# ASSIGNMENT TWO (Deliverable #2)

Muhammad Haziq Ijaz (i21-2692)

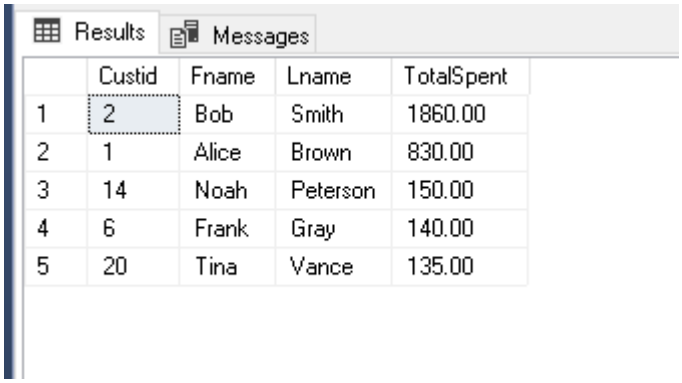
Qasim Rizwan (i21-2678)

CYBER-T

## SQL Questions:

4. List the top 5 customers who spent the most money:

**Explanation :**
First I extracted the records of Customer from customer table and then I joined it with the order table on Custid to extract the orders of the customer and then displayed the information By Descending order on a new Column called Total Spent.

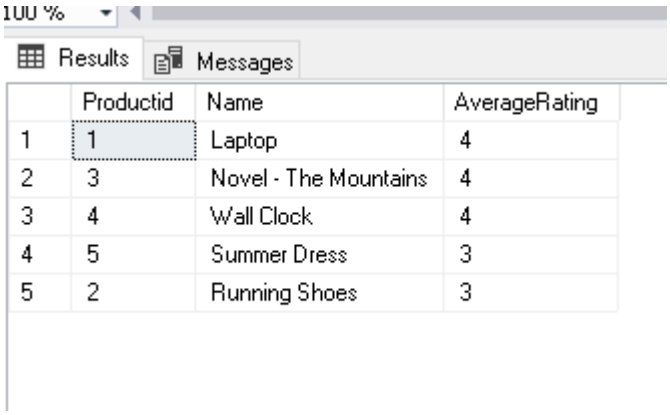


Results | Messages

| | Custid | Fname | Lname | TotalSpent |
|---|---|---|---|---|
| 1 | 2 | Bob | Smith | 1860.00 |
| 2 | 1 | Alice | Brown | 830.00 |
| 3 | 14 | Noah | Peterson | 150.00 |
| 4 | 6 | Frank | Gray | 140.00 |
| 5 | 20 | Tina | Vance | 135.00 |

5. Retrieve products with the highest average ratings:

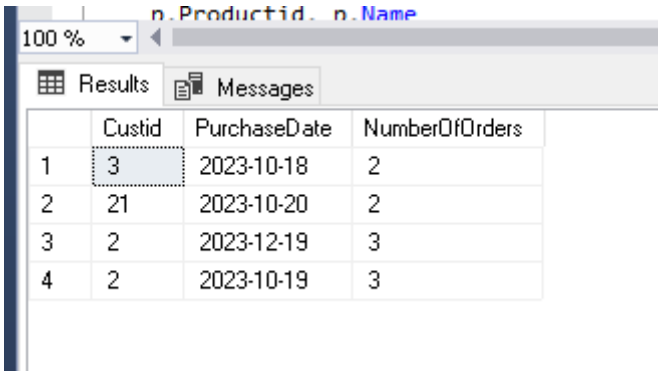
**Explanation:** In this query I retrieved the average rating for each product and sorted the products based on their average rating in descending order. By retrieving the information from product table and joining it with Review table on productid and ordering it by ascending order.




| | Productid | Name | AverageRating |
|---|---|---|---|
| 1 | 1 | Laptop | 4 |
| 2 | 3 | Novel - The Mountains | 4 |
| 3 | 4 | Wall Clock | 4 |
| 4 | 5 | Summer Dress | 3 |
| 5 | 2 | Running Shoes | 3 |

6. Find customers who have made more than one purchase on the same day:

Explanation: In this query, I retrieved the dates on which each customer made purchases and the number of orders they made on those dates. By selecting information from the ORDER_ table and grouping it by the customer ID and the purchase date (with the time component removed), I was able to identify dates where customers made more than one order. Finally, I sorted the results by the number of orders in ascending order, allowing us to easily identify customers who had multiple orders on specific dates, starting with those who had the fewest.

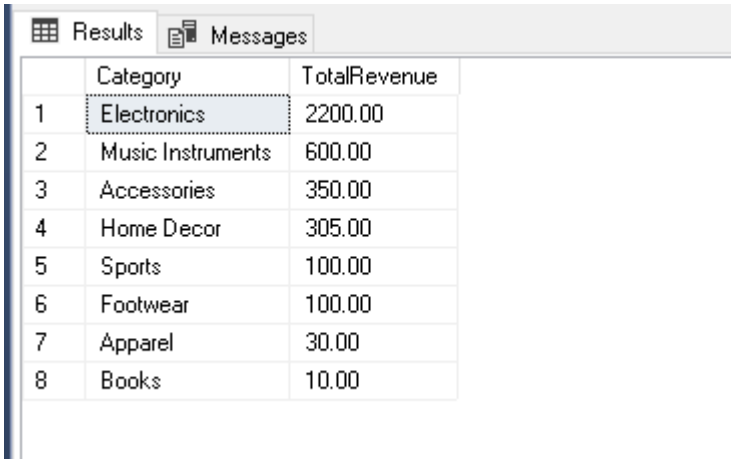



| | Custid | PurchaseDate | NumberOfOrders |
|---|---|---|---|
| 1 | 3 | 2023-10-18 | 2 |
| 2 | 21 | 2023-10-20 | 2 |
| 3 | 2 | 2023-12-19 | 3 |
| 4 | 2 | 2023-10-19 | 3 |

7. Calculate the total revenue for each product category :

Explanation: In this query, I calculated the total revenue generated by each product category. By accessing data from the PRODUCT table and joining it with the ORDERITEM

table on Productid, I aggregated the revenue for each category by multiplying the product's price with the quantity sold. Then, I grouped the results by the Category column to consolidate the revenue for products under the same category. Finally, the categories are sorted by the total revenue they generated in descending order, displaying the highest earning categories at the top of the list.



Results | Messages

| | Category | TotalRevenue |
|---|---|---|
| 1 | Electronics | 2200.00 |
| 2 | Music Instruments | 600.00 |
| 3 | Accessories | 350.00 |
| 4 | Home Decor | 305.00 |
| 5 | Sports | 100.00 |
| 6 | Footwear | 100.00 |
| 7 | Apparel | 30.00 |
| 8 | Books | 10.00 |

8. List customers who have not reviewed any products:

**Explanation**: In this query, I retrieved the details of customers who haven't written any reviews. By accessing data from the CUSTOMER table and performing a LEFT JOIN with the REVIEW table on Custid, I can find records where there is no corresponding review. I then filtered these records with a WHERE clause, specifically looking for instances where ReviewID is NULL, which indicates the absence of a review from that particular customer.

Results | Messages

| | Custid | Fname | Lname |
|---|---|---|---|
| 1 | 6 | Frank | Gray |
| 2 | 7 | Grace | Harris |
| 3 | 8 | Henry | Johnson |
| 4 | 9 | Irene | King |
| 5 | 10 | Jack | Lopez |
| 6 | 11 | Kara | Mills |
| 7 | 12 | Liam | Nelson |
| 8 | 13 | Mia | Ortiz |
| 9 | 14 | Noah | Peterson |
| 10 | 15 | Olivia | Quinn |
| 11 | 16 | Peter | Russell |
| 12 | 17 | Quincy | Stewart |
| 13 | 18 | Rachel | Turner |
| 14 | 19 | Steve | Upton |
| 15 | 20 | Tina | Vance |
| 16 | 21 | John | Doe |

9. Find products with quantities below the average quantity in stock

**Explanation**: In this query, I gathered details about products that have a stock quantity less than the average stock quantity of all products. By accessing data from the PRODUCT table and joining it with the INVENTORY table on Productid, I was able to obtain each product's stock quantity. Then, I added a WHERE clause that filters products with stock quantities less than the average stock quantity found in the INVENTORY table.

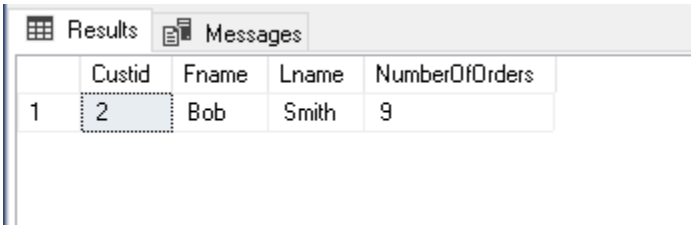
Results | Messages

| | Productid | ProductName | StockQuantity |
|---|---|---|---|
| 1 | 3 | Novel - The Mountains | 80 |
| 2 | 5 | Summer Dress | 90 |
| 3 | 3 | Novel - The Mountains | 75 |
| 4 | 1 | Laptop | 85 |
| 5 | 5 | Summer Dress | 80 |
| 6 | 1 | Laptop | 90 |
| 7 | 3 | Novel - The Mountains | 70 |
| 8 | 5 | Summer Dress | 95 |

10. Calculate the total number of orders for each customer and show only those with more than 5 orders.

**Explanation:**

In this query, I calculated the number of orders placed by each customer. By accessing data from the ORDER_ table and joining it with the CUSTOMER table on Custid, I aggregated the number of orders each customer made. I then grouped the results by Custid, Fname, and Lname to consolidate the count of orders for each customer. The HAVING clause then filters out customers who have placed 5 or fewer orders, ensuring that only customers with more than 5 orders are displayed in the results.

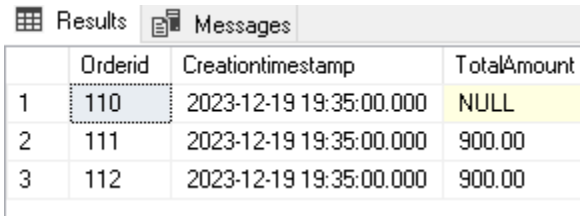


Results | Messages

| | Custid | Fname | Lname | NumberOfOrders |
|---|---|---|---|---|
| 1 | 2 | Bob | Smith | 9 |

11. Retrieve the 3 most recent orders for a specific customer

**Explanation:** In this query, I fetched the top 3 most recent orders placed by the customer with Custid equal to 2. By accessing data from the ORDER_ table and adding a WHERE clause for Custid = 2, I focused on the orders related to this specific customer. To determine the most recent orders, I sorted the results by the Creationtimestamp in descending order and limited the output to the first 3 records using the TOP keyword.

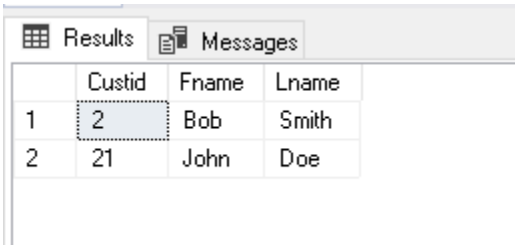


Results | Messages

| | Orderid | Creationtimestamp | TotalAmount |
|---|---|---|---|
| 1 | 110 | 2023-12-19 19:35:00.000 | NULL |
| 2 | 111 | 2023-12-19 19:35:00.000 | 900.00 |
| 3 | 112 | 2023-12-19 19:35:00.000 | 900.00 |

12. List customers who have purchased products from at least two different sellers.

**Explanation:** In this query, I extracted customer details who have purchased products from at least two distinct sellers. By accessing data from the ORDER_ table and joining it successively with the ORDERITEM, PRODUCT, and CUSTOMER tables, I gathered information on every product each customer has bought, along with the sellers of those products. The results are then grouped by customer identifiers and names, with the HAVING clause filtering customers who have bought from two or more unique sellers.



Results | Messages

| | Custid | Fname | Lname |
|---|---|---|---|
| 1 | 2 | Bob | Smith |
| 2 | 21 | John | Doe |

13. Find customers who have placed an order in the last 30 days.

**Explanation:**

In this query, I retrieved a list of distinct customers who have placed an order within the last 30 days. By accessing data from the ORDER_ table and joining it with the CUSTOMER table, I obtained a list of all orders alongside their corresponding customer details. The WHERE clause then filters out orders that were created more than 30 days ago.

Results | Messages

| | Custid | Fname | Lname |
|---|---|---|---|
| 1 | 1 | Alice | Brown |
| 2 | 2 | Bob | Smith |
| 3 | 3 | Charlie | Green |
| 4 | 4 | David | White |
| 5 | 5 | Eva | Black |
| 6 | 6 | Frank | Gray |
| 7 | 7 | Grace | Harris |
| 8 | 8 | Henry | Johnson |
| 9 | 9 | Irene | King |
| 10 | 10 | Jack | Lopez |
| 11 | 11 | Kara | Mills |
| 12 | 12 | Liam | Nelson |
| 13 | 13 | Mia | Ortiz |
| 14 | 14 | Noah | Peterson |
| 15 | 15 | Olivia | Quinn |
| 16 | 16 | Peter | Russell |
| 17 | 17 | Quincy | Stewart |
| 18 | 18 | Rachel | Turner |
| 19 | 19 | Steve | Upton |
| 20 | 20 | Tina | Vance |
| 21 | 21 | John | Doe |

14. List customers who have made a purchase in every product category:

**Explanation**

In this query, I extracted the details of customers who have purchased products from every unique product category available in the store. By accessing data from the ORDER_ table and joining it successively with the ORDERITEM, PRODUCT, and CUSTOMER tables, I gathered information on each product category that a customer has bought from. The results are then grouped by the customer identifiers and names. The HAVING clause is crucial in this query as it filters for customers whose count of distinct product categories they've purchased from equals the total count of unique product categories available in the PRODUCT table. Essentially, this query identifies customers who have sampled products from all available categories in the store.

Results Messages

| | Custid | Fname | Lname |
|---|---|---|---|
| 1 | 511 | John | Doe |
| 2 | 512 | Jane | Smith |

:

15. Calculate the total number of products sold by each seller:

**Explanation:**

In this query, I calculated the total quantity of products sold by each seller. By accessing data from the _SELLER table and joining it sequentially with the PRODUCT and ORDERITEM tables, I was able to aggregate the quantities sold for products associated with each seller. The results are then grouped by seller identifiers and names to consolidate the total quantities sold by each seller.

Results Messages

| | Sellerid | SellerName | TotalProductsSold |
|---|---|---|---|
| 1 | 1 | TechStore | 17 |
| 2 | 2 | FashionHub | 6 |
| 3 | 3 | BookWorld | 14 |
| 4 | 4 | HomeDecor | 18 |
| 5 | 5 | ShoeMart | 5 |
| 6 | 10 | SportsGear | 7 |
| 7 | 20 | CraftCave | 4 |

16. Retrieve the top 5 products with the highest sales in the last month

**Explanation:**

Results | Messages

| | Productid | Name | TotalSales |
|---|---|---|---|
| 1 | 1 | Laptop | 3200.00 |
| 2 | 6 | Smartphone | 2500.00 |
| 3 | 10 | Coffee Table | 750.00 |
| 4 | 8 | Bluetooth Speaker | 360.00 |
| 5 | 17 | Jacket - Men | 330.00 |

17. Retrieve the latest 5 orders along with customer details and order items for each order.

**Explanation:**
In this query, I retrieved the details of the top 5 most recent orders. By accessing the ORDER_ table and joining it with the CUSTOMER and ORDERITEM tables, I collected details of the order, the customer who placed the order, and the items in that order. The results are then sorted by the order creation timestamp in descending order, showing the latest orders first, and the TOP keyword limits the output to just the 5 most recent orders.

Results | Messages

| | Orderid | Creationtimestamp | Custid | Fname | Lname | Productid |
|---|---|---|---|---|---|---|
| 1 | 601 | 2023-10-21 17:08:12.020 | 511 | John | Doe | 1111 |
| 2 | 601 | 2023-10-21 17:08:12.020 | 511 | John | Doe | 1110 |
| 3 | 601 | 2023-10-21 17:08:12.020 | 511 | John | Doe | 1113 |
| 4 | 601 | 2023-10-21 17:08:12.020 | 511 | John | Doe | 1114 |
| 5 | 601 | 2023-10-21 17:08:12.020 | 511 | John | Doe | 1115 |

18. Retrieve customers who have made purchases in every product category, along with the total number of categories they have purchased from.

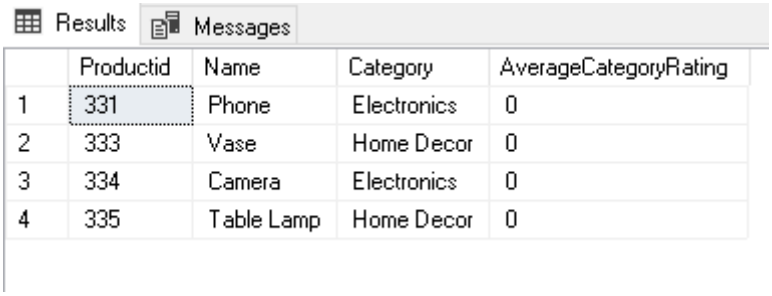
**Explanation:**
it extracts the details of customers who have purchased products from every unique product category available in the store.

Results | Messages

| | Custid | Fname | Lname | TotalCategoriesPurchased |
|---|---|---|---|---|
| 1 | 511 | John | Doe | 8 |
| 2 | 512 | Jane | Smith | 8 |

19. List products that have never been reviewed and have quantities in stock greater than zero, along with the average rating for their category.

**Explanation:** In this query, I extracted product details, specifically products that have never been reviewed but still have a quantity greater than 0. For each product, an average rating is computed; however, since these products haven't been reviewed, the COALESCE function sets the average rating to 0.

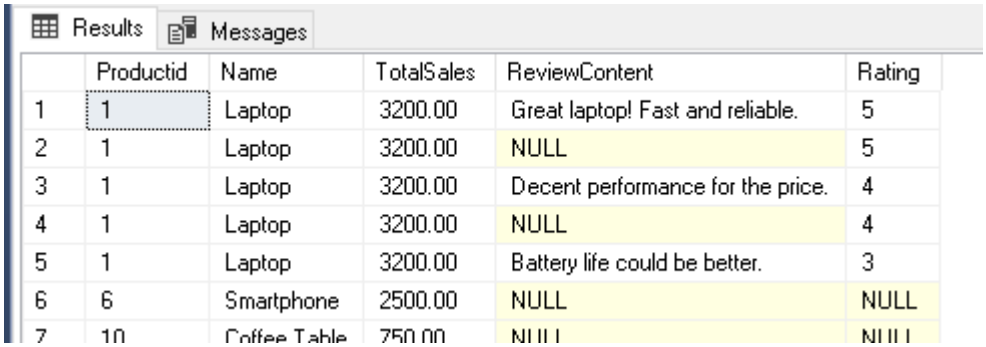


Results | Messages

| | Productid | Name | Category | AverageCategoryRating |
|---|---|---|---|---|
| 1 | 331 | Phone | Electronics | 0 |
| 2 | 333 | Vase | Home Decor | 0 |
| 3 | 334 | Camera | Electronics | 0 |
| 4 | 335 | Table Lamp | Home Decor | 0 |

20. Find the top 3 products with the highest total sales, including details of the reviews for each product.

**Explanation:**
Using a Common Table Expression (CTE) named 'ProductSales', I first determined the top 3 products by sales. Then, in the main query, I retrieved their associated reviews and ratings. If a top-selling product doesn't have a review, it will still appear in the result, but without review details.



Results | Messages

| | Productid | Name | TotalSales | ReviewContent | Rating |
|---|---|---|---|---|---|
| 1 | 1 | Laptop | 3200.00 | Great laptop! Fast and reliable. | 5 |
| 2 | 1 | Laptop | 3200.00 | NULL | 5 |
| 3 | 1 | Laptop | 3200.00 | Decent performance for the price. | 4 |
| 4 | 1 | Laptop | 3200.00 | NULL | 4 |
| 5 | 1 | Laptop | 3200.00 | Battery life could be better. | 3 |
| 6 | 6 | Smartphone | 2500.00 | NULL | NULL |
| 7 | 10 | Coffee Table | 750.00 | NULL | NULL |

**21. Retrieve all customers who have placed orders, and include details of their orders, even for orders with no associated customers. Include information about the shipping addresses for each order:**

**Explanation**:In this query, I extracted order details along with the customer's information and shipping address. By accessing the ORDER_ table and performing LEFT JOINs with the CUSTOMER and ADDRESS tables, I ensured that even orders without associated customer details or addresses are still displayed**.**

Results Messages

| | Custid | Fname | Lname | Orderid | Creationtimestamp | AddressLine1 | City | State | Zip | Country |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Alice | Brown | 1 | 2023-10-20 17:45:49.880 | 123 Tech St | TechCity | CA | 12345 | USA |
| 2 | 2 | Bob | Smith | 2 | 2023-10-20 17:45:49.880 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 3 | 3 | Charlie | Green | 3 | 2023-10-20 17:45:49.880 | 789 Fashion Ave | StyleCity | TX | 78901 | USA |
| 4 | 4 | David | White | 4 | 2023-10-20 17:45:49.880 | 101 Home Ln | DecorTown | OH | 10111 | USA |
| 5 | 5 | Eva | Black | 5 | 2023-10-20 17:45:49.880 | 202 Shoe Blvd | ShoeCity | FL | 20222 | USA |
| 6 | 6 | Frank | Gray | 6 | 2023-10-20 17:45:49.880 | 303 Tech Plaza | CompCity | GA | 30333 | USA |
| 7 | 7 | Grace | Harris | 7 | 2023-10-20 17:45:49.880 | 404 Book Park | LibraTown | CO | 40444 | USA |
| 8 | 8 | Henry | Johnson | 8 | 2023-10-20 17:45:49.880 | 505 Fashion Street | GlamCity | WA | 50555 | USA |
| 9 | 9 | Irene | King | 9 | 2023-10-20 17:45:49.880 | 606 Home Gardens | PeaceTown | UT | 60666 | USA |
| 10 | 10 | Jack | Lopez | 10 | 2023-10-20 17:45:49.880 | 707 Shoe Lane | FootCity | TN | 70777 | USA |
| 11 | 11 | Kara | Mills | 11 | 2023-10-20 17:45:49.880 | 808 Tech Tower | InnovCity | PA | 80888 | USA |
| 12 | 12 | Liam | Nelson | 12 | 2023-10-20 17:45:49.880 | 909 Book Bridge | WriteTown | NM | 90999 | USA |
| 13 | 13 | Mia | Ortiz | 13 | 2023-10-20 17:45:49.880 | 1010 Fashion Drive | ChicCity | AZ | 10101 | USA |
| 14 | 14 | Noah | Peterson | 14 | 2023-10-20 17:45:49.880 | 1111 Home Heights | EleganceTown | NV | 11112 | USA |
| 15 | 15 | Olivia | Quinn | 15 | 2023-10-20 17:45:49.880 | 1212 Shoe Square | StrideCity | NJ | 12123 | USA |
| 16 | 16 | Peter | Russell | 16 | 2023-10-20 17:45:49.880 | 1313 Tech Tunnel | ElectroCity | VA | 13134 | USA |
| 17 | 17 | Quincy | Stewart | 17 | 2023-10-20 17:45:49.880 | 1414 Book Boulevard | LiteraryLands | OR | 14145 | USA |
| 18 | 18 | Rachel | Turner | 18 | 2023-10-20 17:45:49.880 | 1515 Fashion Freeway | VogueVillage | KS | 15156 | USA |
| 19 | 19 | Steve | Upton | 19 | 2023-10-20 17:45:49.880 | 1616 Home Harbor | LushLane | OK | 16167 | USA |
| 20 | 20 | Tina | Vance | 20 | 2023-10-20 17:45:49.880 | 1717 Shoe Shore | WalkwayCity | LA | 17178 | USA |
| 21 | 21 | John | Doe | 21 | 2023-10-20 19:37:44.277 | NULL | NULL | NU... | NULL | NULL |
| 22 | 21 | John | Doe | 22 | 2023-10-20 19:37:47.803 | NULL | NULL | NU... | NULL | NULL |
| 23 | 2 | Bob | Smith | 103 | 2023-10-19 09:15:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 24 | 2 | Bob | Smith | 104 | 2023-10-19 17:05:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 25 | 2 | Bob | Smith | 105 | 2023-10-19 19:35:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 26 | 3 | Charlie | Green | 106 | 2023-10-18 08:15:00.000 | 789 Fashion Ave | StyleCity | TX | 78901 | USA |
| 27 | 3 | Charlie | Green | 107 | 2023-10-18 08:25:00.000 | 789 Fashion Ave | StyleCity | TX | 78901 | USA |
| 28 | 2 | Bob | Smith | 108 | 2023-11-19 09:15:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 29 | 2 | Bob | Smith | 109 | 2023-11-29 17:05:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 30 | 2 | Bob | Smith | 110 | 2023-12-19 19:35:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |
| 31 | 2 | Bob | Smith | 111 | 2023-12-19 19:35:00.000 | 456 Book Rd | ReadTown | NY | 45678 | USA |

**22. Write a SQL query to retrieve the total number of products and the total revenue for each product category. Include products that may not have been sold. Additionally, order the product in descending order based on**

## total revenue:

**Explanation:**
Here, I computed the total revenue and the number of unique products for each product category. Products that haven't been sold are still considered, but their revenue contribution is set to 0 using COALESCE. The results are then sorted by the total revenue in descending order.

Results Messages

| | Category | TotalProducts | TotalRevenue |
|---|---|---|---|
| 1 | NULL | 23 | 7970.00 |
| 2 | Electronics | 8 | 600.00 |
| 3 | Footwear | 3 | 300.00 |
| 4 | Accessories | 3 | 100.00 |
| 5 | Books | 5 | 40.00 |
| 6 | Apparel | 4 | 25.00 |

23.

**Explanation:**
In this query, I extracted details of products within the 'Electronics' category. I filtered the products to show only those that have been ordered in quantities between 5 and 10, or haven't been ordered at all (represented by NULL). The products are then sorted by their total revenue in descending order.

Results Messages

| | Productid | ProductName | Category | OrderQuantity | OrderPrice | TotalRevenue |
|---|---|---|---|---|---|---|
| 1 | 7 | MOUSE | Electronics | 10 | 1000.00 | 10000.00 |
| 2 | 6 | FANS | Electronics | 8 | 800.00 | 6400.00 |
| 3 | 4 | Keyboard | Electronics | 6 | 1000.00 | 6000.00 |
| 4 | 5 | MIC | Electronics | 5 | 200.00 | 1000.00 |
| 5 | 1 | Laptop | Electronics | 0 | 1000.00 | 0.00 |
| 6 | 2 | Headphones | Electronics | 0 | 200.00 | 0.00 |
| 7 | 3 | Smartphone | Electronics | 0 | 800.00 | 0.00 |

24.

**Explanation:**
This query determines the total number of products sold for each product category. I accessed data from the PRODUCT and ORDERITEM tables and filtered to only include categories where more than 10 products have been sold. The results are then sorted by the total number of products sold in descending order.

Results Messages

| | Category | TotalProductsSold |
|---|---|---|
| 1 | NULL | 34 |

25.

**Explanation:**
In this query, I retrieved customer details and counted the total unique orders each customer has placed. By joining the CUSTOMER and ORDER_ tables, I aggregated the data to determine the total orders placed by each customer. The HAVING clause filters the results to only show customers who have placed more than 5 unique orders. Finally, the results are sorted by the total orders placed in descending order.

Results Messages

| | Custid | Fname | Lname | TotalOrdersPlaced |
|---|---|---|---|---|
| 1 | 4 | David | White | 11 |
| 2 | 2 | Bob | Smith | 10 |

## REPO :

PoptartAk47u/Dante1337.