

**Requirement1:** the total amount spent and the country for the Pending delivery status for each country.

**Assumption:** Assuming that the customers having at least one shipping with Pending status is included in the reporting

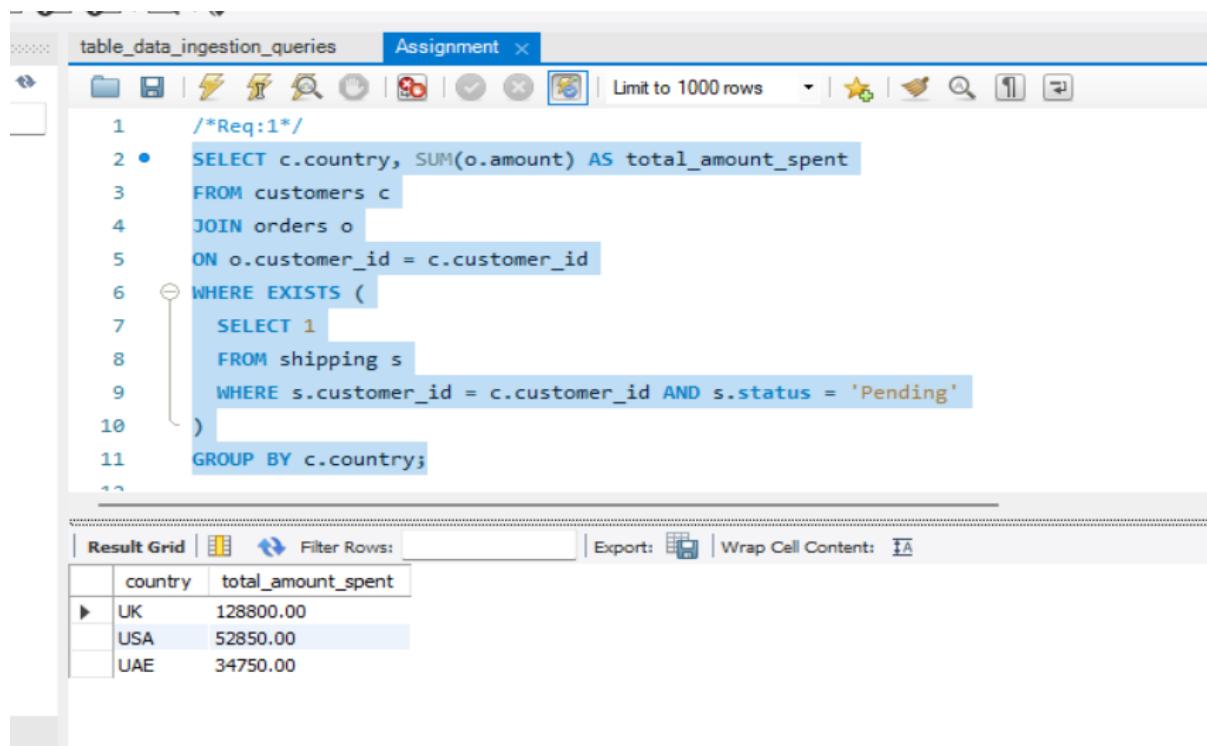
**Test Cases:**

Detailed test scenarios and test cases for this requirement are documented in the attached Excel file Anwita take home Assignment test cases.xlsx, Sheet: Req1.

**SQL query to test the Expected value:**

```
SELECT c.country, SUM(o.amount) AS total_amount_spent
FROM customers c
JOIN orders o
ON o.customer_id = c.customer_id
WHERE EXISTS (
    SELECT 1
    FROM shipping s
    WHERE s.customer_id = c.customer_id AND s.status = 'Pending'
)
GROUP BY c.country;
```

**Test result:**



The screenshot shows a database query editor window titled "Assignment". The query pane displays the SQL code for Requirement 1, which selects the total amount spent by country for customers with pending delivery status. The results pane shows a grid with three rows: UK (128800.00), USA (52850.00), and UAE (34750.00).

| country | total_amount_spent |
|---------|--------------------|
| UK      | 128800.00          |
| USA     | 52850.00           |
| UAE     | 34750.00           |

**Requirement Gap:**

This requirement does not clearly define whether Pending delivery status applies at order level or customer level. Since shipping status is available only at customer level and not mapped to Order\_ID, the calculation includes all orders of a customer when at least one Pending delivery record exists. This may not reflect the accurate amounts for pending order.

**Requirement2: the total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details.**

**Assumptions:**

Assuming the report is validated at customer and product level, i.e. one row per customer per product

Assuming each order row represents one unit sold.

Transaction is not specifically defined in the data tables so it is being counted using distinct Order IDs.

**Test Cases:**

*Detailed test scenarios and test cases for this requirement are documented in the attached Excel file [Anwita take home Assignment test cases.xlsx, Sheet: Req2.](#)*

**SQL query to test the Expected value:**

```
SELECT c.Customer_ID, o.Item,
       COUNT(DISTINCT o.Order_ID) AS total_transactions,
       COUNT(*) AS total_quantity_sold,
       SUM(o.Amount) AS total_amount_spent
  FROM Customers c
 JOIN Orders o
    ON o.Customer_ID = c.Customer_ID
 GROUP BY c.Customer_ID, o.Item;
```

**Test result:**

table\_data\_ingestion\_queries Assignment

```

12
13  /*Req:2*/
14 •   SELECT c.Customer_ID, o.Item,
15      COUNT(DISTINCT o.Order_ID) AS total_transactions,
16      COUNT(*) AS total_quantity_sold,
17      SUM(o.Amount) AS total_amount_spent
18  FROM Customers c
19  JOIN Orders o
20  ON o.Customer_ID = c.Customer_ID
21  GROUP BY c.Customer_ID, o.Item;
22

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

|   | Customer_ID | Item     | total_transactions | total_quantity_sold | total_amount_spent |
|---|-------------|----------|--------------------|---------------------|--------------------|
| ▶ | 4           | Mousepad | 1                  | 1                   | 200.00             |
|   | 5           | DDR RAM  | 1                  | 1                   | 1500.00            |
|   | 8           | DDR RAM  | 1                  | 1                   | 1500.00            |
|   | 8           | Mousepad | 2                  | 2                   | 450.00             |
|   | 8           | Webcam   | 1                  | 1                   | 350.00             |
|   | 10          | Keyboard | 1                  | 1                   | 400.00             |
|   | 12          | Harddisk | 1                  | 1                   | 5000.00            |
|   | 13          | Headset  | 1                  | 1                   | 900.00             |
|   | 13          | Keyboard | 1                  | 1                   | 400.00             |
|   | 13          | Monitor  | 1                  | 1                   | 12000.00           |
|   | 15          | Webcam   | 1                  | 1                   | 350.00             |
|   | 15          | Webcam   | 1                  | 1                   | 350.00             |
|   | 20          | DDR RAM  | 1                  | 1                   | 1500.00            |
|   | 23          | Keyboard | 2                  | 2                   | 800.00             |
|   | 23          | Monitor  | 1                  | 1                   | 12000.00           |
|   | 24          | Keyboard | 1                  | 1                   | 400.00             |
|   | 24          | Mousepad | 1                  | 1                   | 250.00             |
|   | 25          | Mousepad | 1                  | 1                   | 100.00             |

Result 2 ×

Output:

### Requirement Gaps:

The requirement does not clearly specify whether totals should be calculated at customer level or customer-product level

Also, transaction is not defined as to how the total number of transactions should be calculated.

Quantity column is not present in any data table

**Requirement3: the maximum product purchased for each country.**

### Test Cases:

Detailed test scenarios and test cases for this requirement are documented in the attached Excel file [Anwita\\_take\\_home\\_Assignment\\_test\\_cases.xlsx, Sheet: Req3.](#)

### SQL query to test the Expected value:

```
WITH items_country_counts AS (
    SELECT c.Country, o.Item, COUNT(*) AS purchase_count
    FROM Customers c
    JOIN Orders o ON o.Customer_ID = c.Customer_ID
    GROUP BY c.Country, o.Item
)
SELECT Country, Item AS max_purchased_product
FROM (
    SELECT
        Country, Item, purchase_count,
        DENSE_RANK() OVER (PARTITION BY Country ORDER BY purchase_count DESC) AS rnk
    FROM items_country_counts
) t
WHERE rnk = 1;
```

### Test result:

The screenshot shows a database interface with a query editor and a results grid.

**Query Editor:**

```
table_data_ingestion_queries Assignment*
26   FROM Customers c
27   JOIN Orders o ON o.Customer_ID = c.Customer_ID
28   GROUP BY c.Country, o.Item
29 )
30   SELECT Country, Item AS max_purchased_product
31   FROM (
32       SELECT
33           Country, Item, purchase_count,
34           DENSE_RANK() OVER (PARTITION BY Country ORDER BY purchase_count DESC) AS rnk
35       FROM items_country_counts
36   ) t
37   WHERE rnk = 1;
38
39
40
41
```

**Results Grid:**

| Country | max_purchased_product |
|---------|-----------------------|
| UAE     | Keyboard              |
| UK      | Mousepad              |
| USA     | Mousepad              |

### **Requirement Gap:**

The requirement does not define what “maximum product purchased” means (maximum by quantity or number of orders) and since the data source does not have a quantity field, the report can only determine the maximum product using count of order records. This may result in incorrect value reporting

**Requirement4: the most purchased product based on the age category less than 30 and above 30.**

### **Test Cases:**

*Detailed test scenarios and test cases for this requirement are documented in the attached Excel file Anwita\_take\_home Assignment test cases.xlsx, Sheet: Req4.*

### **SQL query to test the Expected value:**

```
WITH age_product_counts AS (
    SELECT
        CASE
            WHEN c.Age < 30 THEN 'Below 30'
            WHEN c.Age > 30 THEN 'Above 30'
        END AS age_category,
        o.Item,
        COUNT(*) AS purchase_count
    FROM Customers c
    JOIN Orders o
    ON o.Customer_ID = c.Customer_ID
    WHERE c.Age < 30 OR c.Age > 30
    GROUP BY age_category, o.Item
)
SELECT age_category, Item AS most_purchased_product
FROM (
    SELECT age_category,Item,
        DENSE_RANK() OVER (PARTITION BY age_category ORDER BY purchase_count DESC) AS rnk
    FROM age_product_counts
) t
WHERE rnk = 1;
```

### Test result:

The screenshot shows a database query editor window titled "Assignment" with the tab "table\_data\_ingestion\_queries". The query itself is as follows:

```
42    CASE
43        WHEN c.Age < 30 THEN 'Below 30'
44        WHEN c.Age > 30 THEN 'Above 30'
45    END AS age_category,
46    o.Item,
47    COUNT(*) AS purchase_count
48    FROM Customers c
49    JOIN Orders o
50        ON o.Customer_ID = c.Customer_ID
51    WHERE c.Age < 30 OR c.Age > 30
52    GROUP BY age_category, o.Item
53 )
54 SELECT age_category, Item AS most_purchased_product
55 FROM (
56     SELECT age_category,Item,
57     DENSE_RANK() OVER (PARTITION BY age_category ORDER BY purchase_count DESC) AS rnk
58     FROM age_product_counts
59 ) t
60 WHERE rnk = 1;
61
62
```

Below the query, there is a result grid table:

|     | age_category | most_purchased_product |
|-----|--------------|------------------------|
| 50) | Above 30     | Keyboard               |
|     | Below 30     | Mousepad               |

### Requirement Gap:

The requirement does not specify how customers aged exactly 30 should be handled

**Requirement5: the country that had minimum transactions and sales amount**

### Test Cases:

*Detailed test scenarios and test cases for this requirement are documented in the attached Excel file [Anwita take home Assignment test cases.xlsx](#), Sheet: Req5.*

**SQL query to test the Expected value:**

```
SELECT c.country
FROM Customers c
JOIN Orders o
ON o.Customer_ID = c.Customer_ID
GROUP BY c.Country
ORDER BY COUNT(DISTINCT o.Order_ID) ASC, SUM(o.Amount) ASC
LIMIT 1;
```

**Test result:**

The screenshot shows a database management system interface with the following details:

- Toolbar:** Includes icons for file operations, search, and navigation.
- Tab Bar:** Shows "table\_data\_ingestion\_queries" and "Assignment\*".
- Query Editor:** Displays the SQL query with line numbers 61 through 69. Lines 63-69 are highlighted in blue, indicating the executed portion of the code.
- Result Grid:** Shows the output of the query. The grid has one column labeled "country" and one row containing the value "UAE".
- Filter Row:** A row at the bottom of the grid with fields for "Filter Rows:", "Export:", "Wrap Cell Content:", and "Fetch rows:".