2021-11-25	13200.00
9	Heath	Mohan	38	Chennai	2018-03-10	42700.00
10	Lakshmi	Narayanan	31	Trichy	2020-09-08	16000.00

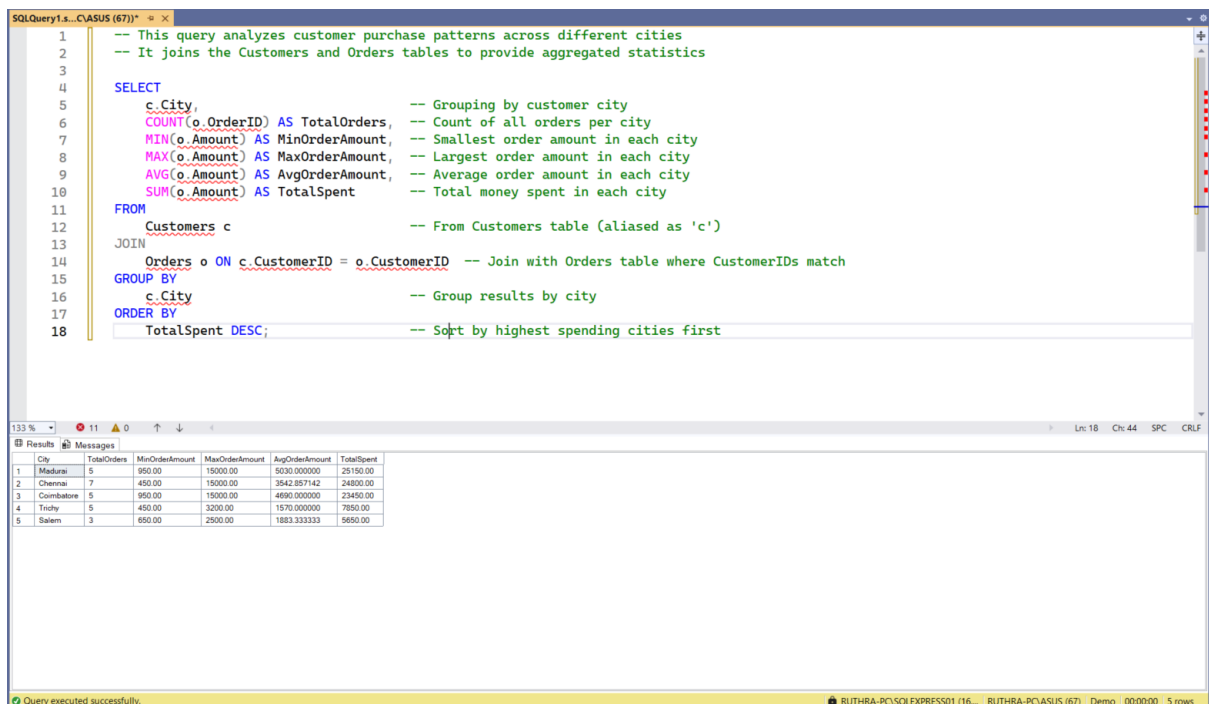
OrderID	CustomerID	OrderDate	Amount	ProductCategory	DiscountApplied
1	1	2022-01-10	2500.00	Clothing	0
2	2	2022-01-12	15000.00	Jewelry	1
3	3	2022-01-15	3500.00	Home Decor	0
4	4	2022-01-18	1800.00	Accessories	1
5	5	2022-01-20	1200.00	Footwear	0
6	6	2022-01-22	650.00	Health	1
7	7	2022-01-25	2200.00	Home Decor	0
8	8	2022-01-28	950.00	Clothing	1
9	9	2022-02-01	3200.00	Jewelry	0
10	10	2022-02-05	450.00	Food	1
11	11	2022-02-08	2500.00	Clothing	0
12	12	2022-02-10	15000.00	Jewelry	1

ProductID	ProductName	Category	Price	StockQuantity
1	Silk Saree	Clothing	2500.00	50
2	Gold Necklace	Jewelry	15000.00	20
3	Bronze Idol	Home	3500.00	30
4	Spice Mix	Food	450.00	100
5	Handbag	Access	1800.00	40
6	Sandals	Footwe	1200.00	60
7	Ayurvedic Oil	Health	650.00	80
8	Brass Lamp	Home	2200.00	25
9	Cotton Kurta	Clothing	950.00	70
10	Silver Earrings	Jewelry	3200.00	35

Each of these queries demonstrates different advanced SQL techniques:

- Query 1 shows basic aggregation with GROUP BY
- Query 2 adds filtering with WHERE and HAVING
- Query 3 includes a subquery in the HAVING clause
- Query 4 demonstrates conditional aggregation with CASE statements

## Query 1: Customer Purchase Statistics by City

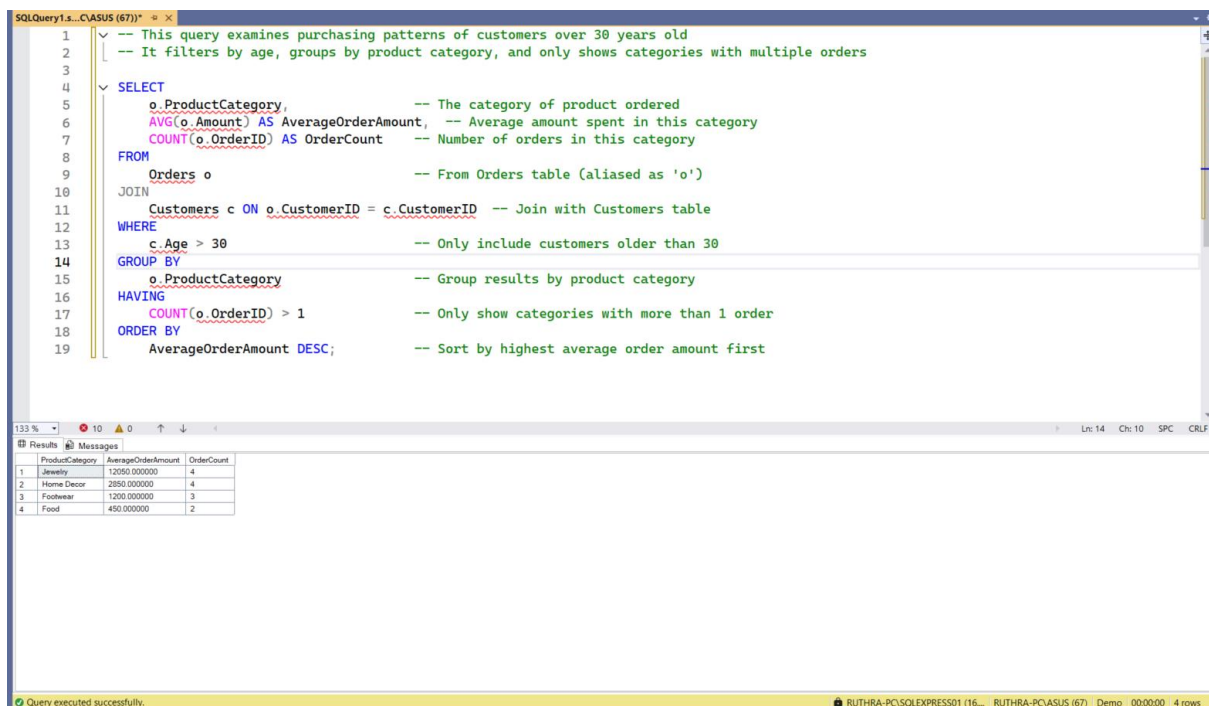


```
1  -- This query analyzes customer purchase patterns across different cities
2  -- It joins the Customers and Orders tables to provide aggregated statistics
3
4  SELECT
5      c.City, -- Grouping by customer city
6      COUNT(o.OrderID) AS TotalOrders, -- Count of all orders per city
7      MIN(o.Amount) AS MinOrderAmount, -- Smallest order amount in each city
8      MAX(o.Amount) AS MaxOrderAmount, -- Largest order amount in each city
9      AVG(o.Amount) AS AvgOrderAmount, -- Average order amount in each city
10     SUM(o.Amount) AS TotalSpent -- Total money spent in each city
11 FROM
12     Customers c -- From Customers table (aliased as 'c')
13 JOIN
14     Orders o ON c.CustomerID = o.CustomerID -- Join with Orders table where CustomerIDs match
15 GROUP BY
16     c.City -- Group results by city
17 ORDER BY
18     TotalSpent DESC; -- Sort by highest spending cities first
```

City	TotalOrders	MinOrderAmount	MaxOrderAmount	AvgOrderAmount	TotalSpent
1 Madurai	5	950.00	15000.00	5030.000000	25150.00
2 Chennai	7	450.00	15000.00	3542.857142	24800.00
3 Coimbatore	5	950.00	15000.00	4690.000000	23450.00
4 Trichy	5	450.00	3200.00	1570.000000	7850.00
5 Salem	3	650.00	2500.00	1883.333333	5650.00

Query executed successfully. RUTHRA-PC\SQLEXPRESS01 (16... RUTHRA-PC\ASUS (67) Demo 00:00:00 5 rows

## Query 2: Average Order Amount by Product Category for Customers Over 30



```
1  -- This query examines purchasing patterns of customers over 30 years old
2  -- It filters by age, groups by product category, and only shows categories with multiple orders
3
4  SELECT
5      o.ProductCategory, -- The category of product ordered
6      AVG(o.Amount) AS AverageOrderAmount, -- Average amount spent in this category
7      COUNT(o.OrderID) AS OrderCount -- Number of orders in this category
8 FROM
9     Orders o -- From Orders table (aliased as 'o')
10 JOIN
11     Customers c ON o.CustomerID = c.CustomerID -- Join with Customers table
12 WHERE
13     c.Age > 30 -- Only include customers older than 30
14 GROUP BY
15     o.ProductCategory -- Group results by product category
16 HAVING
17     COUNT(o.OrderID) > 1 -- Only show categories with more than 1 order
18 ORDER BY
19     AverageOrderAmount DESC; -- Sort by highest average order amount first
```

ProductCategory	AverageOrderAmount	OrderCount
1 Jewelry	12050.000000	4
2 Home Decor	2850.000000	4
3 Footwear	1200.000000	3
4 Food	490.000000	2

Query executed successfully. RUTHRA-PC\SQLEXPRESS01 (16... RUTHRA-PC\ASUS (67) Demo 00:00:00 4 rows

### Query 3: Customers Who Spent More Than Average in Each City

```
SQLQuery1.s...C:\ASUS (67)*
1  -- This query identifies high-spending customers in each city
2  -- It compares each customer's total spending against the overall average order amount
3
4  SELECT
5      c.City, -- Customer's city
6      c.FirstName + ' ' + c.LastName AS CustomerName, -- Full customer name
7      SUM(o.Amount) AS TotalSpent, -- Customer's total spending across all orders
8      (SELECT AVG(Amount) FROM Orders) AS OverallAvgOrderAmount -- Subquery to get overall average
9  FROM
10     Customers c -- From Customers table
11  JOIN
12     Orders o ON c.CustomerID = o.CustomerID -- Join with Orders table
13  GROUP BY
14     c.City, c.FirstName, c.LastName -- Group by city and customer name
15  HAVING
16     SUM(o.Amount) > (SELECT AVG(Amount) FROM Orders) -- Only show customers who spent more than average
17  ORDER BY
18     c.City, TotalSpent DESC; -- Sort by city, then by highest spenders first
```

City	CustomerName	TotalSpent	OverallAvgOrderAmount
Chennai	Divya Srinivasan	15000.00	3476.000000
Chennai	Harish Mohan	5000.00	3476.000000
Coimbatore	Pritya Ramesh	17200.00	3476.000000
Coimbatore	Meena Gopal	4450.00	3476.000000
Madurai	Vijay Chandran	17200.00	3476.000000
Madurai	Suresh Venkat	4450.00	3476.000000
Madurai	Ramesh Iyer	3500.00	3476.000000
Tiruchy	Deepa Rajendran	5000.00	3476.000000

Query executed successfully.

### Query 4: Discount Usage Analysis by Age Group and Product Category

```
SQLQuery1.s...C:\ASUS (67)*
1  -- This query analyzes how discounts are used across different age groups and product categories
2  -- It uses CASE statements to create age brackets and calculates discount percentages
3
4  SELECT
5      CASE -- Create age groups using CASE
6          WHEN c.Age BETWEEN 20 AND 29 THEN '20-29'
7          WHEN c.Age BETWEEN 30 AND 39 THEN '30-39'
8          WHEN c.Age >= 40 THEN '40+'
9      END AS AgeGroup, -- Resulting age group column
10     o.ProductCategory, -- Product category column
11     COUNT(CASE WHEN o.DiscountApplied = 1 THEN 1 END) AS DiscountedOrders, -- Count of orders with discount
12     COUNT(o.OrderID) AS TotalOrders, -- Total orders in this group
13     CAST(COUNT(CASE WHEN o.DiscountApplied = 1 THEN 1 END) AS FLOAT) /
14         COUNT(o.OrderID) * 100 AS DiscountPercentage -- Calculate percentage of orders with discount
15  FROM
16     Customers c -- From Customers table
17  JOIN
18     Orders o ON c.CustomerID = o.CustomerID -- Join with Orders table
19  GROUP BY
20     CASE -- Group by the same age groups
21         WHEN c.Age BETWEEN 20 AND 29 THEN '20-29'
22         WHEN c.Age BETWEEN 30 AND 39 THEN '30-39'
23         WHEN c.Age >= 40 THEN '40+'
24     END,
25     o.ProductCategory -- Also group by product category
26  ORDER BY
27     AgeGroup, DiscountPercentage DESC; -- Sort by age group, then by highest discount percentage
```

AgeGroup	ProductCategory	DiscountedOrders	TotalOrders	DiscountPercentage
20-29	Accessories	2	2	100
20-29	Health	2	2	100
20-29	Clothing	1	4	25
20-29	Home Decor	0	1	0
20-29	Jewelry	0	1	0
20-29	Food	2	2	100
30-39	Accessories	1	1	100
30-39	Jewelry	2	3	66.6666666667
30-39	Footwear	0	3	0
30-39	Home Decor	0	1	0
40+	Clothing	1	1	100
40+	Jewelry	1	1	100

Query executed successfully.