**Amazon Analysis Using MySQL**

**Amazon India is analysing customer and sales data from Amazon Brazil to identify key trends that can be leveraged in the Indian market. The main objective of this analysis is to know customer behaviours, product preferences, and payment patterns to improve customer experience and capture new market opportunities in India.**

**This project uses multiple tables: Customers, Orders, Order Items, Product, Sellers, and Payments. Through SQL queries, various business-critical questions and providing actionable insights are shared.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Analysis1**

**Que1:***To simplify its financial reports, Amazon India needs to standardize payment values. Round the average payment values to an integer (no decimal) for each payment type and display the results sorted in ascending order.*

**Approach:**

1. **Identifying Relevant Tables and Columns:**
   * **Table: payments**
   * **Columns: payment\_type, payment\_value**
2. **Calculating Average Payment Value:**
   * Used the **AVG()** aggregate function on **payment\_value**
   * Grouped the results by payment\_type to get the average for each type
3. **Rounding the Averages:**
   * Used **the ROUND()** function to round the average payment values to the nearest integer.
4. **Sort the Results:**
   * Ordered the final results in ascending order based on the rounded average payment.

**SQL Query:**

**select payment\_type,**

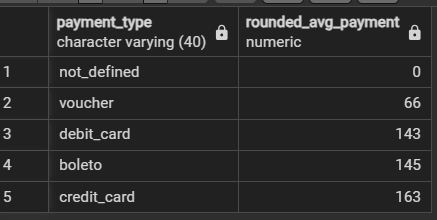
**Round(avg(payment\_value),0) as**

**rounded\_avg\_payment from amazon\_brazil.payments**

**group by payment\_type**

**order by rounded\_avg\_payment asc;**

**Output:**

****

**Recommendations:**

1. **Focus on Popular Payment Methods:**
   * Try making the payment methods like **credit\_card** and **boleto** optimised, as their average payment amounts are higher.
   * Introduce targeted promotions for customers using these methods to make higher revenue.
2. **Improve Use of Underperforming Methods:**
   * For methods like **debit\_card**, open opportunities for growth by giving promotional discounts or improving transaction processes.

**Que2:***To refine its payment strategy, Amazon India wants to know the distribution of orders by payment type. Calculate the percentage of total orders for each payment type, rounded to one decimal place, and display them in descending order.*

**Approach:**

1. **Calculating Total Number of Orders:**
   * Determined the total count of all orders across payment types.
2. **Calculating Orders per Payment Type:**
   * Used **COUNT(\*)** to get the number of orders for each payment\_type
3. **Calculating Percentage of Orders:**
   * Divided the count of orders per payment type by the total number of orders, multiplied by 100, and rounded to one decimal place.
4. **Sorting Results:**
   * Ordered the results in descending order of percentage to highlight the most popular payment methods.

**SQL Query:**

**SELECT payment\_type,**

**ROUND(CAST(COUNT(DISTINCT order\_id) AS NUMERIC) \* 100.0 /**

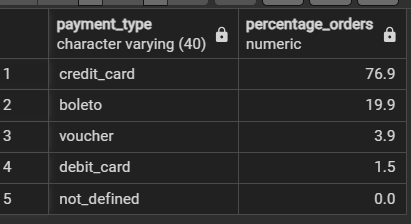
**(SELECT COUNT(DISTINCT order\_id) FROM amazon\_brazil.payments), 1)**

**AS percentage\_orders FROM amazon\_brazil.payments**

**GROUP BY payment\_type**

**ORDER BY percentage\_orders DESC;**

**Output:**

****

**Recommendations:**

1. **Maintain efficiency of Popular Payment Methods**
   * Since **credit\_card** is very popular, ensure this payment method is highly reliable, secure, and easy to use.
   * Consider cashback or reward or loyalty programs for credit card users to incentivize more purchases.
2. **Improve Less Popular Payment Methods:**
   * Explore ways to make **boleto**, **vouche**r, and **debit\_card** more attractive by offering discounts or simplifying the payment process.
   * Also, investigate reasons why there is low usage of certain methods and address any barriers.

**Que3:***Amazon India seeks to create targeted promotions for products within specific price ranges. Identify all products priced between 100 and 500 BRL that contain the word 'Smart' in their name. Display these products, sorted by price in descending order.*

**Approach:**

1. **Filtering Products by Price Range**:
   * Select products priced between **100 and 500 BRL** using the BETWEEN clause.
2. **Searching for the Word 'Smart'**:
   * Use LIKE '%smart%' to filter product names that contain the word 'Smart', making it case-insensitive.
3. **Join Orders and Products**:
   * Perform an **INNER JOIN** between the Order Items and Product tables on the product\_id column to link product details with order prices.
4. **Sort Results**:
   * Sort the filtered products in **descending order** by price to display the most expensive products first.

**SQL Query:**

**select o.product\_id, o.price**

**from amazon\_brazil.order\_items as o**

**join amazon\_brazil.product as p**

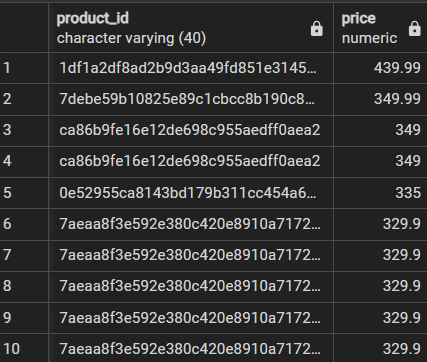
**on o.product\_id=p.product\_id**

**where o.price between 100 and 500**

**and p.product\_category\_name like '%smart%'**

**order by o.price Desc;**

**Output:**



**Recommendations:**

1. **Promotional Campaign**:

* Focus promotions on **"Smart"** products in the specified price range to attract cost-conscious customers seeking smart features.

2. **Highlighting Mid-Range Products**:

* Promote these products as affordable yet feature-rich options for customers.

**Que4:***To identify seasonal sales patterns, Amazon India needs to focus on the most successful months. Determine the top 3 months with the highest total sales value, rounded to the nearest integer.*

**Approach:**

1. **Extracting Month from Purchase Date**:

* Using **EXTRACT(MONTH FROM order\_purchase\_timestamp)** to retrieve the month for each order.

2. **Calculating Total Sales per Month**:

* Aggregating **SUM(price)** to compute total sales for each month.

1. **Rounding Sales Value:**

* Applying **ROUND()** to round total sales to the nearest integer.

4. **Sorting and Filtering Top 3**:

* Ordering results in descending order of sales and using **LIMIT 3** to display the top 3 months.

**SQL Query:**

**select extract (month from order\_purchase\_timestamp) as month,**

**round(sum(oi.price)) as total\_sales**

**from amazon\_brazil.orders as o**

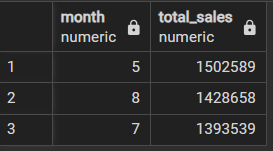
**join amazon\_brazil.order\_items as oi**

**on o.order\_id=oi.order\_id**

**group by extract (month from order\_purchase\_timestamp)**

**order by total\_sales desc limit 3;**

**Output:**



**Recommendations:**

1. **Target Promotions for Peak Months**:

* Allocate more **marketing** and promotion budgets for the top-performing months to maximize revenue.

1. **Analyze Trends**:

* Investigate **customer behavior** in these months to understand what drives high sales (e.g., festivals, holidays, or sales events)

**Que5:***Amazon India is interested in product categories with significant price variations. Find categories where the difference between the maximum and minimum product prices is greater than 500 BRL.*

**Approach:**

1. **Joining Products with Order Items**:

* Joining Product and Order Items tables on product\_id to associate product categories with prices.

2. **Calculating Price Difference**:

* Using **MAX(oi.price) - MIN(oi.price)** to compute the price variation for each category.

3. **Filtering Categories**:

* Applying **HAVING** clause to select categories with a price difference greater than **500 BRL**.

4. **Sorting Results**:

* Ordering categories by **price\_difference** in descending order

**SQL Query:**

**select product\_category\_name ,**

**max(oi.price)- min(oi.price) as price\_difference from**

**amazon\_brazil.product p**

**join amazon\_brazil.order\_items oi**

**on p.product\_id=oi.product\_id**

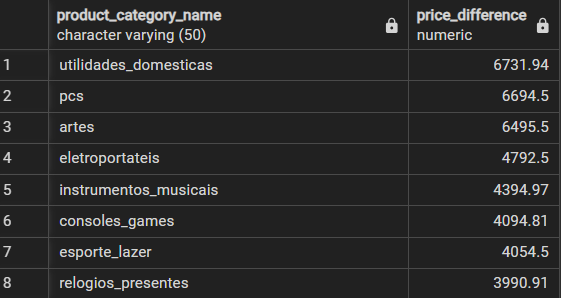
**group by p.product\_category\_name**

**having max(oi.price) - min(oi.price)> 500**

**order by price\_difference desc;**

**Output:**The analysis revealed significant price variations across product categories, with the top categories showing a difference greater than 500 BRL. Key highlights:  
Top Categories:

* Utilidades Domésticas: 6731.94 BRL
* PCS: 6694.5 BRL
* Artes: 6495.5 BRL



**Recommendations:**

### **1.Focus on High-Variation Categories:**

### Prioritize categories like "**utilidades\_domesticas**," "**pcs**," and "**artes**" for promotions as they show the highest price differences.

### Highlight the premium and budget options within these categories to attract a wide audience.

### **2.Enhance Product Offerings:**

### Expand the range of products in categories like "eletroportateis" and "instrumentos\_musicais" to cover diverse price segments.

**Que6:***To enhance the customer experience, Amazon India wants to find which payment types have the most consistent transaction amounts. Identify the payment types with the least variance in transaction amounts, sorting by the smallest standard deviation first.*

**Approach:**

1. **Calculating Standard Deviation**:

* Using **STDDEV**(payment\_value) to calculate the standard deviation of transaction amounts for each payment\_type.

2. **Grouping by Payment Type**:

* Grouping the results by payment\_type to calculate the standard deviation for each type separately.

3. **Sorting Results**:

* Ordering the results in **ascending order** of standard deviation to identify the most consistent payment types first.

4. **Rounding the Output**:

* Applying **ROUND()** to round the standard deviation to **2 decimal places** for clarity.

**SQL Query:**

**select payment\_type,**

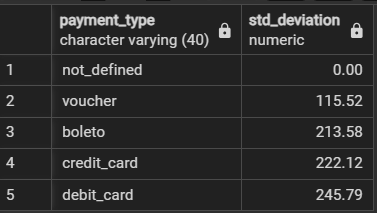
**round(StdDev(payment\_value),2)as std\_deviation**

**from amazon\_brazil.payments**

**group by payment\_type**

**order by std\_deviation asc;**

**Output:**

****

**Recommendations:**

**1. Promote Stable Payment Methods:**

* Highlight payment methods with low variance, such as **voucher**, as reliable and consistent for transactions.

**2. Address Issues with High-Variance Methods:**

* Investigate reasons for high variance in methods like **debit\_card** and ensure better user guidance during transactions.

**Que7:***Amazon India wants to identify products that may have incomplete name in order to fix it from their end. Retrieve the list of products where the product category name is missing or contains only a single character.*

**Approach:**

1. **Filtering Missing Names**:

* Using **IS NULL** to identify products where the product\_category\_name is missing.

2. **Filtering Single-Character Names**:

* Using **LENGTH(product\_category\_name) = 1** to find product names with only a single character.

3. **Combining Filters**:

* Using **OR** to combine both conditions and retrieve products matching either case.

4. **Retrieving Relevant Columns**:

* Selecting product\_id and product\_category\_name for better identification and resolution.

**SQL Query:**

**SELECT product\_id, product\_category\_name**

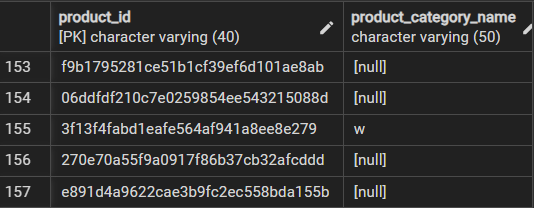
**FROM amazon\_brazil.product**

**WHERE**

**product\_category\_name IS NULL**

**OR LENGTH(product\_category\_name) = 1;**

**Output:**

****

**Recommendations:**

1. **Fix Missing Category Names:**

* Update the **NULL** entries with appropriate category names to maintain data completeness.

2. **Resolve Single-Character Names:**

* Review products with **single-character** names and replace them with full category descriptions.

3. **Improve Data Validation:**

* Implement **data validation rules** to prevent incomplete or incorrect entries during data entry or import processes.

**Analysis2**

**Que1:***Amazon India wants to understand which payment types are most popular across different order value segments (e.g., low, medium, high). Segment order values into three ranges: orders less than 200 BRL, between 200 and 1000 BRL, and over 1000 BRL. Calculate the count of each payment type within these ranges and display the results in descending order of count.*

**Approach:**

1. **Segmenting Order Values**:
   * Used a **CASE** statement to categorize **payment\_value** into three segments: low, medium, and high.
2. **Counting Payment Types**:
   * Used **COUNT(\*)** to calculate the total number of transactions for each payment\_type within each segment.
3. **Grouping by Payment Type and Segment**:
   * Grouped data by **payment\_type** and the segment created in the CASE statement.
4. **Sorting Results**:
   * Ordered results by the count of payment types **(count\_payment)** in descending order.

**SQL Query:**

**select payment\_type, count(\*) as count\_payment,**

**CASE**

**when payment\_value < 200 THEN 'low'**

**when payment\_value Between 200 AND 1000 THEN 'medium'**

**when payment\_value > 100 THEN 'high'**

**END AS segment from amazon\_brazil.payments**

**group by payment\_type, CASE**

**when payment\_value < 200 THEN 'low'**

**when payment\_value Between 200 AND 1000 THEN 'medium'**

**when payment\_value > 100 THEN 'high'**

**END**

**order by count\_payment;**

**Output:**

****

**Recommendations:**

1. **Boost High-Segment Payments**:

* Develop strategies to increase high-value transactions by offering premium services, financing options, or loyalty programs.

2. **Uniform Payment Experience**:

* Ensure all payment types are smooth and reliable across value segments to cater to diverse customer needs.

**Que2:***Amazon India wants to analyse the price range and average price for each product category. Calculate the minimum, maximum, and average price for each category, and list them in descending order by the average price.*

**Approach:**

1. **Joining Products with Order Items**:

* Using a JOIN between the **Product** and **Order Items** tables on **product\_id** to link categories with prices.

2. **Calculating Price Statistics**:

* Applying aggregate functions:
  + **MIN**(oi.price) for the lowest price.
  + **MAX**(oi.price) for the highest price.
  + **AVG**(oi.price) for the average price of products in each category.

3. **Rounding Average Price**:

* Using **ROUND()** to format the average price to **2 decimal places** for clarity.

4. **Grouping and Sorting**:

* Grouping by **product\_category\_name** to calculate statistics per category.
* Sorting the results in descending order of **avg\_price.**

**SQL Query:**

**SELECT product\_category\_name,**

**MIN(oi.price) AS min\_price,**

**MAX(oi.price) AS max\_price,**

**ROUND(AVG(oi.price), 2) AS avg\_price**

**FROM amazon\_brazil.product p**

**join amazon\_brazil.order\_items oi on p.product\_id=oi.product\_id**

**GROUP BY product\_category\_name**

**ORDER BY avg\_price DESC;**

**Output:**

****

**Recommendations:**

1. Focus on categories with high average prices to maximize revenue opportunities.

2. Highlight budget-friendly categories for customers seeking affordable options.

3. Maintain a balanced product mix to cater to different customer segments.

**Que3:***Amazon India wants to identify the customers who have placed multiple orders over time. Find all customers with more than one order, and display their customer unique IDs along with the total number of orders they have placed.*

**Approach:**

1. **Joining Orders with Customers**:
   * Using a **JOIN** between the **Orders** and **Customer** tables on customer\_id to associate orders with unique customer IDs.
2. **Counting Total Orders**:
   * Using **COUNT**(order\_id) to calculate the total number of orders placed by each customer.
3. **Filtering Customers with Multiple Orders**:
   * Applying **HAVING COUNT**(order\_id) > 1 to include only customers who have placed more than one order.
4. **Grouping by Unique ID**:
   * Grouping by **customer\_unique\_id** to calculate the total orders per customer.

**SQL Query:**

**SELECT customer\_unique\_id,**

**COUNT(order\_id) AS total\_orders**

**FROM amazon\_brazil.orders o**

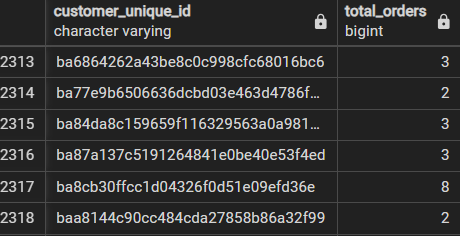
**JOIN amazon\_brazil.customer c**

**ON o.customer\_id = c.customer\_id**

**GROUP BY customer\_unique\_id**

**HAVING COUNT(order\_id) > 1;**

**Output:**

****

**Recommendations:**

1. **Identify Loyal Customers**:

* Use the list of repeat customers to offer loyalty programs and personalized incentives.

2. **Target High-Frequency Buyers**:

* Focus marketing efforts on customers with higher order counts for cross-selling and upselling opportunities.

**3. Segment Repeat Buyers**:

* Group customers based on their purchase frequency for tailored promotions.

4. **Encourage Customer Retention**:

* Provide discounts or rewards to retain customers who make frequent purchases.

**Que4:***Amazon India wants to categorize customers into different types ('New – order qty. = 1’; 'Returning' –order qty. 2 to 4; 'Loyal' – order qty. >4) based on their purchase history. Use a temporary table to define these categories and join it with the customers table to update and display the customer types.*

**Approach:**

1. **Calculating Total Orders**:

* Use a **temporary table** (CustomerOrderCounts) to calculate the total number of orders for each customer\_id.

2. **Defining Categories**:

* Use a CASE statement to categorize customers as **New**, **Returning**, or **Loyal** based on their order count.

3. **Joining with Customers**:

* Join the categorized data with the customers table for seamless integration.

4. **Sorting Results**:

* Order the output by **customer\_id** for better readability.

**SQL Query:**

**WITH CustomerOrderCounts AS (**

**SELECT customer\_id,**

**COUNT(order\_id) AS total\_orders**

**FROM amazon\_brazil.orders**

**GROUP BY customer\_id )**

**SELECT customer\_id,**

**CASE**

**WHEN total\_orders = 1 THEN 'New'**

**WHEN total\_orders BETWEEN 2 AND 4 THEN 'Returning'**

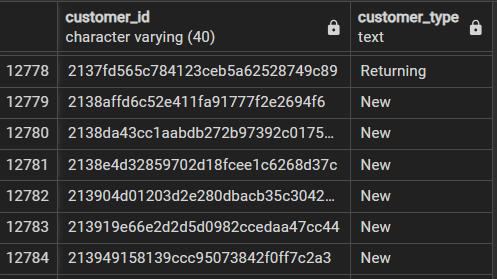
**ELSE 'Loyal'**

**END AS customer\_type**

**FROM CustomerOrderCounts**

**ORDER BY customer\_id;**

**Output:**

****

**Recommendations:**

1. **Target New Customers**:

* Offer onboarding incentives like discounts or free delivery to encourage repeat purchases.

**2**. **Retain Returning Customers**:

* Provide personalized offers or rewards for returning customers to boost their engagement.

**3**. **Reward Loyal Customers**:

* Implement loyalty programs to reward frequent buyers and encourage continued spending.

**Que5:***Amazon India wants to know which product categories generate the most revenue. Use joins between the tables to calculate the total revenue for each product category. Display the top 5 categories.*

**Approach:**

1. **Join Products with Order Items**:

* Use a JOIN between the **Product** and **Order Items** tables on product\_id to link product categories with their revenue.

2. **Calculate Total Revenue**:

* Use **SUM**(oi.price) to calculate the total revenue for each product category.

3. **Group by Product Category**:

* Group by product\_category\_name to aggregate the revenue for each category.

4. **Sort and Limit Results**:

* Order the results in descending order of total revenue and use **LIMIT 5** to display the top 5 categories.

**SQL Query:**

**select p.product\_category\_name, sum(oi.price)**

**as total\_revenue from amazon\_brazil.product p**

**join amazon\_brazil.order\_items oi**

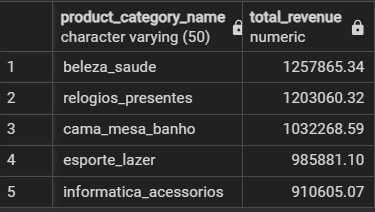
**on p.product\_id=oi.product\_id**

**group by p.product\_category\_name**

**order by total\_revenue desc**

**limit 5;**

**Output:**

****

**Recommendations:**

1. **Focus on High-Revenue Categories**:

* Allocate marketing budgets and resources to the top-performing categories.

2. **Optimize Inventory**:

* Ensure sufficient stock of high-revenue categories to prevent potential revenue loss.

3. **Cross-Sell Opportunities**:

* Bundle products from top categories with complementary products to boost sales further.

.

**Analysis 3**

**Que1:  
*The marketing team wants to compare the total sales between different seasons.****Use a subquery to calculate total sales for each season (Spring, Summer, Autumn, Winter) based on order purchase dates, and display the results. Spring is in the months of March, April and May. Summer is from June to August and Autumn is between September and November and rest months are Winter.*

**Approach:**

1. **Identify Seasons**:
   * Use a CASE statement to classify orders into Spring, Summer, Autumn, and Winter based on the month extracted from **order\_purchase\_timestamp**.
2. **Join Orders and Order Items**:
   * Use an INNER JOIN to combine data from **Orders** and **Order Items** tables using order\_id.
3. **Calculate Total Sales**:
   * Use **SUM** (oi.price) to calculate total sales for each season.
4. **Group and Sort**:
   * Group data by **season** and sort the output by the seasonal order.

**SQL Query:**

**SELECT**

**CASE**

**WHEN EXTRACT(MONTH FROM o.order\_purchase\_timestamp) IN (3, 4, 5) THEN 'Spring'**

**WHEN EXTRACT (MONTH FROM o.order\_purchase\_timestamp) IN (6, 7, 8) THEN 'Summer'**

**WHEN EXTRACT(MONTH FROM o.order\_purchase\_timestamp) IN (9, 10, 11) THEN 'Autumn'**

**ELSE 'Winter'**

**END AS season,**

**SUM(oi.price) AS total\_sales**

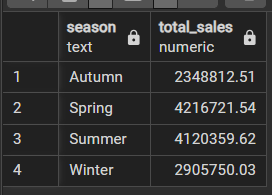
**FROM amazon\_brazil.orders o**

**INNER JOIN amazon\_brazil.order\_items oi**

**ON o.order\_id = oi.order\_id**

**GROUP BY season ORDER BY season;**

**Output:**

****

**Recommendations:**

**1.Allocate Seasonal Marketing Resources**:

* Use the sales data to plan promotions during high-revenue seasons.

**2. Boost Low-Season Sales:**

* Introduce discounts or exclusive deals to drive sales during low-performing seasons.

**3. Targeted Advertising Campaigns:**

* + Focus advertising efforts on customer preferences for specific seasons.

**Que2:***The inventory team is interested in identifying products that have sales volumes above the overall average. Write a query that uses a subquery to filter products with a total quantity sold above the average quantity.*

**Approach:**

1. **Calculating Total Quantity Sold Per Product**:

* Using **COUNT** (order\_item\_id) grouped by product\_id to calculate the total quantity sold for each product.

2. **Calculating Overall Average Quantity**:

* Using a **subquery** to calculate the **average total quantity sold** across all products.

3. **Filtering Above-Average Products**:

* Using a **WHERE** clause to include only products where the total quantity sold is greater than the overall average.

4. **Organizing Results**:

* Selecting and displaying product\_id along with total\_quantity\_sold for **above-average products**

**SQL Query:**

**SELECT product\_id, total\_quantity\_sold**

**FROM (**

**SELECT product\_id, COUNT(order\_item\_id) AS total\_quantity\_sold**

**FROM amazon\_brazil.order\_items**

**GROUP BY product\_id)**

**WHERE total\_quantity\_sold > (**

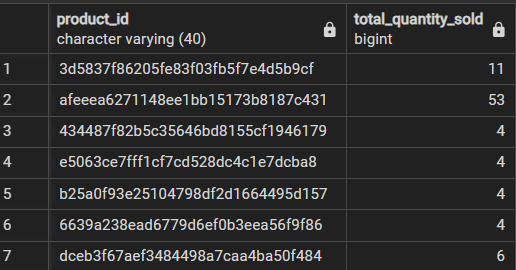
**SELECT AVG(total\_quantity\_sold)**

**FROM ( SELECT COUNT(order\_item\_id) AS total\_quantity\_sold**

**FROM amazon\_brazil.order\_items**

**GROUP BY product\_id ));**

**Output:**

****

**Recommendations:**

**1.** **Prioritize High-Selling Products**:

* Focus on replenishing stock for above-average products to meet demand.

**2.** **Analyze Trends for Top Products**:

* Investigate why these products are performing better to replicate success for other products.

**3. Allocate Marketing Resources**:

* Direct marketing campaigns toward high-performing products to maximize sales.

**Que3:***To understand seasonal sales patterns, the finance team is analysing the monthly revenue trends over the past year (year 2018). Run a query to calculate total revenue generated each month and identify periods of peak and low sales. Export the data to Excel and create a graph to visually represent revenue changes across the months.*

**Approach:**

1. **Filtering Data for 2018**:

* Used **EXTRACT** (YEAR FROM o.order\_purchase\_timestamp) to filter orders from the year 2018.

**2. Extracting Month**:

* Applied EXTRACT (MONTH FROM o.order\_purchase\_timestamp) to identify the month for each order.

**3. Calculating Total Revenue:**

* Used SUM (oi.price) to calculate total revenue for each month.

**4. Grouping and Sorting Data:**

* Grouped by month to aggregate revenue and ordered the results by month for chronological representation.

**5. Exporting and Visualization:**

* Exported the results to Excel and created a graph to represent monthly revenue changes.

**SQL Query:**

**SELECT EXTRACT(MONTH FROM o.order\_purchase\_timestamp) AS month,**

**SUM(oi.price) AS total\_revenue**

**FROm amazon\_brazil.orders o**

**JOIN amazon\_brazil.order\_items oi**

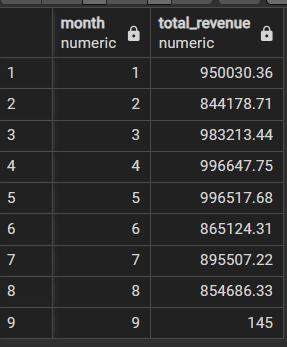
**ON o.order\_id = oi.order\_id**

**WHERE EXTRACT(YEAR FROM o.order\_purchase\_timestamp) = 2018**

**GROUP BY month**

**ORDER BY month;**

**Output:**

****

**Graphical Visualization:**

**Recommendations:**

**1. Seasonal Planning**:

* + Use the revenue data to plan inventory and promotions ahead of seasonal trends.

**2.Analyze Customer Behaviour**:

* + Study customer preferences during peak months to tailor future strategies.

**Que4:**

*A loyalty program is being designed for Amazon India. Create a segmentation based on purchase frequency: ‘Occasional’ for customers with 1-2 orders, ‘Regular’ for 3-5 orders, and ‘Loyal’ for more than 5 orders. Use a CTE to classify customers and their count and generate a chart in Excel to show the proportion of each segment.*

**Approach:**

**1. Calculating Order Frequency:**

* Group by customer\_id and count the number of orders for each customer.

**2. Classifying Customers:**

* Use a CASE statement in SQL or equivalent logic to classify customers as **Occasional**, **Regular**, or **Loyal** based on the number of orders.

**3. Aggregating Segment Counts:**

* Count the number of customers in each segment for proportion analysis.

4. **Export and Visualize**:

* Export the results to Excel and create a chart to visually represent the proportion of customer segments.

**SQL Query:**

**WITH purchase\_frequency AS (**

**SELECT customer\_id,**

**COUNT(order\_id) AS order\_count**

**FROM amazon\_brazil.orders**

**GROUP BY customer\_id)**

**SELECT customer\_id,**

**CASE**

**WHEN order\_count BETWEEN 1 AND 2 THEN 'Occasional'**

**WHEN order\_count BETWEEN 3 AND 5 THEN 'Regular'**

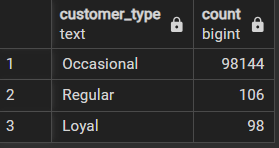
**ELSE 'Loyal'**

**END AS customer\_type**

**FROM purchase\_frequency**

**ORDER BY order\_count;**

**Output:**

****

**Graphical Visualization:**

**Recommendations:**

**1. Engage Occasional Customers:**

* Run personalized campaigns to convert occasional buyers into regular customers.

**2. Reward Loyal Customers:**

* Provide exclusive rewards for loyal customers to retain them and encourage long-term engagement.

**Que5:**

*Amazon wants to identify high-value customers to target for an exclusive rewards program. You are required to rank customers based on their average order value (avg\_order\_value) to find the top 20 customers.*

**Approach:**

**1.** **Calculating Average Order Value**:

* Use **AVG** (price) to calculate the average order value for each customer by combining data from the **Orders** and **Order Items** tables.

**2.** **Ranking Customers**:

* Use the **RANK ()** window function to assign a rank to customers based on their avg\_order\_value in descending order.

**3.** **Limit to Top 20 Customers**:

* Use **LIMIT 20** to retrieve only the top 20 high-value customers.

**4.** **Organize Results**:

* Select and display customer\_id, avg\_order\_value, and customer\_rank for the top 20 customers.

**SQL Query:**

**WITH CustomerOrderValue AS (**

**SELECT customer\_id,**

**AVG (price) AS avg\_order\_value**

**FROM amazon\_brazil.orders o**

**JOIN amazon\_brazil.order\_items oi**

**ON o.order\_id = oi.order\_id**

**GROUP BY customer\_id )**

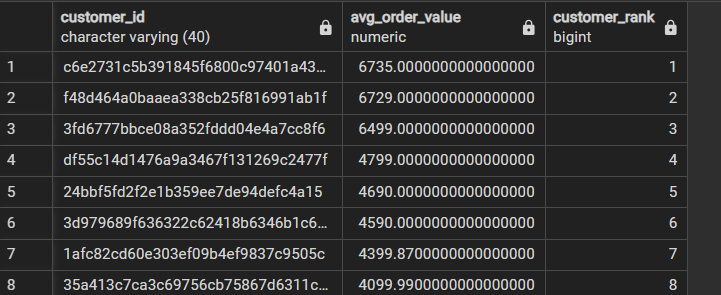
**SELECT customer\_id, avg\_order\_value,**

**RANK () OVER (ORDER BY avg\_order\_value DESC)**

**AS customer\_rank FROM CustomerOrderValue**

**ORDER BY avg\_order\_value DESC LIMIT 20;**

**Output:**

****

**Recommendations:**

1. **Reward High-Value Customers**:

* Create exclusive rewards and loyalty benefits to retain top customers.

2. **Offer Personalized Experiences**:

* Provide personalized offers or premium support to high-value customers to enhance satisfaction.

3. **Monitor Spending Patterns**:

* Analyze the purchasing behavior of these customers to tailor marketing strategies.

**Que6:**

*Amazon wants to analyze sales growth trends for its key products over their lifecycle. Calculate monthly cumulative sales for each product from the date of its first sale. Use a recursive CTE to compute the cumulative sales (total\_sales) for each product month by month.*

**Approach:**

**1.** **Calculate Monthly Sales**:

* Use a **Common Table Expression (CTE)** to calculate the total sales (monthly\_sales) for each product for each month by grouping data by product\_id and month.

**2.** **Extract and Format Month**:

* Use **DATE\_TRUNC**('month', order\_purchase\_timestamp) to extract the month from the purchase date and format it as YYYY-MM for readability.

**3.** **Compute Cumulative Sales**:

* Use a **window function (SUM() OVER)** to calculate cumulative sales (total\_sales) for each product, ordered by month.

**4.** **Organize Results**:

* Select and display product\_id, sale\_month, and total\_sales for each product month by month.

**SQL Query:**

**WITH MonthlySales AS (**

**SELECT product\_id,**

**TO\_CHAR(DATE\_TRUNC('month', o.order\_purchase\_timestamp), 'YYYY-MM') AS sale\_month,**

**SUM(oi.price) AS monthly\_sales**

**FROM amazon\_brazil.orders o**

**JOIN amazon\_brazil.order\_items oi ON o.order\_id = oi.order\_id**

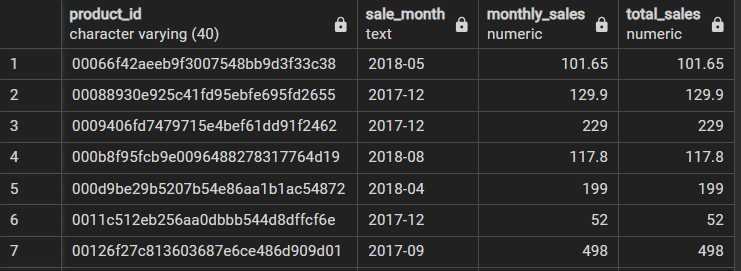
**GROUP BY product\_id, sale\_month )**

**SELECT product\_id, sale\_month, monthly\_sales,**

**SUM(monthly\_sales) OVER (PARTITION BY product\_id ORDER BY sale\_month) AS total\_sales**

**FROM MonthlySales ORDER BY product\_id, sale\_month;**

**Output:**

****

**Recommendations:**

**1.** I**dentify Growth Trends**:

* Use cumulative sales data to track the lifecycle and growth trends of key products.

**2.** **Focus on High-Growth Products**:

* Allocate resources and marketing efforts to products showing consistent monthly growth.

**3.** **Manage Inventory Effectively**:

* Use sales trends to predict demand and optimize inventory management for key products.

**Que7:**

***T****o understand how different payment methods affect monthly sales growth, Amazon wants to compute the total sales for each payment method and calculate the month-over-month growth rate for the past year (year 2018). Write query to first calculate total monthly sales for each payment method, then compute the percentage change from the previous month.*

**Approach:**

1. **Filtering Data for 2018**:
   * Use **EXTRACT** (YEAR FROM order\_purchase\_timestamp) to select orders from 2018.
2. **Calculating Monthly Sales**:
   * Group data by payment\_type and sale\_month to calculate monthly\_total sales using SUM(oi.price).
3. **Computing Month-over-Month Growth**:
   * Use the **LAG ()** window function to access the sales of the previous month for each payment\_type.
   * Calculate the percentage change using the formula:  
     **((current\_month\_sales - previous\_month\_sales) / previous\_month\_sales) \* 100.**
4. **Organizing Results**:
   * Display payment\_type, sale\_month, monthly\_total, and monthly\_change, ordered by payment\_type and sale\_month.

**SQL Query:**

**WITH MonthlyPaymentSales AS (**

**SELECT p.payment\_type,**

**TO\_CHAR(DATE\_TRUNC('month', o.order\_purchase\_timestamp), 'YYYY-MM') AS sale\_month,**

**SUM (oi.price) AS monthly\_total**

**FROM amazon\_brazil.orders o**

**JOIN amazon\_brazil.order\_items oi ON o.order\_id = oi.order\_id**

**JOIN amazon\_brazil.payments p ON o.order\_id = p.order\_id**

**WHERE EXTRACT(YEAR FROM o.order\_purchase\_timestamp) = 2018**

**GROUP BY p.payment\_type, sale\_month )**

**SELECT payment\_type, sale\_month, monthly\_total,**

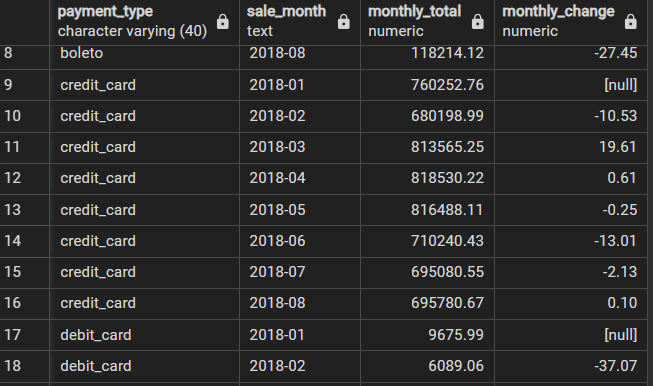
**ROUND (((monthly\_total - LAG(monthly\_total) OVER (PARTITION BY payment\_type ORDER BY sale\_month)) /**

**LAG(monthly\_total) OVER (PARTITION BY payment\_type ORDER BY sale\_month)) \* 100, 2) AS monthly\_change**

**FROM MonthlyPaymentSales**

**ORDER BY payment\_type, sale\_month;**

**Output:**

****

**Recommendations:**

**1**. **Enhance Customer Experience for Growing Methods**:

* Focus on optimizing and promoting payment methods with consistent positive growth.

**2**. **Investigate Declining Trends**:

* Analyze reasons for decline in monthly sales for certain payment methods and address barriers.