**EXECUTIVE SUMMARY** 

Amazon, a global leader in e-commerce, has successfully established its presence in multiple international markets, including the U.S., Europe, and Asia. In Brazil, Amazon has built a strong ecosystem by connecting small and medium-sized businesses with millions of customers. Given the similarities between Brazil and India—such as large populations, diverse consumer bases, and growing digital adoption—there is a strategic opportunity to replicate this success in the Indian market.

This project aims to analyze Amazon Brazil's customer and sales data to identify trends, consumer behavior, and payment preferences that can be leveraged in India. The insights derived from this analysis will help Amazon India make data-driven decisions to enhance customer experience, optimize financial operations, and drive market growth.

**Amazon India Market Analysis Report**

**Project Context:** Amazon India is analyzing customer and sales data from Amazon Brazil to identify key trends that can be leveraged in the Indian market. The objective is to understand customer behavior, product preferences, and payment patterns to enhance customer experience and drive business growth in India.

This project uses multiple datasets, including **Customers, Orders, Order Items, Products, Sellers, and Payments.** Through SQL-based data analysis, key business insights are derived to inform strategic decision-making.

**Analysis 1**

**Q1 To simplify its financial reports, Amazon India needs to standardize payment values. Round the average payment values to integer (no decimal) for each payment type and display the results sorted in ascending order.**

**Approach:**

* 1. **.SELECT payment\_type, ROUND(AVG(payment\_value)) AS rounded\_avg\_payment Computes the average payment value for each payment type using AVG(payment\_value).**
  2. **Rounds the result to the nearest integer using ROUND(). FROM amazon\_brazil.payments**
  3. **Extracts data from the payments table in the amazon\_brazil schema. GROUP BY payment\_type**
  4. **Groups transactions by payment type (e.g., credit card, debit card, PayPal).ORDER BY rounded\_avg\_payment ASC**
  5. **Sorts the results in ascending order, first showing the lowest average payments.Bottom of Form**

|  |
| --- |
| Select payment\_type,round(avg(payment\_value)) as rounded\_avg\_payment from amazon\_brazil.payments group by payment\_type order by rounded\_avg\_payment |

**📢 Insights & Recommendations**

**Top Performer: Credit cards generally show higher and more consistent sales.**

1. **Seasonal Trends: Look for spikes in sales during festivals, like December.**
2. **Monthly Variations: Identify months with significant growth or decline.**
3. **Promotions: Offer credit card-specific promotions to boost sales.**
4. **Targeted Campaigns: Tailor marketing for festive seasons to maximize sales.**
5. **Optimization: Simplify the checkout process for popular payment methods like upi etc.**

|  |  |
| --- | --- |
| payment\_type | rounded\_avg\_payment |
| not\_defined | 0 |
| voucher | 66 |
| debit\_card | 143 |
| boleto | 145 |
| credit\_card | 163 |

Q 2 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. **COUNT(o.order\_id) \* 100.0** Counts the number of orders for each **payment type** and multiplies by 100.0 to convert to a percentage.
2. **/ (SELECT COUNT(\*) FROM amazon\_brazil.orders)** **Divides by the total number of orders** in the orders table to compute the percentage.
3. **ROUND(..., 1)** **Rounds the percentage** to **one decimal place** for better readability.
4. **JOIN amazon\_brazil.payments p ON o.order\_id = p.order\_id** **Joins the payments table** with orders to link payment types to orders.
5. **GROUP BY p.payment\_type** Groups results by **payment type** to calculate percentages separately.

|  |
| --- |
| select p.payment\_type, ROUND((COUNT(o.order\_id) \* 100.0) / (SELECT COUNT(\*) FROM amazon\_brazil.orders), 1) AS percentage\_orders FROM amazon\_brazil.orders as o JOIN amazon\_brazil.payments p ON o.order\_id = p.order\_id GROUP BY p.payment\_type ORDER BY percentage\_orders DESC; |

1. **ORDER BY percentage\_orders DESC** **Sorts results in descending order**, showing the most frequently used payment methods first

* **📢 Insights & Recommendations**
* Focus on Credit Card Payments: With 77.2% of orders, ensuring a smooth and secure credit card transaction process is crucial.
* Optimize Boleto Processing: Since 19.9% of customers use Boleto, simplifying payment confirmation could improve user experience.
* Encourage Voucher Usage: At 5.8%, vouchers can be promoted through cashback or discounts.
* Increase Debit Card Adoption: Only 1.5% of orders use debit cards, so targeted promotions may help drive adoption.
* Address "Not Defined" Payments: Although 0.0%, verifying data accuracy ensures proper classification of transactions.

|  |  |
| --- | --- |
| payment\_type | percentage\_orders |
| credit\_card | 77.2 |
| boleto | 19.9 |
| voucher | 5.8 |
| debit\_card | 1.5 |
| not\_defined | 0.0 |

Q3.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. **JOIN amazon\_brazil.products p ON p.product\_id = o.product\_id → Links products with order details.**
2. **JOIN amazon\_brazil.payments py ON o.order\_id = py.order\_id → Ensures products were purchased.**
3. **WHERE o.price BETWEEN 100 AND 500 → Filters products priced 100-500 BRL.**
4. **AND p.product\_category\_name ILIKE '%Smart%' → Finds "Smart" category items (case-insensitive).**
5. **ORDER BY o.price DESC → Sorts results by highest price first.**

|  |
| --- |
| **SELECT p.product\_id, o.price**  **FROM amazon\_brazil.products AS p**  **JOIN amazon\_brazil.order\_items AS o ON p.product\_id = o.product\_id**  **WHERE o.price BETWEEN 100 AND 500**  **AND p.product\_category\_name ILIKE '%Smart%' ORDER BY o.price DESC;** |

**📢 Insights & Recommendations**

1. **Deals: Action: Combine specific products with complementary items to create a combo.For instance, pair the high-end smart product (ID: 1df1a2df8ad2b9d3aa49fd851e3145ad) with relevant accessories.**
2. **Benefit: Enhances value perception and encourages bulk purchases.**
3. **Limited-Time Offers: Action: Launch time-sensitive discounts on any products.**
4. **Create urgency with promotional banners like "Grab it Now - Limited Time Only!"**
5. **Benefit: Drives immediate sales and taps into impulse buying behavior.**
6. **Combo Targeted Advertising: Run targeted ads on social media and email campaigns. Benefit: Increases ad relevance, improving conversion rates.**

|  |  |
| --- | --- |
| product\_id | price |
| 1df1a2df8ad2b9d3aa49fd851e3145ad | 439.99 |
| 7debe59b10825e89c1cbcc8b190c85e2 | 349.99 |
| ca86b9fe16e12de698c955aedff0aea2 | 349 |
| ca86b9fe16e12de698c955aedff0aea2 | 349 |
| 0e52955ca8143bd179b311cc454a6caa | 335 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| 7aeaa8f3e592e380c420e8910a717255 | 329.9 |
| d1b571cd58267d8cac8b2afd6e288bbc | 299.9 |
|  |  |

Q4.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. **Extracts Month → Gets the month from order\_purchase\_timestamp.**
2. **Joins Tables → Combines orders with order\_items using order\_id.**
3. **Calculates Sales → Sums up price and rounds the total.**

|  |
| --- |
| select extract(month from o.order\_purchase\_timestamp) as month, ROUND(sum(pr.price)) as total\_sales  from amazon\_brazil.orders as o  left JOIN amazon\_brazil.order\_items AS pr ON o.order\_id = pr.order\_id  group by extract(month from o.order\_purchase\_timestamp)  order by total\_sales DESC limit 3; |

1. **Groups & Sorts → Groups by month, orders by sales in descending order, and limits to the top 3 months.**

**📢 Insights & Recommendations**

* **Peak Sales in May: The highest sales (1,502,589) occurred in May. Focus on promotions, discounts, and inventory management during this period.**
* **Stable Sales in Mid-Year: Sales remained strong in July (1,393,539) and August (1,428,658). Consider maintaining marketing efforts to sustain performance.**
* **Seasonal Trend Analysis: Further analysis of external factors (holidays, festivals, or campaigns) can help optimize future sales strategies.**

|  |  |
| --- | --- |
| month | total\_sales |
| 5 | 1502589 |
| 8 | 1428658 |
| 7 | 1393539 |

Q5.**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. **Extracts Price Variation: (MAX(o.price) - MIN(o.price)) AS price\_difference calculates the price range per category.**
2. **Joins Tables: Links products with order\_items to access price data.**
3. **Filters High Variations: HAVING price\_difference > 500 keeps only categories with significant price differences.**

|  |
| --- |
| **select p.product\_category\_name, (max(o.price) -min(o.price)) as price\_difference**  **from amazon\_brazil.products as as p left join amazon\_brazil.order\_items as o on p.product\_id = o.product\_id**  **group by p.product\_category\_name having (max (o.price) - min(o.price)) > 500**  **order by price\_difference desc;** |

1. **Sorts Results: ORDER BY price\_difference DESC ranks categories by highest price gap.**

**📢 Insights & Recommendations**

1. **High Price Variations: Categories like "utilidades\_domesticas" (6,731.94), "pcs" (6,694.5), and "artes" (6,495.5) show major price fluctuations.**

**Optimize pricing to stay competitive and reduce buyer hesitation.**

1. **Moderate Price Variations: "Eletronicos" (2,466.51) and "telefonia" (2,423) have mid-range differences.**

**Adjust pricing based on demand trends and competitor positioning.**

1. **Low Price Variations :Stable pricing seen in "artigos\_de\_festas" (563.21), "audio" (584.09), and "bebidas" (617.0).**

**Use discounts or promotions to drive higher sales.**

1. **Strategic Actions: Optimize high-variation prices for better margins.**

**Apply competitive pricing for moderate-range categories. Leverage targeted discounts for stable-price categories to boost volume.**

|  |  |
| --- | --- |
| product\_category\_name | price\_difference |
| utilidades\_domesticas | 6731.94 |
| pcs | 6694.5 |
| artes | 6495.5 |
| eletroportateis | 4792.5 |
| instrumentos\_musicais | 4394.97 |
| consoles\_games | 4094.81 |
| esporte\_lazer | 4054.5 |

Q6.**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:**

**STDDEV(payment\_value) AS std\_deviation:**

1. **This calculates the standard deviation of payment\_value for each payment\_type.**
2. **Group by This groups the rows based on the payment\_type column**
   1. Order by This sorts the results in **ascending order** based on the standard deviation.

|  |
| --- |
| select payment\_type, STDDEV(payment\_value) as std\_deviation  from amazon\_brazil.payments group by payment\_type order by std\_deviation asc; |

**📢 Insights & Recommendations**

1. **High Variability in Payment Amounts: Debit cards (245.79)** and **credit cards (222.12)** show the highest standard deviation, indicating a wide range of transaction values.

Consider offering personalized offers or cashback incentives to stabilize spending patterns.

1. **Moderate Variability: Boleto (213.58)** transactions have a slightly lower deviation, suggesting less fluctuation but still significant differences in payment amounts. Ensuring flexible payment options could improve user experience.
2. **Low Variability: Voucher (115.52)** has the least variation among defined payment types, indicating a more consistent spending behavior. Expanding voucher-based promotions may encourage repeat purchases.
   1. Monitor high-deviation payment methods to identify transaction trends.
3. Offer structured discounts or installment plans for credit and debit card users to encourage consistent spending.

|  |  |
| --- | --- |
| payment\_type | std\_deviation |
| not\_defined | 0 |
| voucher | 115.5191854 |
| boleto | 213.5810615 |
| credit\_card | 222.1193107 |
| debit\_card | 245.793401 |

Q7.**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. **product\_id and product\_category\_name from the amazon\_brazil.products table**
2. **where the category name is either missing (NULL), empty (''), or only one character long**

|  |
| --- |
| **select product\_id, product\_category\_name**  **from amazon\_brazil.products**  **where product\_category\_name is null**  **or trim(product\_category\_name) = ''**  **or length(product\_category\_name) = 1;** |

**📢 Insights & Recommendations**

|  |  |
| --- | --- |
| product\_id | product\_category\_name |
| a41e356c76fab66334f36de622ecbd3a | NULL |
| d8dee61c2034d6d075997acef1870e9b | NULL |
| 56139431d72cd51f19eb9f7dae4d1617 | NULL |
| 46b48281eb6d663ced748f324108c733 | NULL |
| 5fb61f482620cb672f5e586bb132eae9 | NULL |
| e10758160da97891c2fdcbc35f0f031d | NULL |

Analysis - II

Q1.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. **Case stetment :Orders with payment\_value < 200 → "Low"Orders with payment\_value between 200 and 1000 → "Medium"Orders with payment\_value > 1000 → "High"**
2. **2 COUNT(\*) AS count: Counts the number of orders for each payment type in each segment.**
3. **Group3 data by order\_value\_segment and payment\_type.**
4. **Sorts the results by the number of orders in descending order**

|  |
| --- |
| case  when payment\_value < 200 then 'Low'  when payment\_value between 200 and 1000 then 'Medium'  else 'High'  end as order\_value\_segment,  payment\_type,  count(\*) as count from amazon\_brazil.payments group by  case  when payment\_value < 200 then 'Low'  when payment\_value between 200 AND 1000 THEN 'Medium'  ELSE 'High'  END,  payment\_type  ORDER By count DESC; |

**📢 Insights & Recommendations**

1. Focus on Credit Card Payments: Across all segments (low, medium, high), credit card payments dominate, especially in the low order value segment (60,548 transactions).
2. Strengthen partnerships with credit card providers for cashback offers, EMI options, or rewards to encourage higher spending.
3. Optimize Boleto Payment for Medium & Low Segments: Boleto (Brazilian bank slip) is the second most used payment method, particularly in the low (16,444 transactions) and medium (3,162 transactions) order value segments. Improving boleto processing times or offering small discounts for boleto payments could encourage more usage.
4. Leverage Vouchers for Promotions: 4. 5. Vouchers are significantly used in the low (5,476 transactions) and medium (286 transactions) segments, indicating their effectiveness for discounts and incentives. Expand voucher-based promotions for repeat customers and targeted marketing campaigns.
5. Address "Not Defined" Payment Types: There are a few instances of "not\_defined" in the low order value segment. Investigate whether these are system errors or unidentified payment methods and ensure proper classification.
6. Encourage Debit Card Usage for High-Value Orders: Debit card usage is very low in the high-value segment (only 15 transactions), suggesting that customers prefer credit cards for larger purchases. Offering secure, interest-free debit card installment plans could encourage its adoption

|  |  |  |
| --- | --- | --- |
| order\_value\_segment | payment\_type | count |
| Low | credit\_card | 60548 |
| Low | boleto | 16444 |
| Medium | credit\_card | 15303 |
| Low | voucher | 5476 |
| Medium | boleto | 3162 |
| Low | debit\_card | 1287 |
| High | credit\_card | 944 |
| Medium | voucher | 286 |
| Medium | debit\_card | 227 |
| High | boleto | 178 |
| High | debit\_card | 15 |
| High | voucher | 13 |
| Low | not\_defined | 3 |

Q2.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 descending order by the average price.

**Approach:**

1. **Joins order\_items (o) with products (p) using product\_id.**
2. **Groups by product\_category\_name.**
3. **Calculates min, max, and average price, rounding the average to 2 decimal places.**
4. **Orders results by average price in descending order to show the most expensive categories first.**

|  |
| --- |
| select  p.product\_category\_name,  min(o.price) as min\_price,  max(o.price) as max\_price,  ROUND(avg(o.price), 2) as avg\_price  from amazon\_brazil.order\_items as o  join amazon\_brazil.products as p  on o.product\_id = p.product\_id  group by p.product\_category\_name  order by avg\_price desc; |

**📢 Insights & Recommendations**

1. High-Priced Categories (₹400+) – PCs, home appliances, and electronics. Offer installment plans, premium bundles, and exclusive deals.
2. Mid-Range (₹100-400) – Smart products, gaming consoles, audio devices. Use seasonal discounts, trade-in offers, and cross-selling strategies.
3. Low-Priced (Below ₹100) – Books, clothing, fashion accessories. Leverage flash sales, BOGO deals, and targeted ads.
4. Data Issues – NULL and misclassified categories. Clean and standardize product taxonomy for better insights. Opportunities in Essentials (₹40-₹60) – Food, beverages, personal care. Introduce subscriptions and checkout add-ons to increase sales.

|  |  |  |  |
| --- | --- | --- | --- |
| product\_category\_name | min\_price | max\_price | avg\_price |
| pcs | 34.5 | 6729 | 1098.34 |
| portateis\_casa\_forno\_e\_cafe | 10.19 | 2899 | 624.29 |
| eletrodomesticos\_2 | 13.9 | 2350 | 476.12 |
| agro\_industria\_e\_comercio | 12.99 | 2990 | 341.66 |
| instrumentos\_musicais | 4.9 | 4399.87 | 281.62 |
| eletroportateis | 6.5 | 4799 | 280.78 |
| portateis\_cozinha\_e\_preparadores\_de\_alimentos | 17.42 | 1099 | 264.57 |
| telefonia\_fixa | 6 | 1790 | 225.69 |
| construcao\_ferramentas\_seguranca | 8.9 | 3099.9 | 208.99 |
| relogios\_presentes | 8.99 | 3999.9 | 200.91 |
| climatizacao | 10.9 | 1599 | 185.27 |
| moveis\_quarto | 6.9 | 650 | 183.75 |
| pc\_gamer | 129.99 | 239 | 171.77 |
| cool\_stuff | 7 | 3109.99 | 167.36 |
| moveis\_cozinha\_area\_de\_servico\_jantar\_e\_jardim | 9.6 | 1320 | 164.87 |
| moveis\_escritorio | 25 | 1189.9 | 162.01 |
| musica | 3.85 | 1165.97 | 158.8 |
|  |  |  |  |

Q3.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. **left Join: Links customers (c) with orders (o) using customer\_id.**
2. **Group By: Groups data by customer\_unique\_id.**
3. **Count: Calculates the number of orders per unique customer.**
4. **Having Clause: Filters only customers with more than one order.**
5. **Order By: Sorts the results by total orders in descending order**

|  |
| --- |
| select  c.customer\_unique\_id,  count(o.order\_id) as total\_orders  from amazon\_brazil.customers as c  left join amazon\_brazil.orders AS o  on c.customer\_id = o.customer\_id  group by c.customer\_unique\_id  having count(o.order\_id) > 1  order by total\_orders desc; |

**📢 Insights & Recommendations**

1. **high-Value Customers (16 Orders Each) Implement a loyalty program with exclusive rewards. Offer personalized discounts or early access to sales.**
2. **Retention & Upselling Identify preferred product categories for targeted promotions. Introduce subscription models or offers.**
3. **Customer Segmentation Analyze buying patterns to create engagement strategies. Encourage referrals through incentives.**

|  |  |
| --- | --- |
| customer\_unique\_id | total\_orders |
| a91e80fbe80ddc07de66a5cf9270293c | 16 |
| a6168cd79131e64acef92e3c74d6cc43 | 16 |
| 363f980585bf04c1a88fdb986011c52e | 16 |
| cbd0350d4ccba9772e8e768d4a4a5cbf | 16 |
| f9c4e8531c2fe4159beb562fd7c2bd59 | 16 |
| 3d364a7768fae99678635c4370295d20 | 16 |
| 6af40347f5dd7bdd65437a35e1b2fa7b | 16 |
| f300b00a19af4d4f7bdf9f4524c4587a | 16 |
| 75f15790b1852b42b1dbf645d98ffa1c | 16 |
| 8d50f5eadf50201ccdcedfb9e2ac8455 | 15 |
| 7c396fd4830fd04220f754e42b4e5bff | 14 |
| ce2e0ace655301bc4a8cae4abbd8c0a2 | 11 |
| b11b7871c2b8be2d11fab954f58542f2 | 11 |
| 2ee0d7587a1a04de482c6211ea2988bc | 9 |
| 1d8f1ca82cf6c04de2391eb61b9ae364 | 9 |

Q4.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. COUNT(\*) OVER(PARTITION BY c.customer\_id): Calculates total orders per customer without collapsing rows.
2. CASE Statement:"New" → 1 order,"Returning" → 2 to 4 order,"Loyal" → More than 4 orders
3. GROUP BY: Ensures correct aggregation by customer and order.
4. ORDER BY total\_orders DESC: Lists customers with the most orders first

|  |
| --- |
| Select  c.customer\_id,  count(o.order\_id) over (partition by c.customer\_id) as total\_orders,  case  when count(o.order\_id) over (partition by c.customer\_id) = 1 then 'New'  when count(o.order\_id) over (partition by c.customer\_id) between 2 and 4 then 'Returning'  else'Loyal'  end as customer\_type  from amazon\_brazil.orders AS o  join amazon\_brazil.customers AS c  on o.customer\_id = c.customer\_id  group by c.customer\_id, o.order\_id  order by total\_orders DESC; |

**📢 Insights & Recommendations**

1. Customer Engagement Strategies: New Customers (1 Order):
2. Offer welcome discounts and personalized recommendations to encourage repeat purchases.
3. Returning Customers (2-4 Orders): Implement targeted email campaigns with exclusive deals to build loyalty.
4. Loyal Customers (>4 Orders): Introduce VIP benefits, early access to sales, and personalized product bundles to retain them.
5. Marketing & Promotions: Design category-specific promotions based on customer behavior. Implement automated follow-ups for cart and past purchases.
6. Data-Driven Growth: Regularly analyze customer transitions between categories to refine engagement strategies. Leverage AI-driven insights to improve product recommendations and pricing strategies.

Q5.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. Joins three tables:
2. products (p): Contains product details, including product\_category\_name.
3. order\_items (oi): Links products to orders via product\_id.
4. payments (py): Contains payment details, linked to orders via order\_id.
5. Aggregates revenue: sums up payment\_value (total amount paid per order) for each product\_category\_name.ROUND() ensures the revenue values are rounded to the nearest whole number
6. Sorts and limits the resul total revenue (total\_revenue DESC).

|  |
| --- |
| select p.product\_category\_name,  ROUND(sum(py.payment\_value)) as total\_revenue  from amazon\_brazil.products as p  join amazon\_brazil.order\_items as oi on p.product\_id = oi.product\_id  join amazon\_brazil.payments as py on oi.order\_id = py.order\_id  group by p.product\_category\_name  order by total\_revenue DESC  limit 5; |

**📢 Insights & Recommendations**

1. Increase Inventory & Visibility – Ensure stock availability for top-selling categories and optimize listings with high-quality images & descriptions.
2. Personalized Marketing – Use customer purchase history to promote related products and offer discounts to boost repeat purchases.
3. Offers & Discounts – Create combo deals for complementary products (e.g., bedding set s, skincare kits)
4. . Leverage Seasonal Trends – Promote home & decor items during festive seasons and health products during wellness campaign.

|  |  |
| --- | --- |
| product\_category\_name | total\_revenue |
| cama\_mesa\_banho | 1706394 |
| beleza\_saude | 1656409 |
| informatica\_acessorios | 1583704 |
| moveis\_decoracao | 1429063 |
| relogios\_presentes | 1427211 |

Analysis – III

Q1. 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:**

**EXTRACT(MONTH) and a CASE statement**

**2. Joining the orders and payments tables on order\_id.**

**3. Summing payment values (SUM(payment\_value)) for each season.**

**4. Grouping by season to aggregate total sales.**

**5. Sorting in descending order (ORDER BY total\_sales DESC).**

|  |
| --- |
| select season, total\_sales  from (  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(py.payment\_value) as total\_sales  from amazon\_brazil.orders as o  join amazon\_brazil.payments as py  on o.order\_id = py.order\_id  group by season  ) as seasonal\_sales  order by total\_sales DESC |

**📢 Insights & Recommendations**

1. Highest Sales: Spring (₹4.93M) & Summer (₹4.89M)
2. Lowest Sales: Autumn (₹2.77M) & Winter (₹3.41M)
3. Summer Promotions – Leverage peak seasons by launching exclusive discounts, flash sales, and limited-time offers.
4. Boost Autumn & Winter Sales – Introduce winter-specific products, seasonal bundles, and targeted marketing campaigns to drive demand.
5. Seasonal Inventory Planning – Ensure sufficient stock for top-selling products in Spring & Summer to avoid stockouts. Personalized Marketing – Use customer purchase history to offer relevant products based on seasonal preferences.

|  |  |
| --- | --- |
| season | total\_sales |
| Spring | 4934990 |
| Summer | 4890902 |
| Winter | 3416285 |
| Autumn | 2766695 |

Q2.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. **Calculates total quantity sold** for each product\_id by summing order\_item\_id.
2. **Groups data by product\_id** to get sales per product.
3. **Sorts the results in descending order (ORDER BY total\_quantity DESC)**, showing the highest-selling products first.

|  |
| --- |
| SELECT o.product\_id, SUM(o.order\_item\_id) AS total\_quantity\_sold  FROM amazon\_brazil.order\_items AS o  GROUP BY o.product\_id  HAVING SUM(o.order\_item\_id) > (  SELECT AVG(total\_quantity)  FROM (SELECT product\_id, SUM(order\_item\_id) AS total\_quantity  FROM amazon\_brazil.order\_items  GROUP BY product\_id  ) AS avg\_sales  )ORDER BY total\_quantity\_sold DESC; |

**📢 Insights & Recommendations**

1. Top-Selling Products: Focus marketing efforts on these products by increasing visibility through ads and promotions.
2. Inventory Management: Ensure these products are well-stocked to prevent shortages.
3. Cross-Selling Opportunities: Bundle these high-performing products with related lower-performing items to boost overall sales.
4. Price Optimization: If demand is strong, consider slight price adjustments to maximize revenue without affecting sales

|  |  |
| --- | --- |
| product\_id | total\_quantity\_sold |
| 422879e10f46682990de24d770e7f83d | 793 |
| aca2eb7d00ea1a7b8ebd4e68314663af | 640 |
| 368c6c730842d78016ad823897a372db | 551 |
| 53759a2ecddad2bb87a079a1f1519f73 | 545 |
| 99a4788cb24856965c36a24e339b6058 | 542 |
| 389d119b48cf3043d311335e499d9c6b | 534 |
| d1c427060a0f73f6b889a5c7c61f2ac4 | 369 |
| a62e25e09e05e6faf31d90c6ec1aa3d1 | 367 |
| 53b36df67ebb7c41585e8d54d6772e08 | 359 |
| 3dd2a17168ec895c781a9191c1e95ad7 | 306 |

Q3.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. extracts the month from order\_purchase\_timestamp (EXTRACT(month FROM o.order\_purchase\_timestamp) AS month).
2. Filters orders from 2018 (WHERE EXTRACT(year FROM o.order\_purchase\_timestamp) = 2018).
3. Calculates total revenue for each month by summing payment\_value from the payments table.
4. Groups data by month (GROUP BY month) and sorts it in chronological order (ORDER BY month).

|  |
| --- |
| **select**  **extract(month from o.order\_purchase\_timestamp) as month,**  **sum(**  **(select sum(p.payment\_value)**  **from amazon\_brazil.payments p**  **where p.order\_id = o.order\_id)**  **) as total\_revenue**  **from amazon\_brazil.orders o**  **where extract(year from o.order\_purchase\_timestamp) = 2018**  **group by month**  **order by month;** |

**📢Insights & Recommendations**

1. **Peak Revenue Period (January – May)The highest revenue was recorded between January and May, with each month exceeding 1.1 million in sales.**
2. **April (1,160,785) had the highest revenue, followed closely by March (1,159,652) and May (1,153,982).Stable Performance (June – August)**
3. **Revenue remained consistent between June and August, averaging around 1 million per month.This suggests steady demand without significant seasonal drops.Drastic Drop in September & October**
4. **September (4,439.54) and October (589.67) saw a major decline in revenue.This could indicate seasonality, a drop in demand, stock issues, or platform changes.**
5. **Focus on Q1 & Q2 (Jan – May): Increase marketing, promotions, and inventory during these months maximize revenue.  
   Investigate the Drop (Sept – Oct): Identify reasons for the sharp decline (e.g., demand shifts, supply chain issues, or platform changes)**
6. **Implement promotions, discounts, or ad campaigns to regain sales.  
    Leverage Q3 Stability (June – Aug): Maintain steady stock and targeted marketing to sustain revenue**

|  |  |
| --- | --- |
| month | total\_revenue |
| 1 | 1115004 |
| 2 | 992463.3 |
| 3 | 1159652 |
| 4 | 1160785 |
| 5 | 1153982 |
| 6 | 1023881 |
| 7 | 1066541 |
| 8 | 1022425 |
| 9 | 4439.54 |
| 10 | 589.67 |

Q4A 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. **Define Customer Type (WITH customer\_type AS (...))**
2. **Counts total orders per customer (count(o.order\_id) AS total\_orders).**
3. **Assigns a customer type based on order frequency:1-2 orders → "Occasional",3-5 orders → "Regular".More than 5 orders → "Loyal"**
4. **Groups by customer\_id to classify each customer.Count Customers by Type (SELECT customer\_type, COUNT(\*)Counts how many customers belong to each category.**

|  |
| --- |
| with customer\_type as (  select  count(o.order\_id) AS total\_orders,  case  when count(o.order\_id) between 1 and 2 then 'Occasional'  when count(o.order\_id) between 3 and 5 then 'Regular'  else 'Loyal'  end as customer\_type from amazon\_brazil.orders as o  join amazon\_brazil.customers as c on o.customer\_id = c.customer\_id  group by c.customer\_id  )select customer\_type, count(\*) as customer\_count  FROM customer\_type  group by customer\_type  order by customer\_count DESC;; |

1. **Orders the result in descending order (ORDER BY customer\_count DESC)**

**📢Insights & Recommendations**

1. Majority (98,144) are occasional customers,
2. Indicating a need for better retention strategies. Only 106 regular and 98 loyal customers,
3. Suggesting low long-term engagement. Implement loyalty programs, personalized offers, and targeted marketing to convert occasional buyers into regular and loyal customers.

|  |  |
| --- | --- |
| customer\_type | customer\_count |
| Occasional | 98144 |
| Regular | 106 |
| Loyal | 98 |

Q5.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. **Average Order Value (avg\_order\_value) for Each Customer Joins orders and payments tables to get payment\_value for each order.**
2. **Computes the average payment per customer (AVG(p.payment\_value)). And Rounds the value to one decimal place (ROUND(AVG(p.payment\_value), 1)) Assign Rank to Customers (customer\_rank)**
3. **Uses the RANK() window function to rank customers in descending order of avg\_order\_value.Retrieve the Top 20 Customers**
4. **Selects customer\_id, avg\_order\_value, and customer\_rank. Orders by customer\_rank (highest-paying first). And Limits output to top 20 customers (LIMIT 20).**

|  |
| --- |
| **with customer\_rank as (**  **select**  **o.customer\_id,**  **round (avg(p.payment\_value),1)as avg\_order\_value,**  **rank() over (order by avg(p.payment\_value) desc) aS customer\_rank**  **from amazon\_brazil.orders as o**  **join amazon\_brazil.payments as p**  **on o.order\_id = p.order\_id**  **group by o.customer\_id**  **)**  **select customer\_id, avg\_order\_value, customer\_rank**  **from customer\_rank**  **order by customer\_rank**  **limit 20;** |

|  |  |  |
| --- | --- | --- |
| customer\_id | avg\_order\_value | customer\_rank |
| 1617b1357756262bfa56ab541c47bc16 | 13664.1 | 1 |
| ec5b2ba62e574342386871631fafd3fc | 7274.9 | 2 |
| c6e2731c5b391845f6800c97401a43a9 | 6929.3 | 3 |
| f48d464a0baaea338cb25f816991ab1f | 6922.2 | 4 |
| 3fd6777bbce08a352fddd04e4a7cc8f6 | 6726.7 | 5 |
| 05455dfa7cd02f13d132aa7a6a9729c6 | 6081.5 | 6 |
| df55c14d1476a9a3467f131269c2477f | 4950.3 | 7 |
| e0a2412720e9ea4f26c1ac985f6a7358 | 4809.4 | 8 |
| 24bbf5fd2f2e1b359ee7de94defc4a15 | 4764.3 | 9 |
| 3d979689f636322c62418b6346b1c6d2 | 4681.8 | 10 |
| 1afc82cd60e303ef09b4ef9837c9505c | 4513.3 | 11 |
| cc803a2c412833101651d3f90ca7de24 | 4445.5 | 12 |
| 35a413c7ca3c69756cb75867d6311c0d | 4175.3 | 13 |
| e9b0d0eb3015ef1c9ce6cf5b9dcbee9f | 4163.5 | 14 |
| 3be2c536886b2ea4668eced3a80dd0bb | 4042.7 | 15 |
| c6695e3b1e48680db36b487419fb0398 | 4016.9 | 16 |
| 31e83c01fce824d0ff786fcd48dad009 | 3979.6 | 17 |
| addc91fdf9c2b3045497b57fc710e820 | 3826.8 | 18 |
| 19b32919fa1198aefc0773ee2e46e693 | 3792.6 | 19 |
| 58acc4e2788bf6fc445fddcce9c1db03 | 3782.2 | 20 |

**📢Insights & Recommendations**

1. Top 20 High-Value Customers: Highest avg. order value: ₹13,664 (Customer 1) Top 5 customers have avg. orders above ₹6,700
2. Last in the top 20 still has a strong ₹3,782 avg. order. Prioritize retention strategies like VIP perks, exclusive offers, and personalized engagement to maximize revenue from high-value customers

Q6.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. **Create product\_sales CTE (Common Table Expression)** Extracts **year (sale\_year) and month (sale\_month)** from order\_purchase\_timestamp.
2. Joins the orders, order\_items, products, and payments tables.Groups by product\_id, sale\_year, and sale\_month to calculate **monthly sales (sum(payment\_value))**.
3. **Compute Cumulative Sales (total\_sales)**Uses **SUM() OVER (PARTITION BY product\_id ORDER BY sale\_year, sale\_month)**.This **cumulates sales** for each product **month by month**.
4. Orders by product\_id, sale\_year, and sale\_month

|  |
| --- |
| **with product\_sales as (**  **select**  **pr.product\_id,**  **extract(year from o.order\_purchase\_timestamp) as sale\_year,**  **extract(month from o.order\_purchase\_timestamp) as sale\_month,**  **sum(p.payment\_value) as monthly\_sales**  **from amazon\_brazil.orders as o**  **join amazon\_brazil.order\_items as oi on o.order\_id = oi.order\_id**  **join amazon\_brazil.products as pr on oi.product\_id = pr.product\_id**  **join amazon\_brazil.payments as p on o.order\_id = p.order\_id**  **group by pr.product\_id, sale\_year, sale\_month**  **)select**  **product\_id,**  **sale\_year,**  **sale\_month,**  **sum(monthly\_sales) over (partition by product\_id order by sale\_year, sale\_month) as total\_sales**  **from product\_sales**  **order by product\_id, sale\_year, sale\_month;** |

**📢Insights & Recommendations**

1. **Identify High-Performing Products:Look for products with consistently high sales figures across multiple months and years.**
2. **Consider creating customer retention metrics based on repeat purchases of the same product over time.**
3. **Analyze the data to identify any seasonal trends or patterns in sales. Compare sales figures year-over-year to identify growth or decline in sales for each product.**

|  |  |  |  |
| --- | --- | --- | --- |
| product\_id | sale\_year | sale\_month | total\_sales |
| 00066f42aeeb9f3007548bb9d3f33c38 | 2018 | 5 | 120.24 |
| 00088930e925c41fd95ebfe695fd2655 | 2017 | 12 | 143.83 |
| 0009406fd7479715e4bef61dd91f2462 | 2017 | 12 | 242.1 |
| 000b8f95fcb9e0096488278317764d19 | 2018 | 8 | 157 |
| 000d9be29b5207b54e86aa1b1ac54872 | 2018 | 4 | 218.27 |
| 0011c512eb256aa0dbbb544d8dffcf6e | 2017 | 12 | 166.4 |
| 00126f27c813603687e6ce486d909d01 | 2017 | 9 | 527.73 |
| 001795ec6f1b187d37335e1c4704762e | 2017 | 10 | 47.62 |
| 001795ec6f1b187d37335e1c4704762e | 2017 | 11 | 149.87 |
| 001795ec6f1b187d37335e1c4704762e | 2017 | 12 | 662.7 |
| 001b237c0e9bb435f2e54071129237e9 | 2018 | 8 | 100.09 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 2 | 253.26 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 3 | 306.03 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 7 | 527.2 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 8 | 657.77 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 9 | 757.62 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 11 | 801.33 |
| 001b72dfd63e9833e8c02742adf472e3 | 2017 | 12 | 888.75 |
| 001c5d71ac6ad696d22315953758fa04 | 2017 | 1 | 100.64 |

Q7. .To 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. **Creating monthly\_sales (Common Table Expression - CTE)**
2. **Extracts the payment type and month of the purchase.**
3. **Sums up the payment values for each payment method per month.**
4. **Filters only orders from the year 2018.**
5. **Groups the data by payment\_type and sale\_month.**
6. **Calculating Month-over-Month Growth**
7. **Uses LAG(monthly\_total) to get the sales value of the previous month.**
8. **Computes the growth rate formula:**
9. **MoM Growth=(Current Month Sales−Previous Month Sales)Previous Month Sales×100\text{MoM Growth} = \frac{(\text{Current Month Sales} - \text{Previous Month Sales})}{\text{Previous Month Sales}} \times 100MoM Growth=Previous Month Sales(Current Month Sales−Previous Month Sales)​×100**
10. **NULLIF(LAG(monthly\_total), 0) is used to prevent division by zero errors.**
11. **Orders the result by payment\_type and sale\_month for better readability.**

|  |
| --- |
| **with monthly\_sales as (**  **select**  **py.payment\_type,**  **extract(month from o.order\_purchase\_timestamp) as sale\_month,**  **sum(py.payment\_value) as monthly\_total**  **from amazon\_brazil.payments as py**  **WITH monthly\_sales AS (**  **SELECT**  **py.payment\_type,**  **EXTRACT(month FROM o.order\_purchase\_timestamp) AS sale\_month,**  **ROUND(SUM(py.payment\_value), 2) AS monthly\_total**  **FROM amazon\_brazil.payments AS py**  **JOIN amazon\_brazil.orders AS o**  **ON py.order\_id = o.order\_id**  **WHERE EXTRACT(year FROM o.order\_purchase\_timestamp) = 2018**  **GROUP BY py.payment\_type, sale\_month**  **),**  **growth\_calc AS (**  **SELECT**  **payment\_type,**  **sale\_month,**  **monthly\_total,**  **LAG(monthly\_total) OVER (PARTITION BY payment\_type ORDER BY sale\_month) AS prev\_month\_sales**  **FROM monthly\_sales**  **)**  **SELECT**  **payment\_type,**  **sale\_month,**  **monthly\_total,**  **CASE**  **WHEN prev\_month\_sales IS NULL THEN '-'**  **ELSE ROUND(((monthly\_total - prev\_month\_sales) / NULLIF(prev\_month\_sales, 0)) \* 100, 2) || '%'**  **END AS monthly\_change**  **FROM growth\_calc**  **ORDER BY payment\_type, sale\_month;FROM monthly\_sales ORDER BY payment\_type, sale\_month;** |

**📢Insights & Recommendations**

1. **Boleto: The drop in month 6 and 8 suggests instability—Amazon could analyze reasons for reduced usage (e.g., failed transactions, customer preferences).**
2. **Credit Card: The primary method used but declining in the later months—consider offering credit card EMI or cashback incentives.**
3. **Debit Card: Huge spike in month 6—investigate if a promotional offer or festival sale impacted the increase, and replicate successful strategies**

|  |  |  |  |
| --- | --- | --- | --- |
| payment\_type | sale\_month | monthly\_total | monthly\_change |
| boleto | 1 | 204844.7 | - |
| boleto | 2 | 183112.7 | -10.61% |
| boleto | 3 | 191538 | 4.60% |
| boleto | 4 | 193547.1 | 1.05% |
| boleto | 5 | 195378.9 | 0.95% |
| boleto | 6 | 153350.3 | -21.51% |
| boleto | 7 | 198041.2 | 29.14% |
| boleto | 8 | 143805.9 | -27.39% |
| credit\_card | 1 | 868880.4 | - |
| credit\_card | 2 | 778803 | -10.37% |
| credit\_card | 3 | 933770.1 | 19.90% |
| credit\_card | 4 | 934306 | 0.06% |
| credit\_card | 5 | 927556.4 | -0.72% |
| credit\_card | 6 | 811508.6 | -12.51% |
| credit\_card | 7 | 803674.5 | -0.97% |
| credit\_card | 8 | 797648.9 | -0.75% |
| debit\_card | 1 | 11543.55 | - |
| debit\_card | 2 | 7469.53 | -35.29% |
| debit\_card | 3 | 8375.11 | 12.12% |

**Thank you**