Yogesh Kodgire

**Technical Documentation**

**Data Model, Queries, and Reporting Framework**

[1. Overview 2](#_Toc190856061)

[2. Data Model Design 2](#_Toc190856062)

[2.1 Entities and Schema 2](#_Toc190856063)

[2.1.1 Entity: Order 2](#_Toc190856064)

[2.1.2 Entity: Customer 2](#_Toc190856065)

[2.1.3 Entity: Shipping 2](#_Toc190856066)

[2.2 Entity Relationship Diagram for given Data Model 3](#_Toc190856067)

[3. Data Flow Overview 4](#_Toc190856068)

[4. Data Relationships 4](#_Toc190856069)

[4.1 PowerBI Data Model 4](#_Toc190856070)

[5. Derived Fields / calculations Performed in Report view / Outside Transform data 5](#_Toc190856071)

[6. Database Queries and Corresponding Results 9](#_Toc190856072)

[7. Requirements for Anticipated Datasets 15](#_Toc190856073)

[7.1 Enhanced Table Structure / Entities and Schema 15](#_Toc190856074)

[Customers Table (Existing) 15](#_Toc190856075)

[Orders Table (Existing - Modified) 15](#_Toc190856076)

[Order Details Table (New) 15](#_Toc190856077)

[Products Table (Existing) 16](#_Toc190856078)

[Shipping Table (New) 16](#_Toc190856079)

[7.2 Data Flow Overview 16](#_Toc190856080)

[7.3 Data Relationships 16](#_Toc190856081)

[7.4 SQL Queries for Reporting Requirements 16](#_Toc190856082)

[7.4.1. Total Amount Spent and Country for Pending Delivery 16](#_Toc190856083)

[7.4.2. Customer-wise Transactions, Quantity Sold, and Amount Spent 17](#_Toc190856084)

[7.4.3. Maximum Purchased Product for Each Country 17](#_Toc190856085)

[7.4.4. Most Purchased Product by Age Category 17](#_Toc190856086)

[7.4.5. Country with Minimum Transactions and Sales 17](#_Toc190856087)

## 1. Overview

This document defines the data model for an e-commerce platform, supporting reporting requirements such as total sales, order status tracking, customer segmentation, and product performance analysis. The model includes structured datasets, relationships, and key technical specifications.

## 2. Data Model Design

### 2.1 Entities and Schema

### 2.1.1 Entity: Order

The Order entity captures essential information about a customer's purchase, as outlined below:

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| order\_id | INT | Unique identifier for the order. | PK, AUTO-INCREMENT |
| item | VARCHAR (100) | Name or description of the ordered item. | NOT NULL |
| amount | INT | Price of the ordered item. | NOT NULL |
| customer\_id | INT | Foreign key referencing the Customer table. | FK, NOT NULL |

### 2.1.2 Entity: Customer

The Customer entity captures details about the customer:

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| customer\_id | INT | Unique identifier for the customer. | PK, AUTO-INCREMENT |
| first\_name | VARCHAR (100) | First name of the customer. | NOT NULL |
| last\_name | VARCHAR (100) | Last name of the customer. | NOT NULL |
| age | INT | Age of the customer. | NULLABLE |
| country | VARCHAR (100) | Country of the customer. | NULLABLE |

### 2.1.3 Entity: Shipping

The Shipping entity captures the shipping details associated with orders:

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| shipping\_id | INT | Unique identifier for the shipping entry. | PK, AUTO-INCREMENT |
| status | INT | Status of the shipping (e.g., 0 - Pending, 1 - Shipped). | NOT NULL |
| customer | INT | Foreign key referencing the Customer table. | FK, NOT NULL |

### 2.2 Entity Relationship Diagram for given Data Model

Symbol Information

A black text and a white background

AI-generated content may be incorrect.A diagram of a diagram

AI-generated content may be incorrect.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 1 ER Diagram

## 3. Data Flow Overview

1. Source: Data is ingested from the e-commerce platform, CSV uploads, and APIs.
2. Staging: Raw data is loaded into staging tables for validation.
3. Transformation: Data is cleaned, enriched, and processed.
4. Loading: Processed data is inserted into respective tables.

## 4. Data Relationships

* Customer to Order: One-to-many relationship. One customer can place multiple orders.
* Customer to Shipping: One-to-many relationship. One customer can have multiple shipping records.

### 4.1 PowerBI Data Model

**A screenshot of a computer

AI-generated content may be incorrect.**

## Derived Fields / calculations Performed in Report view / Outside Transform data

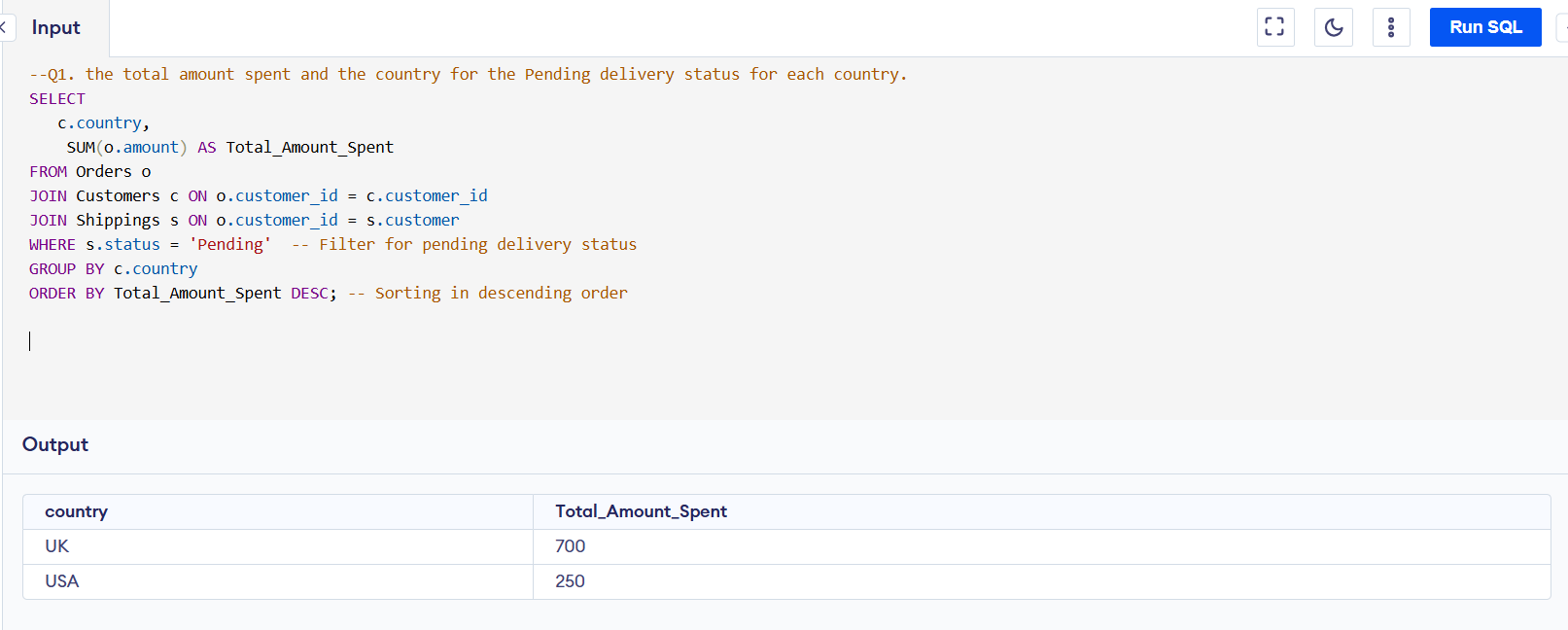
|  |  |  |
| --- | --- | --- |
| **Table/Column** | **Derived From** | **Logic** |
| Customer / TotalAmountSpent | Amount | **TotalAmountSpent = SUM('Order'[Amount])** |
| Customer/ TotalPendingSpend | Amount,  DeliveryStatus | TotalPendingSpend = -- Total Amount Spent for Pending Deliveries (By Country)  CALCULATE(      SUM('Order'[Amount]),      FILTER(Shipping, Shipping[DeliveryStatus] = "Pending")  ) |
| Customer / OrderID | OrderID | TotalTransactions = COUNT('Order'[OrderID]) |
| Order / Country with Min Transactions | Country,  TotalTransactionsByCountry,  MinTransactionsByCountry | Country with Min Transactions =  VAR TotalTransactionsByCountry = COUNTROWS('Order')  VAR MinTransactionsByCountry =  CALCULATE(      MINX(          VALUES('Customer'[Country]),          TotalTransactionsByCountry      )  )  RETURN  CALCULATE(      FIRSTNONBLANK('Customer'[Country], TotalTransactionsByCountry),      FILTER(          ALL('Customer'[Country]),          TotalTransactionsByCountry = MinTransactionsByCountry      )  ) |
| Customer / Customer Name | First,  Last | Customer Name = Customer[First]& " " & Customer[Last] |
| Order / Product Details | Item | Product Details =  CONCATENATEX(      VALUES('Order'[Item]),      'Order'[Item],      ", "  ) |
| Customer / AgeCategory | Age | AgeCategory =  IF(      Customer [Age] < 30, "Under 30", "30 and Above"  ) |
| Order / MostPurchasedProduct\_AgeCategory | Item,  CustomerID,  Age,  OrderID | MostPurchasedProduct\_AgeCategory =  VAR AgeCategoryTable =      ADDCOLUMNS (          SUMMARIZE(              'Order',              'Order'[Item],              'Order'[CustomerID]          ),          "Customer\_Age", LOOKUPVALUE(Customer[Age], Customer[CustomerID], 'Order'[CustomerID]),          "TotalQty", COUNTROWS('Order')      )  VAR Under30 =      TOPN(1,          SUMMARIZE(              FILTER(AgeCategoryTable, [Customer\_Age] < 30),              'Order'[Item],              "TotalQty", COUNT('Order'[OrderID])  -- counting total orders per product          ),          [TotalQty], DESC      )  VAR Above30 =      TOPN(1,          SUMMARIZE(              FILTER(AgeCategoryTable, [Customer\_Age] >= 30),              'Order'[Item],              "TotalQty", COUNT('Order'[OrderID])          ),          [TotalQty], DESC      )  RETURN      CONCATENATEX(          UNION(Under30, Above30),          'Order'[Item],          ", ",          'Order'[Item],          ASC      )  -- Ensures only \*\*one\*\* product is returned per age category. |
| Order / MostPurchasedProduct\_PerCountry | Item,  Country,  Order Table | MostPurchasedProduct\_PerCountry =  VAR ProductSalesTable =      SUMMARIZE(          'Order',          'Order'[Item],          'Customer'[Country],          "TotalQty", COUNTROWS('Order')  -- counting the total orders per product      )  VAR MaxProductSales =      ADDCOLUMNS(          ProductSalesTable,          "MaxQty",          CALCULATE(MAXX(ProductSalesTable, [TotalQty]))  -- getting the maximum total quantity per country      )  VAR MostPurchasedProduct =      FILTER(          MaxProductSales,          [TotalQty] = [MaxQty]      )  RETURN      CONCATENATEX(          MostPurchasedProduct,          'Order'[Item],          ", "  -- Concatenates product names with commas if there are multiple      ) |
| Order / MostPurchasedProduct\_AgeCategory | Item,  CustomerID,  Age,  OrderID | MostPurchasedProduct\_AgeCategory =  VAR AgeCategoryTable =      ADDCOLUMNS(          SUMMARIZE(              'Order',              'Order'[Item],              'Order'[CustomerID]          ),          "Customer\_Age", LOOKUPVALUE(Customer[Age], Customer[CustomerID], 'Order'[CustomerID]),          "TotalQty", COUNTROWS('Order')      )  VAR Under30 =      TOPN(1,          SUMMARIZE(              FILTER(AgeCategoryTable, [Customer\_Age] < 30),              'Order'[Item],              "TotalQty", COUNT('Order'[OrderID])  -- counting total orders per product          ),          [TotalQty], DESC      )  VAR Above30 =      TOPN(1,          SUMMARIZE(              FILTER(AgeCategoryTable, [Customer\_Age] >= 30),              'Order'[Item],              "TotalQty", COUNT('Order'[OrderID])          ),          [TotalQty], DESC      )  RETURN      CONCATENATEX(          UNION(Under30, Above30),          'Order'[Item],          ", ",          'Order'[Item],          ASC      )  -- Ensures only \*\*one\*\* product is returned per age category. |
| Order / TopProductByCountry | Orders\_Country,  Item | TopProductByCountry =  VAR ProductTable =      ADDCOLUMNS(          SUMMARIZE('Order', 'Order'[Orders\_Country], 'Order'[Item]),          "TotalQty",          CALCULATE(COUNTROWS('Order'), ALLEXCEPT('Order', 'Order'[Orders\_Country], 'Order'[Item]))      )  VAR TopProduct =      TOPN(1, ProductTable, [TotalQty], DESC)   --  top product per country instead of globally.  RETURN      CONCATENATEX(TopProduct, 'Order'[Item], ", ") -- If multiple top products have the same quantity, they will be concatenated. |

## Database Queries and Corresponding Results

Q1. the total amount spent and the country for the Pending delivery status for each country.

|  |
| --- |
| SELECT  c.country,  SUM(o.amount) AS Total\_Amount\_Spent  FROM Orders o  JOIN Customers c ON o.customer\_id = c.customer\_id  JOIN Shippings s ON o.customer\_id = s.customer  WHERE s.status = 'Pending' -- Filter for pending delivery status  GROUP BY c.country  ORDER BY Total\_Amount\_Spent DESC; -- Sorting in descending order |

Output:

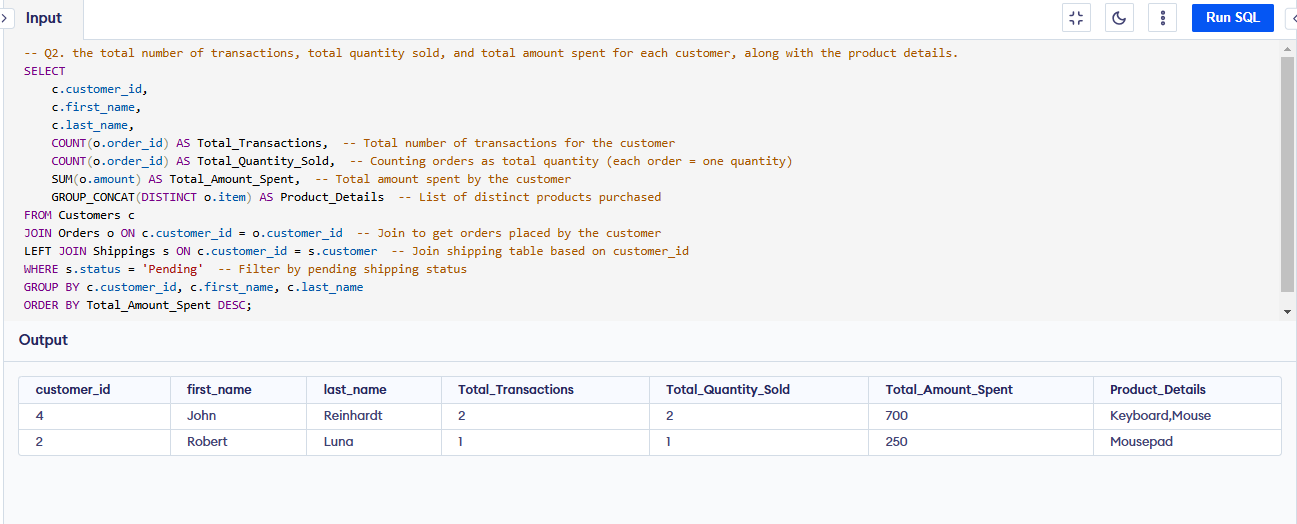


Q2. the total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details.

Ans:

|  |
| --- |
| SELECT  c.customer\_id, c.first\_name, c.last\_name,  COUNT(o.order\_id) AS Total\_Transactions, -- Total number of transactions for the customer  COUNT(o.order\_id) AS Total\_Quantity\_Sold, -- Counting orders as total quantity (each order = one quantity)  SUM(o.amount) AS Total\_Amount\_Spent, -- Total amount spent by the customer  GROUP\_CONCAT(DISTINCT o.item) AS Product\_Details -- List of distinct products purchased  FROM Customers c  JOIN Orders o ON c.customer\_id = o.customer\_id -- Join to get orders placed by the customer  LEFT JOIN Shippings s ON c.customer\_id = s.customer -- Join shipping table based on customer\_id  WHERE s.status = 'Pending' -- Filter by pending shipping status  GROUP BY c.customer\_id, c.first\_name, c.last\_name  ORDER BY Total\_Amount\_Spent DESC; |

**OutPut:**



Q3. the maximum product purchased for each country.

Ans:

|  |
| --- |
| -- This Query will return only one product per country (even if multiple products have the same count  SELECT country, product AS most\_purchased\_product, total\_transactions  FROM (  SELECT  c.country,  o.item AS product,  COUNT(o.order\_id) AS total\_transactions,  ROW\_NUMBER() OVER (PARTITION BY c.country ORDER BY COUNT(o.order\_id) DESC) AS rank  FROM Orders o  JOIN Customers c ON o.customer\_id = c.customer\_id  GROUP BY c.country, o.item  ) ranked  WHERE rank = 1; |

**OutPut:**

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
| --Keeps multiple products if they have the same max transaction count in a country.  SELECT country, product AS most\_purchased\_product, total\_transactions  FROM (  SELECT  c.country,  o.item AS product,  COUNT(o.order\_id) AS total\_transactions,  DENSE\_RANK() OVER (PARTITION BY c.country ORDER BY COUNT(o.order\_id) DESC) AS rank  FROM Orders o  JOIN Customers c ON o.customer\_id = c.customer\_id  GROUP BY c.country, o.item  ) ranked  WHERE rank = 1; |

A screenshot of a computer

AI-generated content may be incorrect.

Q4. the most purchased product based on the age category less than 30 and above 30.

Ans:

|  |
| --- |
| WITH AgeCategory AS ( -- categorizes customers into two groups based on their age: "Under 30" and "30 and Above"  SELECT  c.customer\_id, c.age,  CASE  WHEN c.age < 30 THEN 'Under 30'  ELSE '30 and Above'  END AS age\_category  FROM  Customers c  ),  -- calculate the total number of transactions (or total sales) for each product within each age category.  ProductSales AS (  SELECT  a.age\_category,  o.item,  COUNT(o.order\_id) AS total\_transactions  FROM  Orders o  JOIN  AgeCategory a ON o.customer\_id = a.customer\_id  GROUP BY  a.age\_category, o.item  ),  -- rank the products within each age category based on the total number of transactions in descending order  RankedProducts AS (  SELECT  age\_category, item AS product, total\_transactions,  ROW\_NUMBER() OVER (PARTITION BY age\_category ORDER BY total\_transactions DESC) AS rn  FROM  ProductSales  )  SELECT  age\_category,  product AS most\_purchased\_product,  total\_transactions  FROM  RankedProducts  WHERE  rn = 1; -- filter out the top-ranked product for each age category |

**OutPut:**

A screenshot of a computer

AI-generated content may be incorrect.

Q5. the country that had minimum transactions and sales amount.

Ans:

|  |
| --- |
| WITH CountrySales AS (  SELECT  c.country, COUNT(o.order\_id) AS total\_transactions,  SUM(o.amount) AS total\_sales  FROM  Orders o  JOIN  Customers c ON o.customer\_id = c.customer\_id  GROUP BY  c.country  )  SELECT  country, total\_transactions, total\_sales  FROM  CountrySales  ORDER BY  total\_transactions ASC, total\_sales ASC  LIMIT 2; -- One can change the limit value ans see number of countries with minimum transaction |

**OutPut:**



## Requirements for Anticipated Datasets

With the current table structure, it's difficult to capture Order Quantity, Order Information (Ordered Date, Price Per Item), and Shipping Details (Shipping Date, etc.). To improve data granularity and enable better reporting, we can introduce additional tables.

### 7.1 Enhanced Table Structure / Entities and Schema

#### Customers Table (Existing)

Purpose: Stores customer demographic details.

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| customer\_id | INT (PK) | Unique identifier for each customer. | Primary Key |
| first\_name | VARCHAR (100) | Customer's first name. | NOT NULL |
| last\_name | VARCHAR (100) | Customer's last name. | NOT NULL |
| age | INT | Customer's age. | NULLABLE |
| country | VARCHAR (100) | Customer's country. | NOT NULL |

#### Orders Table (Existing - Modified)

Purpose: Captures purchase transactions.

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| order\_id | INT (PK) | Unique identifier for each order. | Primary Key, AUTO-INCREMENT |
| customer\_id | INT (FK) | References Customers (customer\_id). | Foreign Key, NOT NULL |
| order\_date | DATETIME | Date and time when the order was placed. | NOT NULL |
| total\_amount | DECIMAL(10,2) | Total order amount. | NOT NULL |
| status | VARCHAR(20) | Order status (PENDING, SHIPPED, DELIVERED, etc.). | NULLABLE |
| country | VARCHAR(100) | Country where the order was placed. | NOT NULL |

#### Order Details Table (New)

Purpose: This table will store detailed information about each order item.

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| order\_detail\_id | INT (PK) | Unique identifier for each order detail. | Primary Key |
| order\_id | INT (FK) | References Orders (order\_id). | Foreign Key, NOT NULL |
| product\_id | INT (FK) | References Products (product\_id). | Foreign Key, NOT NULL |
| quantity | INT | Number of units purchased. | NOT NULL |
| price | DECIMAL(10,2) | Price per unit. | NOT NULL |
| total\_price | DECIMAL(10,2) | Calculated as (quantity \* price) | NOT NULL |

#### Products Table (Existing)

Purpose: Stores information about available products.

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| product\_id | INT (PK) | Unique identifier for each product. | Primary Key |
| product\_name | VARCHAR(100) | Name of the product. | NOT NULL |
| category | VARCHAR(100) | Product category. | NOT NULL |

#### Shipping Table (New)

Purpose: Captures shipment status of orders.

| Column Name | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| shipping\_id | INT (PK) | Unique identifier for each shipment. | Primary Key |
| order\_id | INT (FK) | References Orders (order\_id). | Foreign Key, NOT NULL |
| status | VARCHAR(20) | Shipping status (PENDING, IN TRANSIT, DELIVERED). | NULLABLE |
| customer\_id | INT (FK) | References Customers (customer\_id). | Foreign Key, NOT NULL |

### 7.2 Data Flow Overview

1. Source: Data is ingested from the e-commerce platform, CSV uploads, and APIs.
2. Staging: Raw data is loaded into staging tables for validation.
3. Transformation: Data is cleaned, enriched, and processed.
4. Loading: Processed data is inserted into respective tables.

### 7.3 Data Relationships

* One-to-Many: Customers → Orders (one customer can have multiple orders).
* One-to-Many: Orders → Order Details (one order can have multiple products).
* Many-to-One: Order Details → Products (multiple order details reference a single product).
* One-to-One: Orders → Shipping (each order has one shipping record).

### 7.4 SQL Queries for Reporting Requirements

### 7.4.1. Total Amount Spent and Country for Pending Delivery

|  |
| --- |
| SELECT country, SUM(total\_amount) AS total\_spent  FROM Orders  WHERE status = 'PENDING'  GROUP BY country; |

### 7.4.2. Customer-wise Transactions, Quantity Sold, and Amount Spent

|  |
| --- |
| SELECT c.customer\_id, c.first\_name, c.last\_name, p.product\_name,  COUNT(o.order\_id) AS total\_transactions,  SUM(od.quantity) AS total\_quantity,  SUM(od.total\_price) AS total\_spent  FROM Customers c  JOIN Orders o ON c.customer\_id = o.customer\_id  JOIN Order\_Details od ON o.order\_id = od.order\_id  JOIN Products p ON od.product\_id = p.product\_id  GROUP BY c.customer\_id, c.first\_name, c.last\_name, p.product\_name; |

### 7.4.3. Maximum Purchased Product for Each Country

|  |
| --- |
| SELECT o.country, p.product\_name, SUM(od.quantity) AS total\_sold  FROM Orders o  JOIN Order\_Details od ON o.order\_id = od.order\_id  JOIN Products p ON od.product\_id = p.product\_id  GROUP BY o.country, p.product\_name  ORDER BY o.country, total\_sold DESC; |

### 7.4.4. Most Purchased Product by Age Category

|  |
| --- |
| SELECT  CASE WHEN c.age < 30 THEN 'Under 30'  ELSE '30 and Above' END  AS age\_category,  p.product\_name, SUM(od.quantity) AS total\_sold  FROM Customers c  JOIN Orders o ON c.customer\_id = o.customer\_id  JOIN Order\_Details od ON o.order\_id = od.order\_id  JOIN Products p ON od.product\_id = p.product\_id  GROUP BY age\_category, p.product\_name  ORDER BY age\_category, total\_sold DESC; |

### 7.4.5. Country with Minimum Transactions and Sales

|  |
| --- |
| SELECT country,  COUNT(order\_id) AS total\_orders,  SUM(total\_amount) AS total\_sales  FROM Orders  GROUP BY country  ORDER BY total\_sales ASC  LIMIT 1; |