Amazon Data Analysis Report

SQL-Based Insights & Recommendations

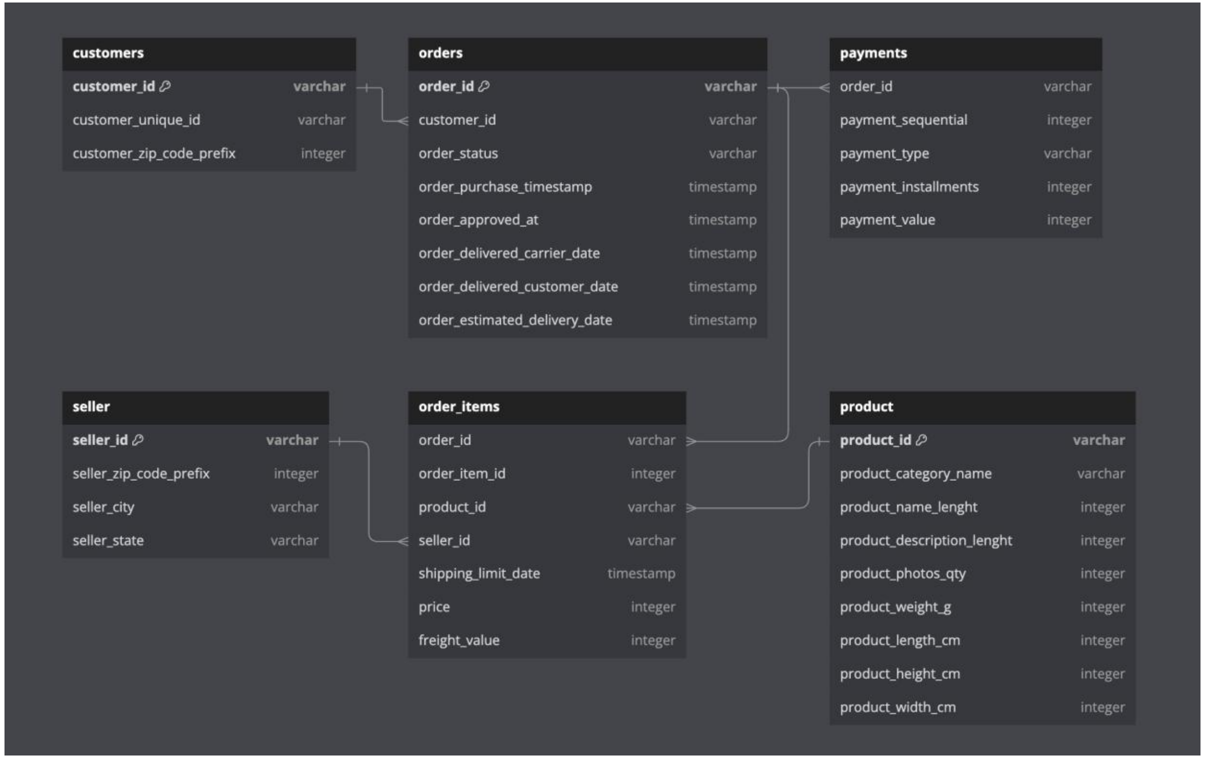
Prepared by :
Manikanta Pudi

Introduction

- Amazon India aims to improve operations using data analysis.
- SQL queries analyze transactions, payments, and customer trends.
- Insights will help in making data-driven decisions.

Database Schema Overview

- **Schema Diagram** (Visual representation of database relationships)



customers	
customer_id	**varchar**
customer_unique_id	varchar
customer_zip_code_prefix	integer

orders	
order_id	**varchar**
customer_id	varchar
order_status	varchar
order_purchase_timestamp	timestamp
order_approved_at	timestamp
order_delivered_carrier_date	timestamp
order_delivered_customer_date	timestamp
order_estimated_delivery_date	timestamp

payments	
order_id	varchar
payment_sequential	integer
payment_type	varchar
payment_installments	integer
payment_value	integer

seller	
seller_id	**varchar**
seller_zip_code_prefix	integer
seller_city	varchar
seller_state	varchar

order_items	
order_id	varchar
order_item_id	integer
product_id	varchar
seller_id	varchar
shipping_limit_date	timestamp
price	integer
freight_value	integer

product	
product_id	**varchar**
product_category_name	varchar
product_name_lenght	integer
product_description_lenght	integer
product_photos_qty	integer
product_weight_g	integer
product_length_cm	integer
product_height_cm	integer
product_width_cm	integer

Table Descriptions

- **Customers:** Stores unique customer details.
- **Orders:** Tracks order details and timestamps.
- **Order Items:** Contains product price, seller, and shipping information.
- **Products:** Includes category, size, and weight details.
- **Sellers:** Provides seller identification and location.
- **Payments:** Records payment types and values.

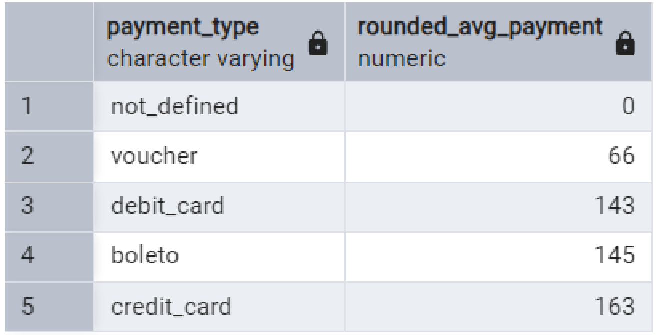
Analysis - I

Task 1 : Standardizing Payment Values

Query: Round the average payment values to integers for each payment type and display the results sorted in ascending order.

```
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;
```

Output:



	payment_type character varying	rounded_avg_payment numeric
1	not_defined	0
2	voucher	66
3	debit_card	143
4	boleto	145
5	credit_card	163

Analysis: This ensures consistency in financial reporting and simplifies data interpretation.

Recommendation: Use standardized values in financial reports for better accuracy in tracking revenue.

Task 2 : Payment Distribution by Orders

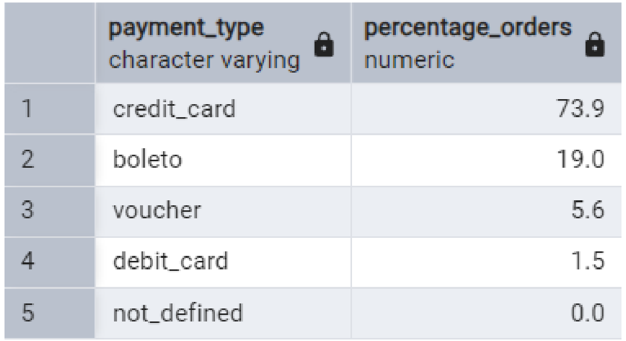
Query: Calculate the percentage of total orders for each payment type, rounded to one decimal place.

```
SELECT payment_type,
round((count(*) * 100) / sum(count(*)) over(),1)
AS percentage_orders
FROM amazon_brazil.payments
GROUP BY payment_type
ORDER BY percentage_orders desc;
```

Analysis: Helps identify preferred payment methods among customers.

Recommendation: Optimize payment methods based on customer preferences to improve checkout experiences.

Output:



	payment_type character varying	percentage_orders numeric
1	credit_card	73.9
2	boleto	19.0
3	voucher	5.6
4	debit_card	1.5
5	not_defined	0.0

Task 3 : Product Promotions Analysis

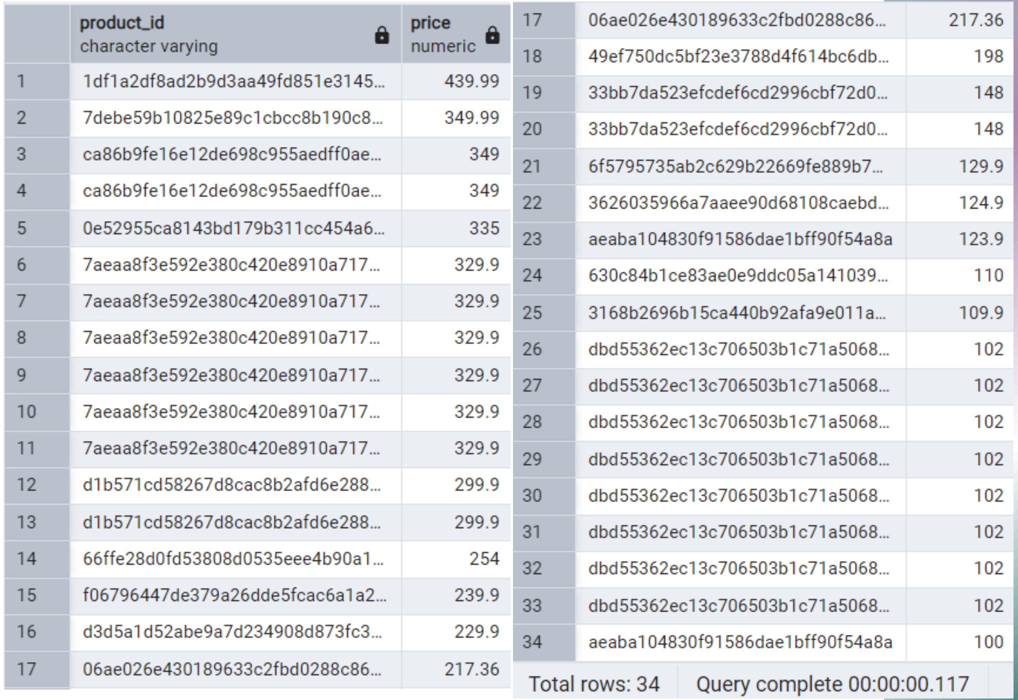
Query: Retrieve products priced between 100 and 500 BRL containing 'Smart' in their name.

```sql
SELECT o.product_id , o.price
FROM amazon_brazil.order_items o
INNER JOIN amazon_brazil.product p
ON o.product_id = p.product_id
AND p.product_category_name like '%smart%'
WHERE o.price BETWEEN 100 AND 500
ORDER BY o.price desc;
```

Analysis: Targets product promotions effectively.

Recommendation: Focus marketing campaigns on these products to boost sales.

Output:

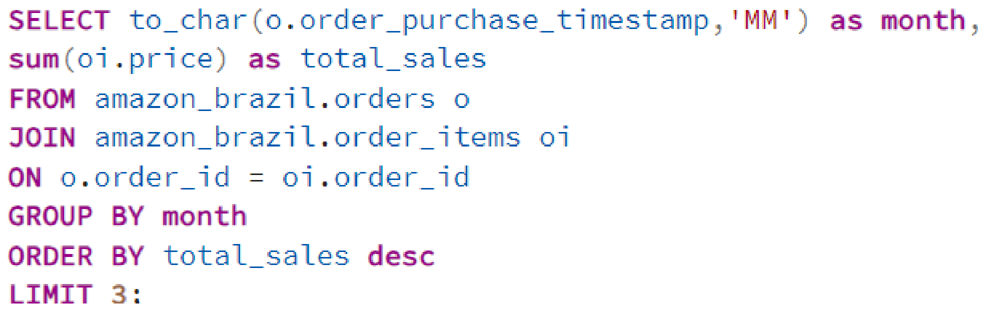


	product_id character varying	price numeric
1	1df1a2df8ad2b9d3aa49fd851e3145...	439.99
2	7debe59b10825e89c1cbcc8b190c8...	349.99
3	ca86b9fe16e12de698c955aedff0ae...	349
4	ca86b9fe16e12de698c955aedff0ae...	349
5	0e52955ca8143bd179b311cc454a6...	335
6	7aeaa8f3e592e380c420e8910a717...	329.9
7	7aeaa8f3e592e380c420e8910a717...	329.9
8	7aeaa8f3e592e380c420e8910a717...	329.9
9	7aeaa8f3e592e380c420e8910a717...	329.9
10	7aeaa8f3e592e380c420e8910a717...	329.9
11	7aeaa8f3e592e380c420e8910a717...	329.9
12	d1b571cd58267d8cac8b2afd6e288...	299.9
13	d1b571cd58267d8cac8b2afd6e288...	299.9
14	66ffe28d0fd53808d0535eee4b90a1...	254
15	f06796447de379a26dde5fcac6a1a2...	239.9
16	d3d5a1d52abe9a7d234908d873fc3...	229.9
17	06ae026e430189633c2fbd0288c86...	217.36
18	49ef750dc5bf23e3788d4f614bc6db...	198
19	33bb7da523efcdef6cd2996cbf72d0...	148
20	33bb7da523efcdef6cd2996cbf72d0...	148
21	6f5795735ab2c629b22669fe889b7...	129.9
22	3626035966a7aaee90d68108caebd...	124.9
23	aeaba104830f91586dae1bff90f54a8a	123.9
24	630c84b1ce83ae0e9ddc05a141039...	110
25	3168b2696b15ca440b92afa9e011a...	109.9
26	dbd55362ec13c706503b1c71a5068...	102
27	dbd55362ec13c706503b1c71a5068...	102
28	dbd55362ec13c706503b1c71a5068...	102
29	dbd55362ec13c706503b1c71a5068...	102
30	dbd55362ec13c706503b1c71a5068...	102
31	dbd55362ec13c706503b1c71a5068...	102
32	dbd55362ec13c706503b1c71a5068...	102
33	dbd55362ec13c706503b1c71a5068...	102
34	aeaba104830f91586dae1bff90f54a8a	100

Total rows: 34 Query complete 00:00:00.117

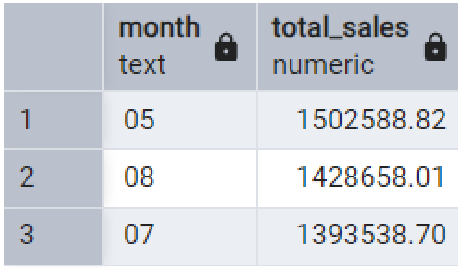
Task 4 : Identifying Most Successful Sales of Top 3 Months

Query: Determine the top 3 months with the highest total sales value, rounded to the nearest integer.



```sql
SELECT to_char(o.order_purchase_timestamp,'MM') as month,
sum(oi.price) as total_sales
FROM amazon_brazil.orders o
JOIN amazon_brazil.order_items oi
ON o.order_id = oi.order_id
GROUP BY month
ORDER BY total_sales desc
LIMIT 3;
```

Output:



	month text	total_sales numeric
1	05	1502588.82
2	08	1428658.01
3	07	1393538.70

Analysis: Helps detect peak seasons.

Recommendation: Optimize inventory and marketing efforts during peak months.

Task 5 : Identifying Categories with High Price Variations

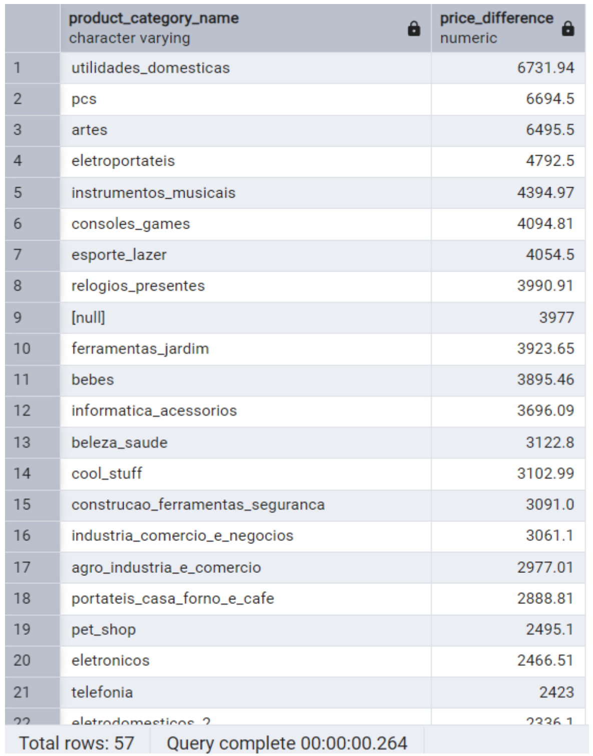
Query: Find product categories where the difference between max and min prices is greater than 500 BRL.

```sql
SELECT p.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;
```

Analysis: Highlights price diversity within product categories.

Recommendation: Adjust pricing strategies for high-variance categories.

Output:



	product_category_name character varying	price_difference numeric
1	utilidades_domesticas	6731.94
2	pcs	6694.5
3	artes	6495.5
4	eletroportateis	4792.5
5	instrumentos_musicais	4394.97
6	consoles_games	4094.81
7	esporte_lazer	4054.5
8	relogios_presentes	3990.91
9	[null]	3977
10	ferramentas_jardim	3923.65
11	bebes	3895.46
12	informatica_acessorios	3696.09
13	beleza_saude	3122.8
14	cool_stuff	3102.99
15	construcao_ferramentas_seguranca	3091.0
16	industria_comercio_e_negocios	3061.1
17	agro_industria_e_comercio	2977.01
18	portateis_casa_forno_e_cafe	2888.81
19	pet_shop	2495.1
20	eletronicos	2466.51
21	telefonia	2423
22	eletrodomesticos_2	2336.1

Total rows: 57 Query complete 00:00:00.264

Task 6 : Identifying Consistent Payment Types

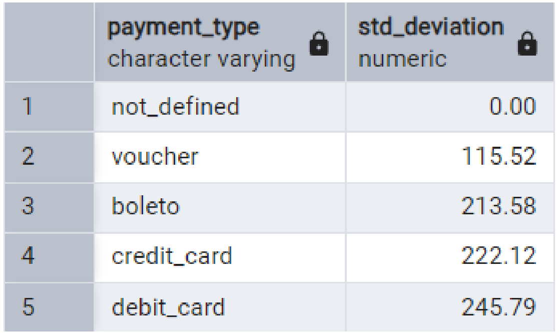
Query: Identify payment types with the least variance in transaction amounts.

```
SELECT payment_type,
round(STDDEV(payment_value),2) as std_deviation
FROM amazon_brazil.payments
GROUP BY payment_type
ORDER BY std_deviation asc;
```

Analysis: Provides insights into stable payment methods.

Recommendation: Promote consistent payment types to reduce transactional risks.

Output:



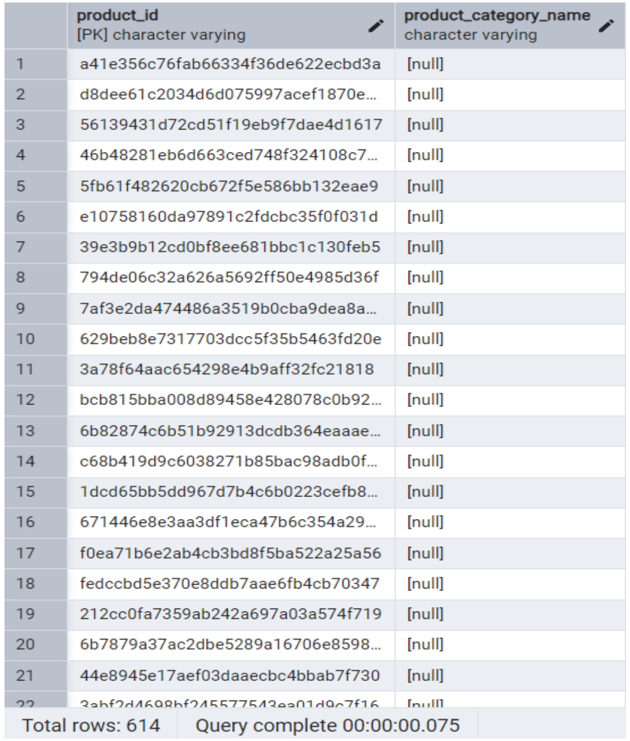
	payment_type character varying	std_deviation numeric
1	not_defined	0.00
2	voucher	115.52
3	boleto	213.58
4	credit_card	222.12
5	debit_card	245.79

Task 7 : Identifying Incomplete Product Names

Query: Retrieve products where the product category name is missing or contains only a single character.

```sql
SELECT product_id , product_category_name
FROM amazon_brazil.product
WHERE product_category_name IS NULL
OR LENGTH(product_category_name) = 1;
```

Output:



	product_id [PK] character varying	product_category_name character varying
1	a41e356c76fab66334f36de622ecbd3a	[null]
2	d8dee61c2034d6d075997acef1870e...	[null]
3	56139431d72cd51f19eb9f7dae4d1617	[null]
4	46b48281eb6d663ced748f324108c7...	[null]
5	5fb61f482620cb672f5e586bb132eae9	[null]
6	e10758160da97891c2fdcbc35f0f031d	[null]
7	39e3b9b12cd0bf8ee681bbc1c130feb5	[null]
8	794de06c32a626a5692ff50e4985d36f	[null]
9	7af3e2da474486a3519b0cba9dea8a...	[null]
10	629beb8e7317703dcc5f35b5463fd20e	[null]
11	3a78f64aac654298e4b9aff32fc21818	[null]
12	bcb815bba008d89458e428078c0b92...	[null]
13	6b82874c6b51b92913dcdb364eaaae...	[null]
14	c68b419d9c6038271b85bac98adb0f...	[null]
15	1dcd65bb5dd967d7b4c6b0223cefb8...	[null]
16	671446e8e3aa3df1eca47b6c354a29...	[null]
17	f0ea71b6e2ab4cb3bd8f5ba522a25a56	[null]
18	fedccbd5e370e8ddb7aae6fb4cb70347	[null]
19	212cc0fa7359ab242a697a03a574f719	[null]
20	6b7879a37ac2dbe5289a16706e8598...	[null]
21	44e8945e17aef03daaecbc4bbab7f730	[null]
22	3abf2d4698bf245577543ea01d9c7f16	[null]

Total rows: 614   Query complete 00:00:00.075

Analysis: Ensures data quality and completeness.

Recommendation: Fix incomplete product names to maintain dataset integrity.

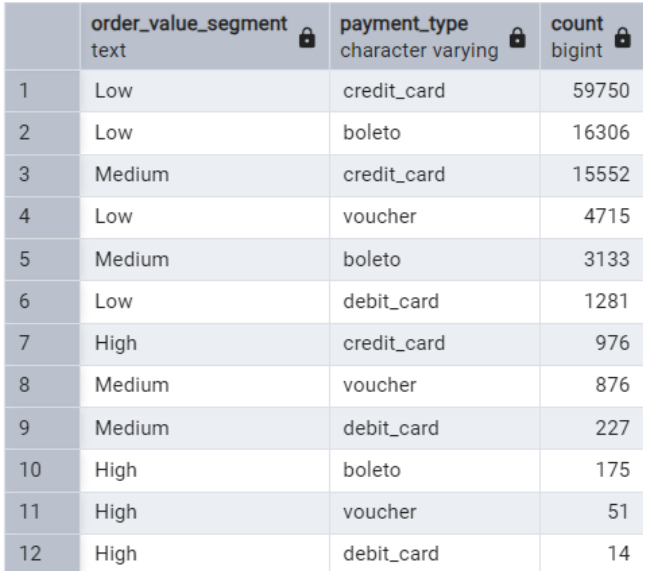
Analysis – II

Task 1 : Payment Type Popularity by Order Value Segments

Query: Segment orders into three price ranges and calculate payment type counts.

```
WITH order_value AS (
SELECT oi.order_id , sum(oi.price + oi.freight_value) as order_value
FROM amazon_brazil.order_items oi
GROUP BY oi.order_id
),
Segment_table AS(
SELECT ov.order_id ,
CASE
    WHEN ov.order_value < 200 THEN 'Low'
    WHEN ov.order_value BETWEEN 200 AND 1000 THEN 'Medium'
    ELSE 'High'
END AS order_value_segment
FROM order_value ov
)
SELECT st.order_value_segment , p.payment_type,
count(p.payment_type) as count
FROM segment_table st
JOIN amazon_brazil.payments p
ON st.order_id = p.order_id
GROUP BY st.order_value_segment,p.payment_type
ORDER BY count desc;
```

Output:



	order_value_segment text	payment_type character varying	count bigint
1	Low	credit_card	59750
2	Low	boleto	16306
3	Medium	credit_card	15552
4	Low	voucher	4715
5	Medium	boleto	3133
6	Low	debit_card	1281
7	High	credit_card	976
8	Medium	voucher	876
9	Medium	debit_card	227
10	High	boleto	175
11	High	voucher	51
12	High	debit_card	14

Analysis: Identifies preferred payment types across price ranges.

Recommendation: Offer customized payment incentives for different segments.

Task 2 : Product Category Price Ranges

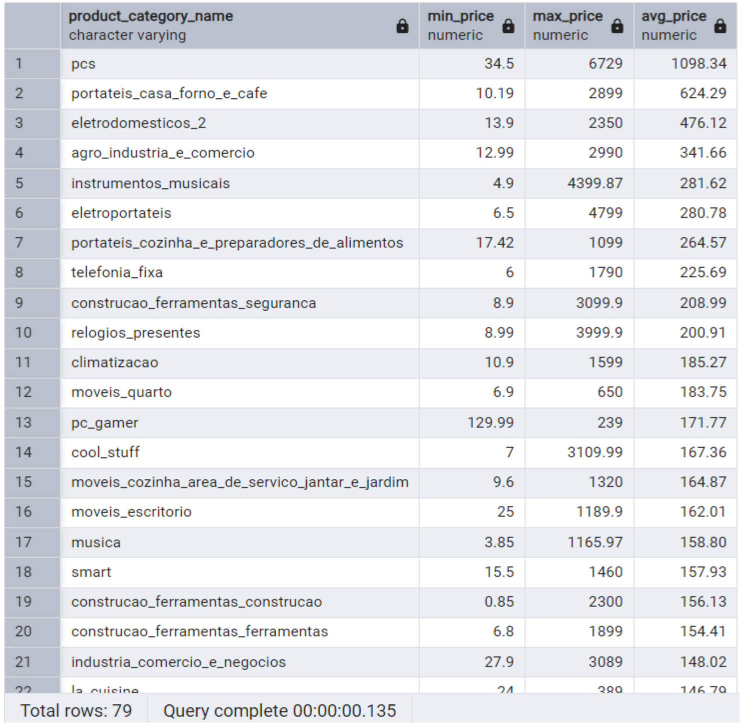
Query: Calculate the minimum, maximum, and average price for each category.

```
SELECT p.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
LEFT JOIN amazon_brazil.order_items oi
ON p.product_id = oi.product_id
GROUP BY p.product_category_name
ORDER BY avg_price desc;
```

Analysis: Helps in pricing strategies and market positioning.

Recommendation: Adjust category pricing for competitive advantage.

Output:



	product_category_name character varying	min_price numeric	max_price numeric	avg_price numeric
1	pcs	34.5	6729	1098.34
2	portateis_casa_forno_e_cafe	10.19	2899	624.29
3	eletrodomesticos_2	13.9	2350	476.12
4	agro_industria_e_comercio	12.99	2990	341.66
5	instrumentos_musicais	4.9	4399.87	281.62
6	eletroportateis	6.5	4799	280.78
7	portateis_cozinha_e_preparadores_de_alimentos	17.42	1099	264.57
8	telefonia_fixa	6	1790	225.69
9	construcao_ferramentas_seguranca	8.9	3099.9	208.99
10	relogios_presentes	8.99	3999.9	200.91
11	climatizacao	10.9	1599	185.27
12	moveis_quarto	6.9	650	183.75
13	pc_gamer	129.99	239	171.77
14	cool_stuff	7	3109.99	167.36
15	moveis_cozinha_area_de_servico_jantar_e_jardim	9.6	1320	164.87
16	moveis_escritorio	25	1189.9	162.01
17	musica	3.85	1165.97	158.80
18	smart	15.5	1460	157.93
19	construcao_ferramentas_construcao	0.85	2300	156.13
20	construcao_ferramentas_ferramentas	6.8	1899	154.41
21	industria_comercio_e_negocios	27.9	3089	148.02
22	la_cuisine	24	389	146.79

Total rows: 79 Query complete 00:00:00.135

Task 3 : Identifying Frequent Customers

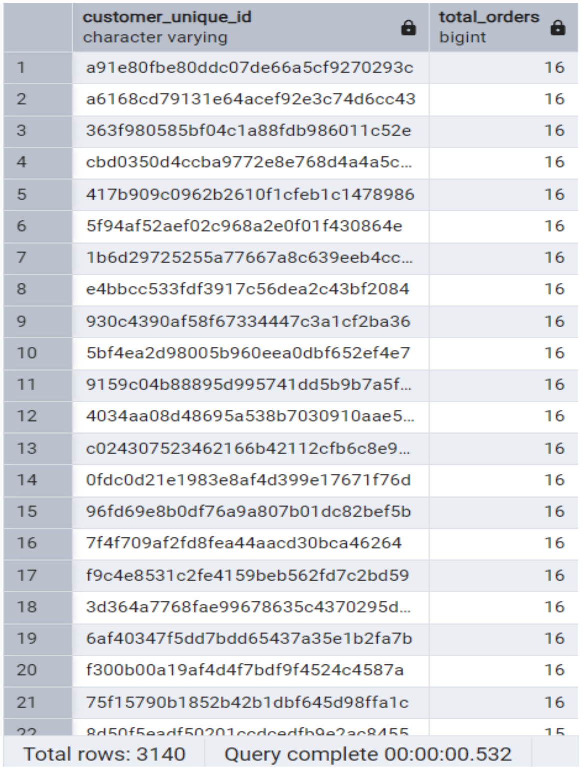
Query: Find customers with more than one order.

```sql
SELECT c.customer_unique_id, count(o.order_id) as total_orders
FROM amazon_brazil.customer c
JOIN amazon_brazil.orders 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;
```

Analysis: Helps track repeat buyers.

Recommendation: Reward frequent customers with loyalty programs.

Output:



	customer_unique_id character varying	total_orders bigint
1	a91e80fbe80ddc07de66a5cf9270293c	16
2	a6168cd79131e64acef92e3c74d6cc43	16
3	363f980585bf04c1a88fdb986011c52e	16
4	cbd0350d4ccba9772e8e768d4a4a5c...	16
5	417b909c0962b2610f1cfeb1c1478986	16
6	5f94af52aef02c968a2e0f01f430864e	16
7	1b6d29725255a77667a8c639eeb4cc...	16
8	e4bbcc533fdf3917c56dea2c43bf2084	16
9	930c4390af58f67334447c3a1cf2ba36	16
10	5bf4ea2d98005b960eea0dbf652ef4e7	16
11	9159c04b88895d995741dd5b9b7a5f...	16
12	4034aa08d48695a538b7030910aae5...	16
13	c02430752346216 6b42112cfb6c8e9...	16
14	0fdc0d21e1983e8af4d399e17671f76d	16
15	96fd69e8b0df76a9a807b01dc82bef5b	16
16	7f4f709af2fd8fea44aacd30bca46264	16
17	f9c4e8531c2fe4159beb562fd7c2bd59	16
18	3d364a7768fae99678635c4370295d...	16
19	6af40347f5dd7bdd65437a35e1b2fa7b	16
20	f300b00a19af4d4f7bdf9f4524c4587a	16
21	75f15790b1852b42b1dbf645d98ffa1c	16
22	8d50f5eadf50201ccdcedfb9e2ac8455	15

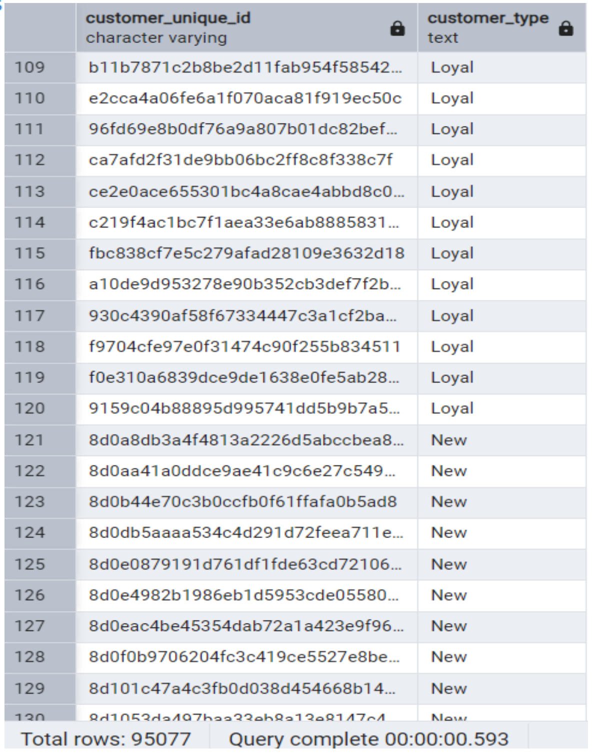
Total rows: 3140   Query complete 00:00:00.532

Task 4 : Customer Segmentation by Order Frequency

Query: Categorize customers into New, Returning, and Loyal segments.

```sql
WITH categorize_customer AS (
SELECT c.customer_unique_id, count(o.order_id) as total_orders
FROM amazon_brazil.customer c
JOIN amazon_brazil.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_unique_id
)
SELECT cc.customer_unique_id,
CASE
    WHEN cc.total_orders = 1 THEN 'New'
    WHEN cc.total_orders BETWEEN 2 AND 4 THEN 'Returning'
    ELSE 'Loyal'
END AS customer_type
FROM categorize_customer cc;
```

Output:



	customer_unique_id character varying	customer_type text
109	b11b7871c2b8be2d11fab954f58542...	Loyal
110	e2cca4a06fe6a1f070aca81f919ec50c	Loyal
111	96fd69e8b0df76a9a807b01dc82bef...	Loyal
112	ca7afd2f31de9bb06bc2ff8c8f338c7f	Loyal
113	ce2e0ace655301bc4a8cae4abbd8c0...	Loyal
114	c219f4ac1bc7f1aea33e6ab8885831...	Loyal
115	fbc838cf7e5c279afad28109e3632d18	Loyal
116	a10de9d953278e90b352cb3def7f2b...	Loyal
117	930c4390af58f67334447c3a1cf2ba...	Loyal
118	f9704cfe97e0f31474c90f255b834511	Loyal
119	f0e310a6839dce9de1638e0fe5ab28...	Loyal
120	9159c04b88895d995741dd5b9b7a5...	Loyal
121	8d0a8db3a4f4813a2226d5abccbea8...	New
122	8d0aa41a0ddce9ae41c9c6e27c549...	New
123	8d0b44e70c3b0ccfb0f61ffafa0b5ad8	New
124	8d0db5aaaa534c4d291d72feea711e...	New
125	8d0e0879191d761df1fde63cd72106...	New
126	8d0e4982b1986eb1d5953cde05580...	New
127	8d0eac4be45354dab72a1a423e9f96...	New
128	8d0f0b9706204fc3c419ce5527e8be...	New
129	8d101c47a4c3fb0d038d454668b14...	New
130	8d1053da497baa33eb8a13e8147c4	New

Total rows: 95077 Query complete 00:00:00.593

Analysis: Aids in customer retention strategies.

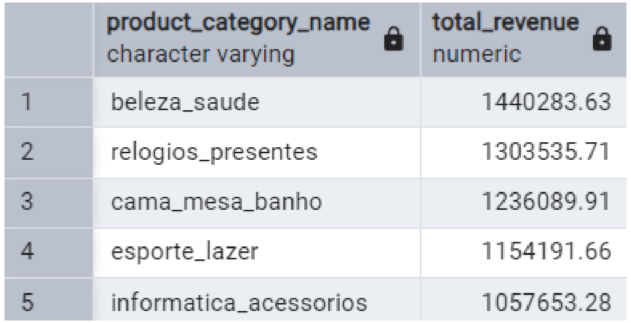
Recommendation: Tailor engagement strategies for each customer segment.

Task 5 : Identifying Revenue-Generating Product Categories

Query: Calculate total revenue for each product category and list the top five.

```sql
SELECT p.product_category_name ,
sum(oi.price + oi.freight_value) 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:



	product_category_name character varying	total_revenue numeric
1	beleza_saude	1440283.63
2	relogios_presentes	1303535.71
3	cama_mesa_banho	1236089.91
4	esporte_lazer	1154191.66
5	informatica_acessorios	1057653.28

Analysis: Determines top-selling categories.

Recommendation: Focus on high-revenue categories for promotions.

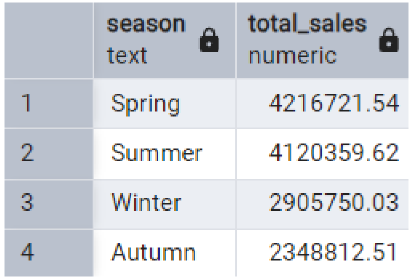
Analysis – III

Task 1 : Seasonal Sales Comparison

Query: Calculate total sales for Spring, Summer, Autumn, and Winter.

```sql
SELECT season , sum(total_sales) as 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 , oi.price as total_sales
    FROM amazon_brazil.orders o
    JOIN amazon_brazil.order_items oi
    ON o.order_id = oi.order_id
) AS Season_sales
GROUP BY season
ORDER BY total_sales desc;
```

Output:



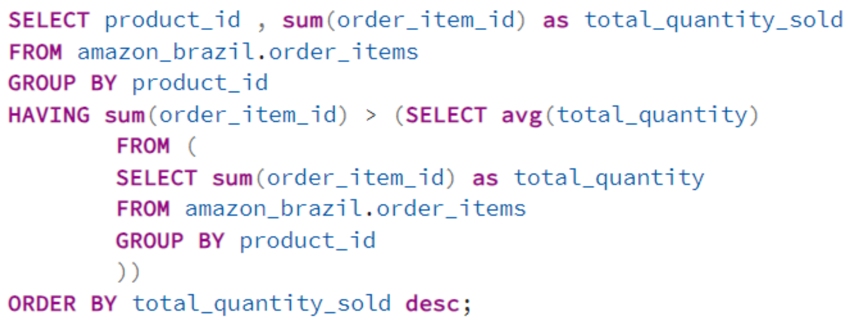
	season text	total_sales numeric
1	Spring	4216721.54
2	Summer	4120359.62
3	Winter	2905750.03
4	Autumn	2348812.51

Analysis: Helps in seasonal inventory planning.

Recommendation: Stock up on seasonal products accordingly.

Task 2 : Identifying High-Volume Products

Query: Filter products with a total quantity sold above the average.

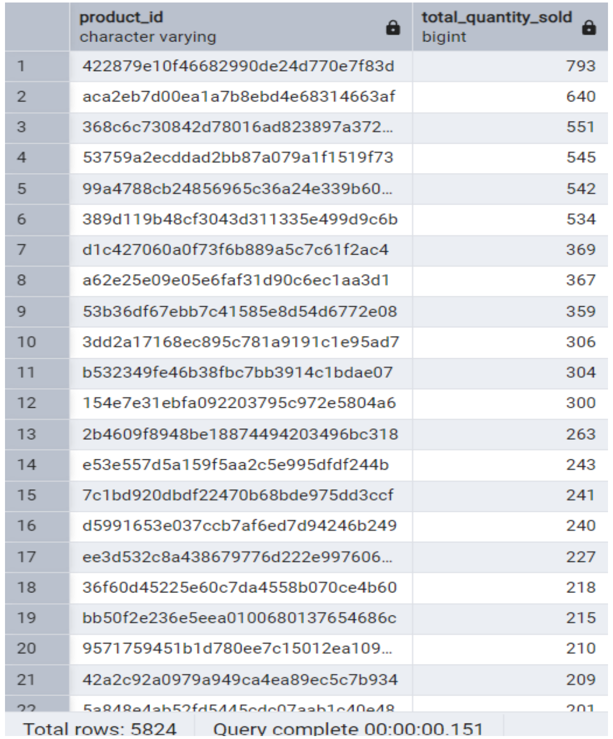


```sql
SELECT product_id , sum(order_item_id) as total_quantity_sold
FROM amazon_brazil.order_items
GROUP BY product_id
HAVING sum(order_item_id) > (SELECT avg(total_quantity)
        FROM (
        SELECT sum(order_item_id) as total_quantity
        FROM amazon_brazil.order_items
        GROUP BY product_id
        ))
ORDER BY total_quantity_sold desc;
```

Analysis: Highlights top-performing products.

Recommendation: Increase stock and promotions for these products.

Output:

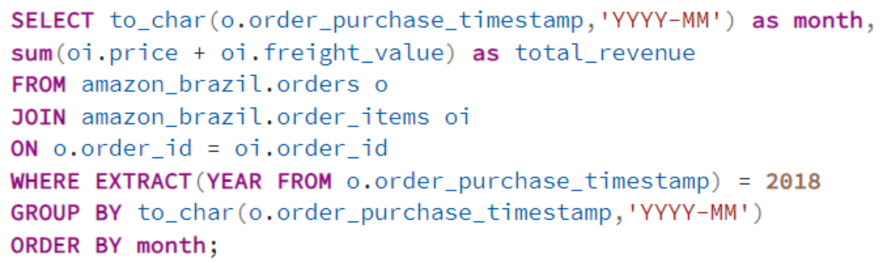


	product_id character varying	total_quantity_sold bigint
1	422879e10f46682990de24d770e7f83d	793
2	aca2eb7d00ea1a7b8ebd4e68314663af	640
3	368c6c730842d78016ad823897a372...	551
4	53759a2ecddad2bb87a079a1f1519f73	545
5	99a4788cb24856965c36a24e339b60...	542
6	389d119b48cf3043d311335e499d9c6b	534
7	d1c427060a0f73f6b889a5c7c61f2ac4	369
8	a62e25e09e05e6faf31d90c6ec1aa3d1	367
9	53b36df67ebb7c41585e8d54d6772e08	359
10	3dd2a17168ec895c781a9191c1e95ad7	306
11	b532349fe46b38fbc7bb3914c1bdae07	304
12	154e7e31ebfa092203795c972e5804a6	300
13	2b4609f8948be18874494203496bc318	263
14	e53e557d5a159f5aa2c5e995dfdf244b	243
15	7c1bd920dbdf22470b68bde975dd3ccf	241
16	d5991653e037ccb7af6ed7d94246b249	240
17	ee3d532c8a438679776d222e997606...	227
18	36f60d45225e60c7da4558b070ce4b60	218
19	bb50f2e236e5eea0100680137654686c	215
20	9571759451b1d780ee7c15012ea109...	210
21	42a2c92a0979a949ca4ea89ec5c7b934	209
22	5a848e4ab52fd5445cdc07aab1c40e48	201

Total rows: 5824   Query complete 00:00:00.151

Task 3 : Monthly Revenue Trends

Query: Calculate total revenue per month in 2018

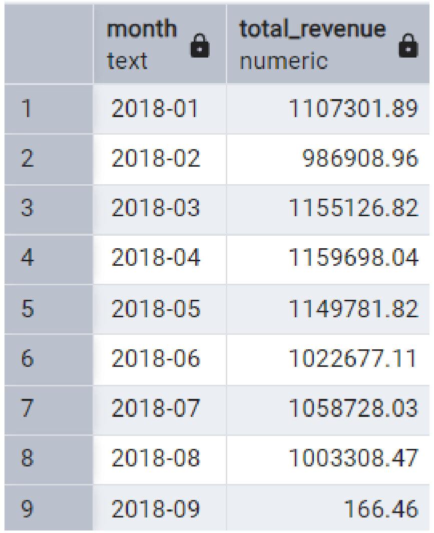


```
SELECT to_char(o.order_purchase_timestamp,'YYYY-MM') as month,
sum(oi.price + oi.freight_value) as total_revenue
FROM amazon_brazil.orders o
JOIN amazon_brazil.order_items oi
ON o.order_id = oi.order_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
GROUP BY to_char(o.order_purchase_timestamp,'YYYY-MM')
ORDER BY month;
```

Analysis: Detects revenue trends.

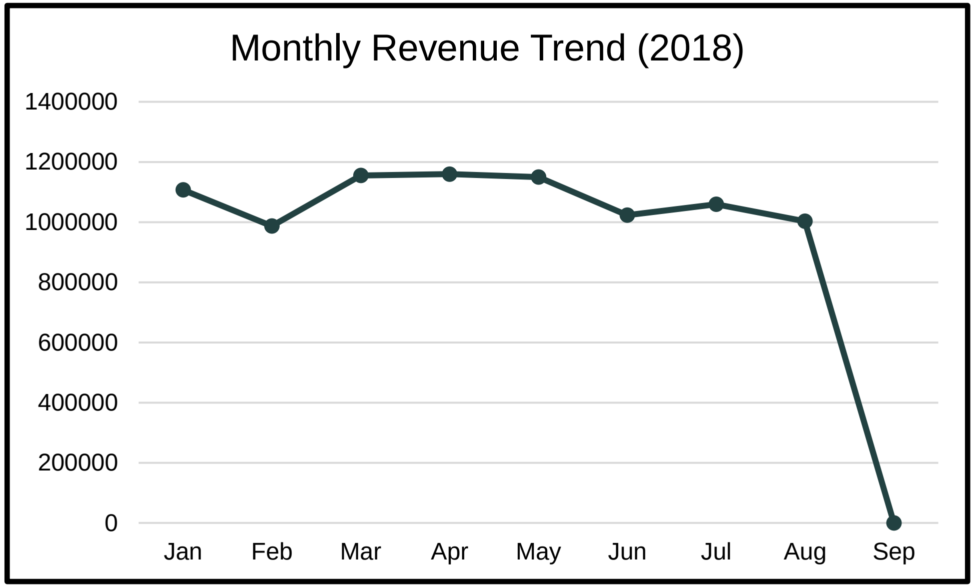
Recommendation: Align marketing campaigns with revenue trends.

Output:



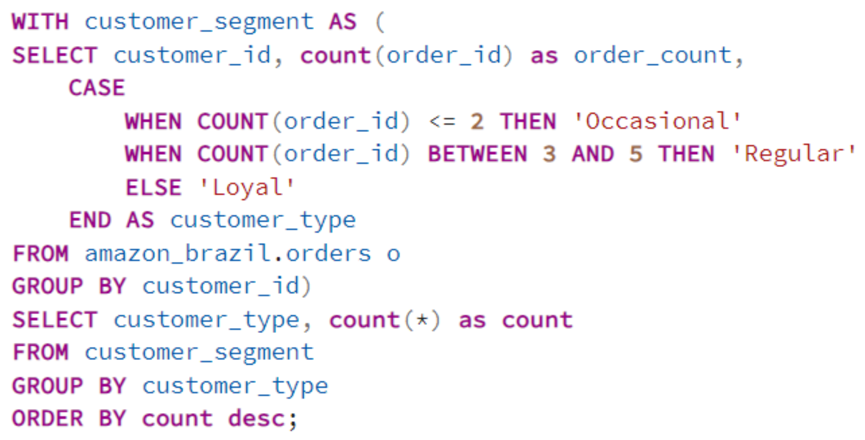
	month text	total_revenue numeric
1	2018-01	1107301.89
2	2018-02	986908.96
3	2018-03	1155126.82
4	2018-04	1159698.04
5	2018-05	1149781.82
6	2018-06	1022677.11
7	2018-07	1058728.03
8	2018-08	1003308.47
9	2018-09	166.46

Identifying Seasonal Revenue Patterns and Business Implications



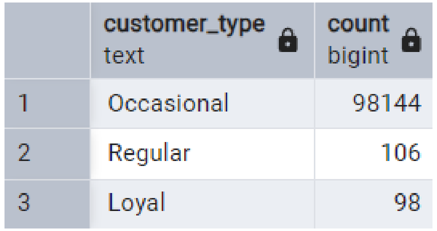
Task 4 : Customer Loyalty Segmentation

Query: Classify customers based on order frequency into Occasional, Regular, and Loyal.



```
WITH customer_segment AS (
SELECT customer_id, count(order_id) as order_count,
    CASE
        WHEN COUNT(order_id) <= 2 THEN 'Occasional'
        WHEN COUNT(order_id) BETWEEN 3 AND 5 THEN 'Regular'
        ELSE 'Loyal'
    END AS customer_type
FROM amazon_brazil.orders o
GROUP BY customer_id)
SELECT customer_type, count(*) as count
FROM customer_segment
GROUP BY customer_type
ORDER BY count desc;
```

Output:



	customer_type text	count bigint
1	Occasional	98144
2	Regular	106
3	Loyal	98

Analysis: Enhances customer retention strategies.

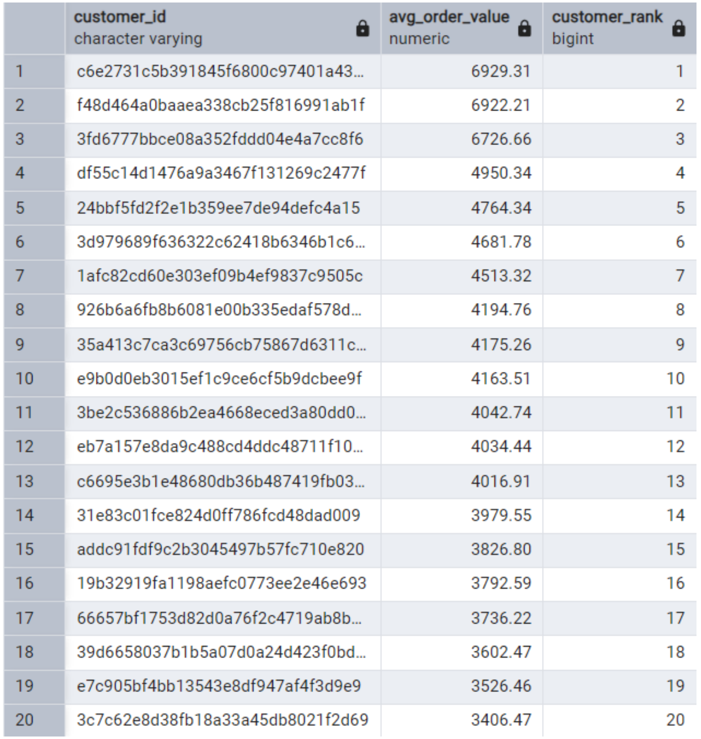
Recommendation: Design personalized loyalty rewards for each segment.

Task 5 : High-Value Customers Identification

Query: Rank top 20 customers based on average order value.

```sql
WITH high_value_customers AS (
SELECT o.customer_id ,
round(avg(oi.price + oi.freight_value),2) as avg_order_value,
dense_rank() over(order by avg(oi.price + oi.freight_value) desc)
AS customer_rank
FROM amazon_brazil.orders o
JOIN amazon_brazil.order_items oi
ON o.order_id = oi.order_id
GROUP BY customer_id )
SELECT * FROM high_value_customers
WHERE customer_rank <=20
ORDER BY avg_order_value desc;
```

Output:



	customer_id character varying	avg_order_value numeric	customer_rank bigint
1	c6e2731c5b391845f6800c97401a43...	6929.31	1
2	f48d464a0baaea338cb25f816991ab1f	6922.21	2
3	3fd6777bbce08a352fddd04e4a7cc8f6	6726.66	3
4	df55c14d1476a9a3467f131269c2477f	4950.34	4
5	24bbf5fd2f2e1b359ee7de94defc4a15	4764.34	5
6	3d979689f636322c62418b6346b1c6...	4681.78	6
7	1afc82cd60e303ef09b4ef9837c9505c	4513.32	7
8	926b6a6fb8b6081e00b335edaf578d...	4194.76	8
9	35a413c7ca3c69756cb75867d6311c...	4175.26	9
10	e9b0d0eb3015ef1c9ce6cf5b9dcbee9f	4163.51	10
11	3be2c536886b2ea4668eced3a80dd0...	4042.74	11
12	eb7a157e8da9c488cd4ddc48711f10...	4034.44	12
13	c6695e3b1e48680db36b487419fb03...	4016.91	13
14	31e83c01fce824d0ff786fcd48dad009	3979.55	14
15	addc91fdf9c2b3045497b57fc710e820	3826.80	15
16	19b32919fa1198aefc0773ee2e46e693	3792.59	16
17	66657bf1753d82d0a76f2c4719ab8b...	3736.22	17
18	39d6658037b1b5a07d0a24d423f0bd...	3602.47	18
19	e7c905bf4bb13543e8df947af4f3d9e9	3526.46	19
20	3c7c62e8d38fb18a33a45db8021f2d69	3406.47	20

Analysis: Identifies top-spending customers.

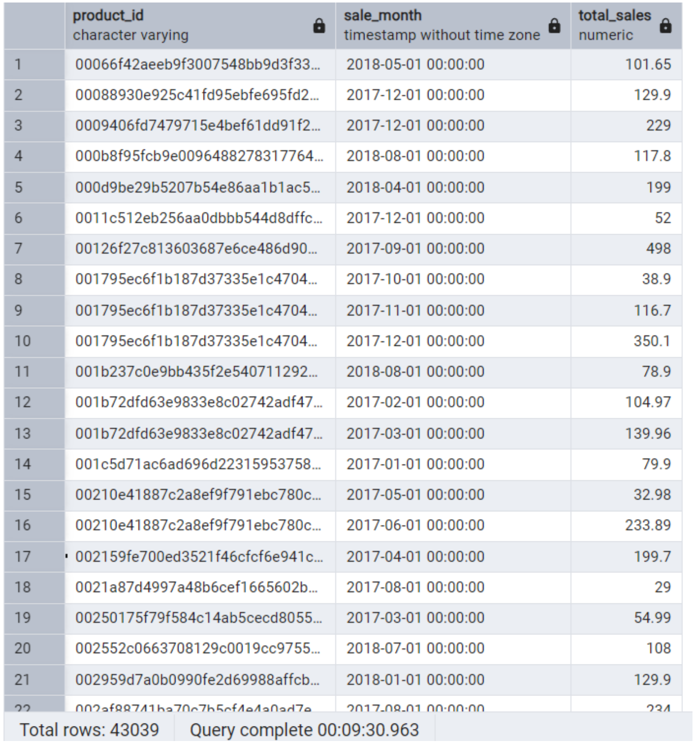
Recommendation: Provide exclusive offers to high-value customers.

Task 6 : Monthly Cumulative Sales per Product

Query: Calculate cumulative sales for each product month by month.

```sql
WITH RECURSIVE sales_data AS (
    SELECT oi.product_id,
    DATE_TRUNC('month', o.order_purchase_timestamp) AS sale_month,
    SUM(oi.price) AS monthly_sales
    FROM amazon_brazil.order_items oi
    JOIN amazon_brazil.orders o ON oi.order_id = o.order_id
    GROUP BY oi.product_id, sale_month),
recursive_sales AS (
    SELECT s.product_id, s.sale_month,
    s.monthly_sales AS total_sales
    FROM sales_data s
    WHERE s.sale_month = (
        SELECT MIN(s2.sale_month)
        FROM sales_data s2
        WHERE s2.product_id = s.product_id)
    UNION ALL
    SELECT s.product_id, s.sale_month,
        rs.total_sales + s.monthly_sales
    FROM sales_data s
    JOIN recursive_sales rs
    ON s.product_id = rs.product_id
    AND s.sale_month = rs.sale_month + INTERVAL '1 month')
SELECT product_id, sale_month, total_sales
FROM recursive_sales
ORDER BY product_id, sale_month;
```

Output:



	product_id character varying	sale_month timestamp without time zone	total_sales numeric
1	00066f42aeeb9f3007548bb9d3f33...	2018-05-01 00:00:00	101.65
2	00088930e925c41fd95ebfe695fd2...	2017-12-01 00:00:00	129.9
3	0009406fd7479715e4bef61dd91f2...	2017-12-01 00:00:00	229
4	000b8f95fcb9e0096488278317764...	2018-08-01 00:00:00	117.8
5	000d9be29b5207b54e86aa1b1ac5...	2018-04-01 00:00:00	199
6	0011c512eb256aa0dbbb544d8dffc...	2017-12-01 00:00:00	52
7	00126f27c813603687e6ce486d90...	2017-09-01 00:00:00	498
8	001795ec6f1b187d37335e1c4704...	2017-10-01 00:00:00	38.9
9	001795ec6f1b187d37335e1c4704...	2017-11-01 00:00:00	116.7
10	001795ec6f1b187d37335e1c4704...	2017-12-01 00:00:00	350.1
11	001b237c0e9bb435f2e540711292...	2018-08-01 00:00:00	78.9
12	001b72dfd63e9833e8c02742adf47...	2017-02-01 00:00:00	104.97
13	001b72dfd63e9833e8c02742adf47...	2017-03-01 00:00:00	139.96
14	001c5d71ac6ad696d22315953758...	2017-01-01 00:00:00	79.9
15	00210e41887c2a8ef9f791ebc780c...	2017-05-01 00:00:00	32.98
16	00210e41887c2a8ef9f791ebc780c...	2017-06-01 00:00:00	233.89
17	002159fe700ed3521f46cfcf6e941c...	2017-04-01 00:00:00	199.7
18	0021a87d4997a48b6cef1665602b...	2017-08-01 00:00:00	29
19	00250175f79f584c14ab5cecd8055...	2017-03-01 00:00:00	54.99
20	002552c0663708129c0019cc9755...	2018-07-01 00:00:00	108
21	002959d7a0b0990fe2d69988affcb...	2018-01-01 00:00:00	129.9
22	002af88741ba70c7b5cf4e4a0ad7e...	2017-08-01 00:00:00	234

Total rows: 43039 Query complete 00:09:30.963

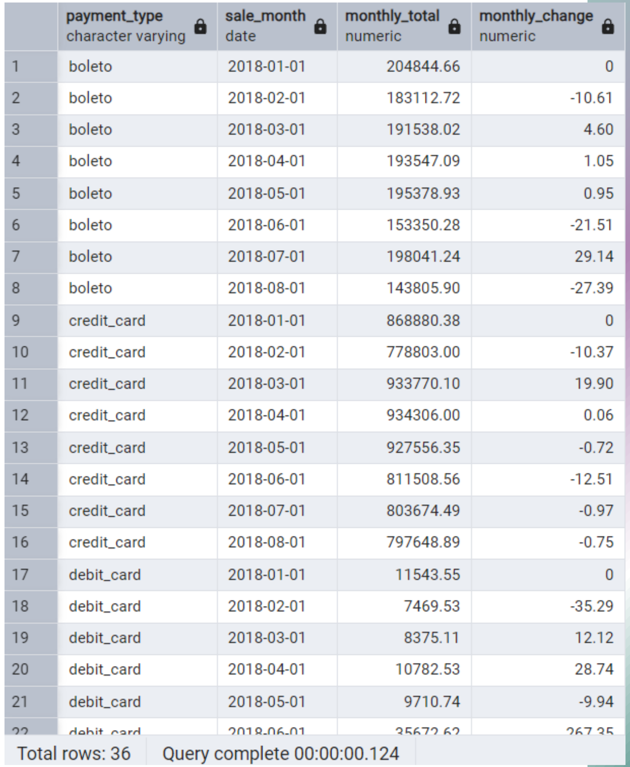
Analysis: Tracks product sales trends over time.
Recommendation: Optimize inventory and sales strategies based on trends.

Task 7 : Payment Methods and Monthly Sales Growth

Query: Compute total monthly sales per payment method and calculate month-over-month growth.

```sql
WITH MonthlySales AS (
SELECT p.payment_type,
DATE_TRUNC('month', o.order_purchase_timestamp)::DATE AS sale_month,
SUM(p.payment_value) AS monthly_total
FROM amazon_brazil.orders o
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),
SalesWithChange AS (
SELECT ms.payment_type, ms.sale_month, ms.monthly_total,
LAG(ms.monthly_total) OVER (PARTITION BY ms.payment_type ORDER BY ms.sale_month) AS prev_month_sales,
ROUND(
((ms.monthly_total - LAG(ms.monthly_total) OVER (PARTITION BY ms.payment_type ORDER BY ms.sale_month))
/ NULLIF(LAG(ms.monthly_total) OVER (PARTITION BY ms.payment_type ORDER BY ms.sale_month), 0)) * 100, 2
) AS monthly_change
FROM MonthlySales ms)
SELECT payment_type, sale_month, monthly_total, COALESCE(monthly_change, 0) AS monthly_change
FROM SalesWithChange
ORDER BY payment_type, sale_month;
```

Output:



	payment_type character varying	sale_month date	monthly_total numeric	monthly_change numeric
1	boleto	2018-01-01	204844.66	0
2	boleto	2018-02-01	183112.72	-10.61
3	boleto	2018-03-01	191538.02	4.60
4	boleto	2018-04-01	193547.09	1.05
5	boleto	2018-05-01	195378.93	0.95
6	boleto	2018-06-01	153350.28	-21.51
7	boleto	2018-07-01	198041.24	29.14
8	boleto	2018-08-01	143805.90	-27.39
9	credit_card	2018-01-01	868880.38	0
10	credit_card	2018-02-01	778803.00	-10.37
11	credit_card	2018-03-01	933770.10	19.90
12	credit_card	2018-04-01	934306.00	0.06
13	credit_card	2018-05-01	927556.35	-0.72
14	credit_card	2018-06-01	811508.56	-12.51
15	credit_card	2018-07-01	803674.49	-0.97
16	credit_card	2018-08-01	797648.89	-0.75
17	debit_card	2018-01-01	11543.55	0
18	debit_card	2018-02-01	7469.53	-35.29
19	debit_card	2018-03-01	8375.11	12.12
20	debit_card	2018-04-01	10782.53	28.74
21	debit_card	2018-05-01	9710.74	-9.94
22	debit_card	2018-06-01	35672.62	267.35

Total rows: 36 Query complete 00:00:00.124

Analysis: Shows how payment methods affect sales trends.

Recommendation: Promote high-performing payment methods for sustained growth.

Conclusion

This analysis provides critical insights into Amazon India's sales, payment trends, customer behaviors, and product performances. Implementing these recommendations will enhance efficiency, boost sales, and optimize customer experience.