Dataset: Retail_Transaction_Dataset.csv

Context: Retail and Consumer Behavior

The objective of this assessment is to organize, explore, and transform transactional data to produce **actionable, reproducible, and technically sound analyses.**
I performed **cohort analysis, time-series exploration, window functions, and textual data processing** to identify trends in consumer behavior and product performance.

Skills demonstrated: SQL analysis, cohort analysis, window functions, time-series, text analysis
Language: SQL

Question 1 - Preparation for analysis

1.1 Considering that this is a single CSV file, we are dealing here with only one table. In this we can identify some columns, the main ones among them and their relationships are:

- "Product ID" and "ProductCategory" - link to the product portfolio. We can analyze which category has the most output;
- "TransactionDate" - allows temporal and seasonal analysis, identifying the period where more purchases and fewer purchases are made;
- "StoreLocation" - allows you to analyze if there are stores that sell more/less, checking the number of purchases made.

1.2 After analyzing the price ranges, we can detect that most of the products sold are concentrated in the range of values between 50 and 99.99, adding up to a total of 55.71%. We can also note that the range of values between 10 and 49.99 is also well represented, with a total of 44.27%.

1.3 In view of the previous question, the prices were placed in the categories indicated.

1.4 By viewing this pair of columns "ProductCategory" and "PaymentMethod" using PARTITION BY and adding the total payment method per product category, we can conclude that there are four payment methods and that these are present in all categories almost uniformly, not having a preference for method.

1.5 After verifying that the maximum quantities sold per CostumerID present in the Dataset is 9, "low volume" transactions were defined representing 1 to 3 items, typical of individual consumption; "medium volume" of 4 to 6 items, an intermediate purchase, which can be seen as familiar, and finally "high volume" of 7 to 9 items, indicating customers with greater purchasing power.

```
CREATE TABLE Retail_Transaction AS SELECT * FROM read_csv_auto('Retail_Transaction_Dataset.csv', HEADER=TRUE); SELECT * F
LIMIT 10;
```

	CustomerID int64 109318 - 993229	Productid object D 40% The 30% 2 others 30%	Quantity int64 3 - 8	Price float64 13.12193739 - 98....	TransactionDate o. 12/26/2023... 10% 8/5/2023 0... 10% 8 others 80%	PaymentMethod o. Cash 50% PayPal 20% 2 others 30%	StoreLocation ob... 176 Andrew... 10% 11635 Willia... 10% 8 others 80%	P B E 2
0	109318	C	7	80.07984415	12/26/2023 12:32	Cash	176 Andrew Cliffs...	B
1	993229	C	4	75.19522942	8/5/2023 0:00	Cash	11635 William Wel...	H
2	579675	The	8	31.52881648	3/11/2024 18:51	Cash	910 Mendez Ville ...	B
3	799826	D	5	98.88021828	10/27/2023 22:00	PayPal	87522 Sharon Co...	B
4	121413	The	7	93.18851246	12/22/2023 11:38	Cash	0070 Michelle Isl...	E
5	463050	D	3	54.09315249	8/15/2023 4:24	Cash	8492 Jonathan D...	E
6	888163	D	7	13.12193739	12/26/2023 5:32	PayPal	USNV Harrell FP...	C
7	843385	The	8	56.02516419	10/11/2023 6:48	Debit Card	489 Juan Loop A...	H
8	839609	B	5	23.85798105	2/27/2024 11:13	Credit Card	528 Justin Expre...	E
9	184135	D	4	63.3427768	11/5/2023 1:46	Debit Card	189 Wright Mews...	B

10 rows, 10 cols | 10 / page | Page 1 of 1

```
-- Question 1.1 : Below are indicated the types of each column through the "PRAGMA" instruction at the top of the table
```

	Cid int32 0 - 9	name object CustomerID 10% Productid 10% 8 others 80%	type object VARCHAR 50% DOUBLE 30% BIGINT 20%	notnull Bool False 100%	dflt_value object Missing 100%	Pk Bool False 100%
0	0	CustomerID	BIGINT	False	None	False
1	1	Productid	VARCHAR	False	None	False
2	2	Quantity	BIGINT	False	None	False
3	3	Price	DOUBLE	False	None	False
4	4	TransactionDate	VARCHAR	False	None	False
5	5	PaymentMethod	VARCHAR	False	None	False
6	6	StoreLocation	VARCHAR	False	None	False
7	7	ProductCategory	VARCHAR	False	None	False
8	8	DiscountApplied...	DOUBLE	False	None	False
9	9	TotalAmount	DOUBLE	False	None	False

10 rows, 6 cols 10 / page Page 1 of 1

```
SELECT CASE WHEN Price < 10 THEN '0 - 9.99' WHEN Price BETWEEN 10 AND 49.99 THEN 'Baixo_custo' WHEN Price BETWEEN 50 AND
 ROUND(100.0 * COUNT(*) / (SELECT COUNT(*) FROM Retail_Transaction), 2) AS Percentage FROM Retail_Transaction GROUP BY Pr
```

	PriceRange object	TotalProducts int...	Percentage float...
0	Baixo_custo	44266	44.27
1	Medio_custo	55714	55.71
2	Alto_custo	20	0.02

3 rows, 3 cols 10 / page Page 1 of 1

```
SELECT ProductCategory,
 PaymentMethod, COUNT(*) AS TotalTransactions,
 ROUND(100.0 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY ProductCategory), 2) AS PercentagePer Category FROM Retail_Tra
```

	ProductCategory o Books 25% Clothing 25% 2 others 50%	PaymentMethod o. PayPal 25% Credit Card 25% 2 others 50%	Total Transactions 6138 - 6345	PercentageByCa... 24.54 - 25.33
0	Books	PayPal	6341	25.33
1	Books	Credit Card	6256	24.99
2	Books	Cash	6243	24.94
3	Books	Debit Card	6191	24.73
4	Clothing	Credit Card	6345	25.32
5	Clothing	PayPal	6285	25.08
6	Clothing	Cash	6278	25.06
7	Clothing	Debit Card	6148	24.54
8	Electronics	Credit Card	6295	25.19
9	Electronics	Cash	6282	25.14
10	Electronics	Debit Card	6240	24.97
11	Electronics	PayPal	6173	24.7
12	Home Decor	Debit Card	6312	25.33
13	Home Decor	PayPal	6268	25.15
14	Home Decor	Cash	6205	24.9
15	Home Decor	Credit Card	6138	24.63

16 rows, 4 cols 50 / page Page 1 of 1

```
SELECT MAX(Quantity) AS Max_Quantity FROM Retail_Transaction; SELECT CustomerID,
 Productid
 Quantity,  CASE WHEN Quantity BETWEEN 1 AND 3 THEN 'Low Volume' WHEN Quantity BETWEEN 4 AND 6 THEN 'Medium Volume' WHEN
 Price
 TotalAmount FROM Retail_Transaction
LIMIT 20;
```

	CustomerID int64 26863 - 993229	Productid object D 45% B 25% 2 others 30%	Quantity int64 2 - 8	VolumeCategory o. High Volume 40% Medium Vol... 40% Low Volume 20%	Price float64 13.12193739 - 98....	TotalAmount floa... 34.06846168 - 6...
0	109318	C	7	High Volume	80.07984415	455.8627638
1	993229	C	4	Medium Volume	75.19522942	258.3065464
2	579675	The	8	High Volume	31.52881648	212.0156509
3	799826	D	5	Medium Volume	98.88021828	461.3437694
4	121413	The	7	High Volume	93.18851246	626.0304837
5	463050	D	3	Low Volume	54.09315249	144.6092233
6	888163	D	7	High Volume	13.12193739	76.88590745
7	843385	The	8	High Volume	56.02516419	419.7660521
8	839609	B	5	Medium Volume	23.85798105	96.97792465
9	184135	D	4	Medium Volume	63.3427768	234.0120175

20 rows, 6 cols 10 / page Page 1 of 2

Questão 2 - Padrões com funções de janela

2.1 Considerando que nese Dataset não existe uma coluna que indique o país e/ou o estado explicitamente, a análise foi feita em cima da coluna "CustomerID" e "ProductCategory".

2.2 Através deste código e do seu resultado podemos visualizar quais clientes fizeram mais que uma compra e o seu engajamento, dependendo se a média está aumentando ou diminuindo conforme suas mais recentes compras.

2.3 Considerando que nese Dataset não existe uma coluna que indique o país e/ou o estado explicitamente, a análise foi feita em cima da coluna "TotalAmount" e "ProductCategory". Analisando o dataframe criado, podemos ver quais os clientes que mais consumiram em cada categoria de produto.

2.4 Neste código foi definida uma janela reutilizável cuja expressão "w" foi utilizada em uma consulta com RANK() e SUM() para identificar os clientes que mais consumiram de cada categoria.

2.5 Neste código foi utilizada a função WINDOW com a expressão "w" para deixar o código mais limpo e o CASE WHEN com o LAG() para classificar a relação compra atual x compra anterior como "Primeira Compra", "Redução" ou "Crescimento" para cada CustomerID. Pelo fato de a grande maioria dos clientes fazerem apenas uma compra, a classificação (quase que de forma homogênea) fica como "Primeira Compra".

```
SELECT CustomerID,
 ProductCategory, COUNT(*) AS TotalOrders, SUM(TotalAmount) AS TotalAmountSpent, SUM(Quantity) AS TotalItems FROM Retail_
```

	CustomerID int64 32895 - 960650	ProductCategory o Clothing 30% Electronics 30% 2 others 40%	TotalPedidos int64 2 - 3	ValorTotalGasto f... 1150.038094999...	ItensTotais float64 14.0 - 22.0
0	903169	Books	2	1563.85116	17
1	780013	Clothing	2	1519.193458	17
2	823783	Clothing	3	1441.329882	22
3	887487	Home Decor	2	1352.743062	16
4	598581	Home Decor	2	1342.070296	17
5	536978	Clothing	2	1327.266128	15
6	400993	Electronics	2	1314.806049	18
7	32895	Electronics	2	1291.913188	15
8	895452	Electronics	2	1241.740674	15
9	465262	Electronics	2	1222.491943	18

20 rows, 5 cols 10 / page Page 1 of 2

```sql
CREATE TABLE Retail_Transaction_Clean AS
SELECT
    CustomerID,
    ProductID,
    Quantity,
    Price,
    MAKE_TIMESTAMP(
        CAST(SPLIT_PART(SPLIT_PART(TransactionDate, ' ', 1), '/', 3) AS INTEGER), -- Ano
        CAST(SPLIT_PART(SPLIT_PART(TransactionDate, ' ', 1), '/', 1) AS INTEGER), -- Mês
        CAST(SPLIT_PART(SPLIT_PART(TransactionDate, ' ', 1), '/', 2) AS INTEGER), -- Dia
        CAST(SPLIT_PART(SPLIT_PART(TransactionDate, ' ', 2), ':', 1) AS INTEGER), -- Hora
        CAST(SPLIT_PART(SPLIT_PART(TransactionDate, ' ', 2), ':', 2) AS INTEGER), -- Minuto
        0 -- Segundos
    ) AS TransactionDate,
    PaymentMethod,
    StoreLocation,
    ProductCategory,
    'DiscountApplied(%)',
    TotalAmount
FROM Retail_Transaction;
```

	Count int64
0	100000

1 row, 1 col 10 / page Page 1 of 1

```sql
SELECT
    CustomerID,
    TransactionDate,
    TotalAmount,
    ROUND(
        AVG(TotalAmount) OVER (
            PARTITION BY CustomerID
            ORDER BY TransactionDate
            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
        ), 2
    ) AS MediaMovel3Compras
FROM Retail_Transaction_Clean
ORDER BY CustomerID, TransactionDate;
```

	CustomerID int64 14 - 999997	TransactionDate d.	TotalAmount floa...	MediaMovel3Co...
0	14	2023-08-06 06:4...	256.2327913	256.23
1	42	2023-05-19 21:5...	502.6565234	502.66
2	49	2023-06-05 13:1...	21.39904723	21.4
3	59	2023-08-19 03:5...	139.6120364	139.61
4	59	2024-04-01 01:0...	109.8806596	124.75
5	65	2023-06-18 10:2...	548.0066252	548.01
6	87	2023-08-21 15:2...	41.51590547	41.52
7	96	2024-04-06 12:3...	194.356816	194.36
8	98	2023-12-20 11:49...	166.9346467	166.93
9	100	2023-05-13 17:17...	710.0625762	710.06

100,000 rows, 4 cols 10 / page Page 1 of 10,000

```sql
SELECT
    ProductCategory,
    CustomerID,
    SUM(TotalAmount) AS ValorTotalGasto,
    RANK() OVER(
        PARTITION BY ProductCategory
        ORDER BY SUM(TotalAmount) DESC
    ) AS PosicaoRanking
FROM Retail_Transaction_Clean
GROUP BY ProductCategory, CustomerID
QUALIFY PosicaoRanking <= 3
ORDER BY ProductCategory, PosicaoRanking;
```

	ProductCategory o Books 25% Clothing 25% 2 others 50%	CustomerID int64 32895 - 903169	ValorTotalGasto f... 1165.9135849 - 1...	PosicaoRanking i... 1 - 3
0	Books	903169	1563.85116	1
1	Books	326398	1185.374449	2
2	Books	392763	1165.913585	3
3	Clothing	780013	1519.193458	1
4	Clothing	823783	1441.329882	2
5	Clothing	536978	1327.266128	3
6	Electronics	400993	1314.806049	1
7	Electronics	32895	1291.913188	2
8	Electronics	895452	1241.740674	3
9	Home Decor	887487	1352.743062	1

12 rows, 4 cols 10 / page Page 1 of 2

```sql
WITH Totais AS (
    SELECT
        CustomerID,
        ProductCategory,
        SUM(TotalAmount) AS ValorTotalGasto
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID, ProductCategory
)
SELECT
    CustomerID,
    ProductCategory,
    ValorTotalGasto,
    RANK() OVER w AS RankingCliente,
    SUM(ValorTotalGasto) OVER w AS SomaCategoria,
    ROUND(
        100.0 * ValorTotalGasto / SUM(ValorTotalGasto) OVER (PARTITION BY ProductCategory),
        2
    ) AS PercentualCategoria
FROM Totais
WINDOW w AS (
    PARTITION BY ProductCategory
    ORDER BY ValorTotalGasto DESC
)
ORDER BY ProductCategory, RankingCliente;
```

	CustomerID int64 14 - 999997	ProductCategory o	ValorTotalGasto f...	RankingCliente i...	SomaCategoria fl...	PercentualCateg...
0	903169	Books	1563.85116	1	1563.85116	0.02
1	326398	Books	1185.374449	2	2749.225608	0.02
2	392763	Books	1165.913585	3	3915.139193	0.02
3	39999	Books	1148.629689	4	5063.768882	0.02
4	980072	Books	1136.449742	5	6200.218624	0.02
5	206249	Books	1130.104536	6	7330.32316	0.02
6	312478	Books	1123.582595	7	8453.905756	0.02
7	837176	Books	1075.012673	8	9528.918429	0.02
8	143058	Books	1063.256943	9	10592.17537	0.02
9	591305	Books	1057.207556	10	11649.38293	0.02

98,810 rows, 6 cols 10 / page Page 1 of 9,881

```sql
SELECT
    CustomerID,
    TransactionDate,
    TotalAmount,
    LAG(TotalAmount) OVER w AS CompraAnterior,
    CASE
        WHEN LAG(TotalAmount) OVER w IS NULL
            THEN 'Primeira Compra'
        WHEN TotalAmount > LAG(TotalAmount) OVER w
            THEN 'Crescimento'
        WHEN TotalAmount < LAG(TotalAmount) OVER w
            THEN 'Redução'
        ELSE 'Estável'
    END AS Tendencia
FROM Retail_Transaction_Clean
WINDOW w AS (
    PARTITION BY CustomerID
    ORDER BY TransactionDate
)
ORDER BY CustomerID, TransactionDate
LIMIT 50;
```

	CustomerID int64 14 - 514	TransactionDate d. 2023-05-11 22:18...	TotalAmount floa... 13.25847646 - 71...	CompraAnterior f... 139.6120364 - 4...	Tendencia object Primeira Co... 94% Redução 6%
40	393	2024-03-14 00:5...	99.08964269	Nan	Primeira Compra
41	399	2024-02-11 08:1...	109.7249415	Nan	Primeira Compra
42	402	2023-12-16 08:2...	48.18508029	Nan	Primeira Compra
43	415	2023-10-12 19:25...	66.12504916	Nan	Primeira Compra
44	417	2023-09-23 09:0...	65.51688989	Nan	Primeira Compra
45	442	2023-08-30 03:3...	408.1609442	Nan	Primeira Compra
46	442	2024-03-20 08:2...	13.25847646	408.1609442	Redução
47	474	2023-10-27 08:3...	288.8941216	Nan	Primeira Compra
48	475	2023-11-20 11:27:...	554.7768083	Nan	Primeira Compra
49	514	2023-08-09 17:4...	285.4230248	Nan	Primeira Compra

Questão 3 - Séries temporais

3.1 Análise de informações temporais a partir do EXTRACT() feito em cima dos meses, dias da semana e hora:

- Meses: podemos observar que o mês com menos pedidos e menos receita foi Abril, enquanto o com mais pedidos foi Junho.
- Dia da Semana: considerando que o DuckDB contabiliza os dias da semana de 0 a 6, sendo 0=Domingo e 6=Sábado, podemos ver que o dia da semana com mais saída de produtos/maior lucro é sábado e o dia com menos saída/maior lucro é terça-feira.
- Hora: a hora com mais saída de pedidos é 19h, entretanto com mais lucro é 17h, indicando que neste último os valores dos produtos e/ou a quantidade de produtos por pedido foi maior do que das 19h.

3.2 A média de dias entre compras consecutivas por cada cliente varia de 0 a 343 days. A partir desse insight, podemos fazer futuras análises para descobrir em quais dias exatos os clientes compraram, se há um mês que tais clientes compraram mais, quantas compras foram realizadas em um todo por cada cliente e dessa forma gerar estratégias de alavancagem por intermédio de campanhas de marketing.

3.3 Considerando que temos somente esta tabela como referência, a análise foi feita em cima das colunas "TotalAmount", "Quantity" e foi criada uma terceira coluna "DisccountApplied_Derived" para verificar se foi aplicado algum desconto ou não em cima do valor total. Esta alternativa foi criada devido ao problema com o nome da coluna x DeepNote (que não interpreta o % corretamente). A linha do tempo foi aplicada para cada Customer da tabela.

3.4 A partir desta query, podemos verificar que há algumas semanas com quantidade de pedidos acima de 2000 como: 2024-01-29, 2023-10-30, 2023-07-31, o que pode indicar alguma promoção, campanha ou até mesmo mudança de salário/comportamento como 13 salário, férias... Além disso podemos notar que estas datas caem no final do mês, o que coincide com a data de pagamento de muitas empresas. Através desta análise, também podemos notar que há uma dinâmica regular de pedidos feitos, que rondam os 1800-1900, com exceção da primeira semana, com um total de 294 pedidos feitos.

3.5 Esta query complementa a última, tendo as maiores variações percentuais relacionadas à primeira semana e as datas 2024-01-29, 2023-10-30, 2023-07-31. Este tipo de análise é importante para saber como as vendas estão acontecendo, qual a margem de lucro e se há muita variação entre uma semana e outra.

3.6 A aplicação desta janela deslizante permite comparações semana a semana e reduz o impacto de variações pontuais causadas por promoções ou datas específicas.

3.7 Através desta query podemos ter um apanhado geral do volume de vendas por mês e sua influencia no valor total de um período de um ano a partir da data de início , 2023/04.

3.8 Analisando o resultado podemos concluir que não há muita variação de pedidos/receita de um mês para o outro. Como o dataset é limitado ao período de um ano, não temos a possibilidade de comparar estes mesmos meses em outros anos.
Esta análise quando aplicada em diferentes anos, pode levar a conclusões sobre a própria empresa e a estratégia que está sendo utilizada, se está funcionando ou se deve ser estudada uma nova estratégia.

```
SELECT
    CustomerID,
    TransactionDate,
    EXTRACT(MONTH FROM TransactionDate) AS Mes,
    EXTRACT(DAYOFWEEK FROM TransactionDate) AS DiaSemana,
    EXTRACT(HOUR FROM TransactionDate) AS Hora
FROM Retail_Transaction_Clean
LIMIT 20;
```

	CustomerID int64 26863 - 993229	TransactionDate d. 2023-08-05 00:0...	Mes int64 2 - 12	DiaSemana int64 0 - 6	Hora int64 0 - 22
0	109318	2023-12-26 12:3...	12	2	12
1	993229	2023-08-05 00:0...	8	6	0
2	579675	2024-03-11 18:51...	3	1	18
3	799826	2023-10-27 22:0...	10	5	22
4	121413	2023-12-22 11:38...	12	5	11
5	463050	2023-08-15 04:2...	8	2	4
6	888163	2023-12-26 05:3...	12	2	5
7	843385	2023-10-11 06:4...	10	3	6
8	839609	2024-02-27 11:13...	2	2	11
9	184135	2023-11-05 01:4...	11	0	1

20 rows, 5 cols 10 / page Page 1 of 2

```sql
SELECT
    EXTRACT(HOUR FROM TransactionDate) AS Hora,
    COUNT(*) AS TotalPedidos,
    SUM(TotalAmount) AS ReceitaTotal
FROM Retail_Transaction_Clean
GROUP BY Hora
ORDER BY Hora;
```

	Hora int64 0 - 23	TotalPedidos int64 4008 - 4283	ReceitaTotal floa... 990850.3274371...
10	10	4162	1027157.519
11	11	4180	1042295.832
12	12	4078	1003044.078
13	13	4130	1031290.803
14	14	4142	1039908.796
15	15	4052	990850.3274
16	16	4210	1051154.4
17	17	4186	1049278.258
18	18	4184	1013364.658
19	19	4283	1047779.074

24 rows, 3 cols 10 / page Page 2 of 3

```sql
SELECT
    EXTRACT(MONTH FROM TransactionDate) AS Mes,
    COUNT(*) AS TotalPedidos,
    SUM(TotalAmount) AS ReceitaTotal
FROM Retail_Transaction_Clean
GROUP BY Mes
ORDER BY Mes;
```

	Mes int64 1 - 12	TotalPedidos int64 7921 - 8597	ReceitaTotal floa... 1939190.320200...
0	1	8543	2128345.277
1	2	8076	1973154.117
2	3	8457	2108248.042
3	4	7921	1939190.32
4	5	8388	2099576.101
5	6	8243	2066364.823
6	7	8597	2132550.518
7	8	8498	2109352.646
8	9	8181	2050334.601
9	10	8325	2049450.698

12 rows, 3 cols 10 / page Page 1 of 2

```sql
SELECT
    EXTRACT(DAYOFWEEK FROM TransactionDate) AS DiaSemana,
    COUNT(*) AS TotalPedidos,
    SUM(TotalAmount) AS ReceitaTotal
FROM Retail_Transaction_Clean
GROUP BY DiaSemana
ORDER BY DiaSemana;
```

	DiaSemana int64	TotalPedidos int64	ReceitaTotal floa...
0	0	14291	3535349.551
1	1	14259	3493463.018
2	2	14212	3529847.983
3	3	14288	3540644.812
4	4	14344	3604946.268
5	5	14252	3547941.974
6	6	14354	3581301.9

7 rows, 3 cols 10 / page Page 1 of 1

```sql
WITH Diferencas AS (
    SELECT
        CustomerID,
        TransactionDate,
        LAG(TransactionDate) OVER(
            PARTITION BY CustomerID
            ORDER BY TransactionDate
        ) AS CompraAnterior
    FROM Retail_Transaction_Clean
)
SELECT
    CustomerID,
    ROUND(AVG(DATEDIFF('day', CompraAnterior, TransactionDate)), 2) AS MediaDiasEntreCompras
FROM Diferencas
WHERE CompraAnterior IS NOT NULL
GROUP BY CustomerID
ORDER BY MediaDiasEntreCompras DESC;
```

	CustomerID int64 59 - 999781	MediaDiasEntreC... 0.0 - 363.0
10	861933	343
11	154596	343
12	430135	341
13	657577	341
14	702541	340
15	531812	340
16	920893	339
17	333426	338
18	69986	338
19	3887	338

```sql
WITH base AS (
  SELECT
    CustomerID,
    TransactionDate,
    TotalAmount,
    Quantity,
    Price,
    CASE
      WHEN Quantity * Price IS NULL OR Quantity * Price = 0 THEN NULL
      ELSE 100.0 * (1 - TotalAmount / (Quantity * Price))
    END AS DiscountApplied_Derived
  FROM Retail_Transaction_Clean
)

SELECT
  CustomerID,
  TransactionDate,
  'Compra' AS TipoInteracao,
  TotalAmount
FROM base

UNION ALL

SELECT
  CustomerID,
  TransactionDate,
  'Compra com Desconto' AS TipoInteracao,
  TotalAmount
FROM base

UNION ALL

SELECT
  CustomerID,
  TransactionDate,
  'Grande Volume de Itens' AS TipoInteracao,
  TotalAmount
FROM base
WHERE Quantity >= 7

ORDER BY CustomerID, TransactionDate;
```

	CustomerID int64	TransactionDate d.	TipoInteracao ob...	TotalAmount floa...
0	14	2023-08-06 06:4...	Compra	256.2327913
1	14	2023-08-06 06:4...	Compra com Des...	256.2327913
2	42	2023-05-19 21:5...	Compra	502.6565234
3	42	2023-05-19 21:5...	Compra com Des...	502.6565234
4	42	2023-05-19 21:5...	Grande Volume d...	502.6565234
5	49	2023-06-05 13:1...	Compra	21.39904723
6	49	2023-06-05 13:1...	Compra com Des...	21.39904723
7	59	2023-08-19 03:5...	Compra	139.6120364
8	59	2023-08-19 03:5...	Compra com Des...	139.6120364
9	59	2024-04-01 01:0...	Compra	109.8806596

```sql
SELECT
    DATE_TRUNC('week', TransactionDate) AS Semana,
    COUNT(*) AS TotalPedidos,
    SUM(TotalAmount) AS ReceitaTotal
FROM Retail_Transaction_Clean
GROUP BY 1
ORDER BY Semana;
```

	Semana datetime... 2023-04-24 00:0...	TotalPedidos int64 264 - 2066	ReceitaTotal floa... 60840.78785402...
0	2023-04-24 00:0...	264	60840.78785
1	2023-05-01 00:0...	1931	481984.0841
2	2023-05-08 00:0...	1885	476509.1603
3	2023-05-15 00:0...	1869	458240.8488
4	2023-05-22 00:0...	1914	481252.9245
5	2023-05-29 00:0...	1902	479776.1442
6	2023-06-05 00:0...	1934	474594.4081
7	2023-06-12 00:0...	1913	482094.3013
8	2023-06-19 00:0...	1856	471040.7402
9	2023-06-26 00:0...	1960	490290.3122

53 rows, 3 cols 10 / page Page 1 of 6

```sql
WITH VendasSemanais AS (
    SELECT
        DATE_TRUNC('week', TransactionDate) AS Semana,
        SUM(TotalAmount) AS ReceitaTotal
    FROM Retail_Transaction_Clean
    GROUP BY DATE_TRUNC('week', TransactionDate)
)
SELECT
    Semana,
    ReceitaTotal,
    LAG(ReceitaTotal) OVER (ORDER BY Semana) AS ReceitaAnterior,
    ROUND(
        100.0 * (ReceitaTotal - LAG(ReceitaTotal) OVER (ORDER BY Semana))
        / NULLIF(LAG(ReceitaTotal) OVER (ORDER BY Semana), 0),
        2
    ) AS VariacaoPercentual
FROM VendasSemanais
ORDER BY Semana;
```

	Semana datetime... 2023-04-24 00:0...	ReceitaTotal floa... 60840.78785402...	ReceitaAnterior f... 60840.78785402...	VariacaoPercent... -10.23 - 692.21
50	2024-04-08 00:0...	465631.0291	483742.3118	-3.74
51	2024-04-15 00:0...	475585.1449	465631.0291	2.14
52	2024-04-22 00:0...	453391.0466	475585.1449	-4.67

53 rows, 4 cols 10 / page Page 6 of 6

```sql
WITH VendasDiarias AS (
    SELECT
        DATE_TRUNC('day', TransactionDate) AS Dia,
        SUM(TotalAmount) AS ReceitaTotal
    FROM Retail_Transaction_Clean
    GROUP BY DATE_TRUNC('day', TransactionDate)
)
SELECT
    Dia,
    ReceitaTotal,
    ROUND(
        AVG(ReceitaTotal) OVER(
            ORDER BY Dia
            RANGE BETWEEN INTERVAL 6 DAY PRECEDING AND CURRENT ROW
        ),
        2
    ) AS MediaMovel7d
FROM VendasDiarias
ORDER BY Dia;
```

	Dia datetime64[us] 2023-04-29 00:0...	ReceitaTotal floa... 2241.8692792 - 8...	MediaMovel7d fl... 2241.87 - 74080.98
0	2023-04-29 00:0...	2241.869279	2241.87
1	2023-04-30 00:0...	58598.91857	30420.39
2	2023-05-01 00:0...	62845.5301	41228.77
3	2023-05-02 00:0...	76296.20725	49995.63
4	2023-05-03 00:0...	63935.34683	52783.57
5	2023-05-04 00:0...	70570.06811	55747.99
6	2023-05-05 00:0...	67671.91953	57451.41
7	2023-05-06 00:0...	72797.91436	67530.84
8	2023-05-07 00:0...	67867.09793	68854.87
9	2023-05-08 00:0...	66493.56181	69376.02

366 rows, 3 cols 10 / page Page 1 of 37

```sql
WITH VendasMensais AS (
    SELECT
        DATE_TRUNC('month', TransactionDate) AS Mes,
        SUM(TotalAmount) AS ReceitaMensal
    FROM Retail_Transaction_Clean
    GROUP BY DATE_TRUNC('month', TransactionDate)
)
SELECT
    Mes,
    ReceitaMensal,
    SUM(ReceitaMensal) OVER (
        ORDER BY Mes
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS ReceitaAcumulada
FROM VendasMensais
ORDER BY Mes;
```

	Mes datetime64[... 2023-04-01 00:0...	ReceitaMensal fl... 60840.78785402...	ReceitaAcumulada 60840.78785402...
0	2023-04-01 00:0...	60840.78785	60840.78785
1	2023-05-01 00:0...	2099576.101	2160416.889
2	2023-06-01 00:0...	2066364.823	4226781.712
3	2023-07-01 00:0...	2132550.518	6359332.23
4	2023-08-01 00:0...	2109352.646	8468684.876
5	2023-09-01 00:0...	2050334.601	10519019.48
6	2023-10-01 00:0...	2049450.698	12568470.17
7	2023-11-01 00:0...	2051277.297	14619747.47
8	2023-12-01 00:0...	2125651.066	16745398.54
9	2024-01-01 00:0...	2128345.277	18873743.81

13 rows, 3 cols 10 / page Page 1 of 2

```sql
WITH VendasMensais AS (
    SELECT
        EXTRACT(MONTH FROM TransactionDate) AS Mes,
        SUM(TotalAmount) AS ReceitaMensal,
        COUNT(*) AS Pedidos
    FROM Retail_Transaction_Clean
    WHERE TransactionDate BETWEEN '2023-11-01' AND '2023-12-31'
    GROUP BY Mes
)
SELECT
    SUM(CASE WHEN Mes = 11 THEN ReceitaMensal ELSE 0 END) AS ReceitaNovembro,
    SUM(CASE WHEN Mes = 12 THEN ReceitaMensal ELSE 0 END) AS ReceitaDezembro,
    SUM(CASE WHEN Mes = 11 THEN Pedidos ELSE 0 END) AS PedidosNovembro,
    SUM(CASE WHEN Mes = 12 THEN Pedidos ELSE 0 END) AS PedidosDezembro
FROM VendasMensais;
```

	ReceitaNovembro f	ReceitaDezembro f	PedidosNovembro	PedidosDezembro
0	2051277.297	2059764.983	8298	8206

1 row, 4 cols 10 / page Page 1 of 1

```sql
WITH PrimeiraCompra AS (
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
)
SELECT
    p.CustomerID,
    p.MesEntrada
FROM PrimeiraCompra p
ORDER BY p.MesEntrada, p.CustomerID;

WITH PrimeiraCompra AS (
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
)
SELECT
    MesEntrada,
    COUNT(DISTINCT CustomerID) AS TotalClientesCoorte
FROM PrimeiraCompra
GROUP BY MesEntrada
ORDER BY MesEntrada;
```

	MesEntrada date... 2023-04-01 00:0...	TotalClientesCoo... 264 - 8436
0	2023-04-01 00:0...	264
1	2023-05-01 00:0...	8330
2	2023-06-01 00:0...	8149
3	2023-07-01 00:0...	8436
4	2023-08-01 00:0...	8260
5	2023-09-01 00:0...	7903
6	2023-10-01 00:0...	7941
7	2023-11-01 00:0...	7843
8	2023-12-01 00:0...	7985
9	2024-01-01 00:0...	7966

13 rows, 2 cols 10 / page Page 1 of 2

```sql
WITH PrimeiraCompra AS (
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
),
ComprasComCoorte AS (
    SELECT
        r.CustomerID,
        DATE_TRUNC('month', r.TransactionDate) AS MesCompra,
        p.MesEntrada,
        DATE_DIFF('month', p.MesEntrada, DATE_TRUNC('month', r.TransactionDate)) AS MesRelativo
    FROM Retail_Transaction_Clean r
    JOIN PrimeiraCompra p USING(CustomerID)
)
SELECT
    MesEntrada,
    MesRelativo,
    COUNT(DISTINCT CustomerID) AS ClientesAtivos
FROM ComprasComCoorte
GROUP BY MesEntrada, MesRelativo
ORDER BY MesEntrada, MesRelativo;
```

	MesEntrada date... 2023-04-01 00:0...	**MesRelativo** int64 0 - 11	**ClientesAtivos** in... 1 - 8436
60	2023-10-01 00:0...	0	7941
61	2023-10-01 00:0...	1	65
62	2023-10-01 00:0...	2	63
63	2023-10-01 00:0...	3	52
64	2023-10-01 00:0...	4	62
65	2023-10-01 00:0...	5	64
66	2023-10-01 00:0...	6	61
67	2023-11-01 00:0...	0	7843
68	2023-11-01 00:0...	1	60
69	2023-11-01 00:0...	2	63

```sql
WITH PrimeiraCompra AS(
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
),
ComprasComCoorte AS(
    SELECT
        r.CustomerID,
        DATE_TRUNC('month', r.TransactionDate) AS MesCompra,
        p.MesEntrada,
        DATE_DIFF('month', p.MesEntrada, DATE_TRUNC('month', r.TransactionDate)) AS MesRelativo
    FROM Retail_Transaction_Clean r
    JOIN PrimeiraCompra p USING (CustomerID)
),
AtivosPorCoorte AS (
    SELECT
        MesEntrada,
        MesRelativo,
        COUNT(DISTINCT CustomerID) AS ClientesAtivos
    FROM ComprasComCoorte
    GROUP BY MesEntrada, MesRelativo
),
TamanhoCoorte AS (
    SELECT
        MesEntrada,
        ClientesAtivos AS ClientesIniciais
    FROM AtivosPorCoorte
    WHERE MesRelativo = 0
)
SELECT
    a.MesEntrada,
    a.MesRelativo,
    a.ClientesAtivos,
    ROUND(100.0 * a.ClientesAtivos / t.ClientesIniciais, 2) AS TaxaSobrevivencia
FROM AtivosPorCoorte a
JOIN TamanhoCoorte t USING (MesEntrada)
ORDER BY a.MesEntrada, a.MesRelativo;
```

	MesEntrada date... 2023-04-01 00:0...	MesRelativo int64 0 - 11	ClientesAtivos in... 1 - 8436	TaxaSobrevivencia 0.38 - 100.0
80	2024-01-01 00:0...	2	71	0.89
81	2024-01-01 00:0...	3	52	0.65
82	2024-02-01 00:0...	0	7472	100
83	2024-02-01 00:0...	1	57	0.76
84	2024-02-01 00:0...	2	68	0.91
85	2024-03-01 00:0...	0	7731	100
86	2024-03-01 00:0...	1	71	0.92
87	2024-04-01 00:0...	0	6935	100

```sql
WITH ComprasOrdenadas AS (
    SELECT
        CustomerID,
        TransactionDate,
        LAG(TransactionDate) OVER (
            PARTITION BY CustomerID
            ORDER BY TransactionDate
        ) AS CompraAnterior
    FROM Retail_Transaction_Clean
)
SELECT
    CustomerID,
    TransactionDate AS DataReativacao,
    CompraAnterior,
    DATEDIFF('day', CompraAnterior, TransactionDate) AS DiasInatividade,
    CASE
        WHEN DATEDIFF('day', CompraAnterior, TransactionDate) >= 30 THEN 'Reativado 30+ dias'
        WHEN DATEDIFF('day', CompraAnterior, TransactionDate) >= 15 THEN 'Reativado 15+ dias'
        ELSE NULL
    END AS StatusReativacao
FROM ComprasOrdenadas
WHERE CompraAnterior IS NOT NULL
    AND DATEDIFF('day', CompraAnterior, TransactionDate) >= 15
ORDER BY CustomerID, TransactionDate;
```

	CustomerID int64 59 - 999781	DataReativacao d... 2023-05-14 16:5...	CompraAnterior d.. 2023-04-29 23:3...	DiasInatividade i... 15 - 363	StatusReativacao c Reativado... 91.3% Reativado 1... 8.7%
0	59	2024-04-01 01:0...	2023-08-19 03:5...	226	Reativado 30+ dias
1	299	2024-01-06 14:3...	2023-12-06 18:0...	31	Reativado 30+ dias
2	442	2024-03-20 08:2...	2023-08-30 03:3...	203	Reativado 30+ dias
3	959	2023-08-03 23:5...	2023-07-06 08:2...	28	Reativado 15+ dias
4	1614	2023-05-25 01:4...	2023-05-10 18:1...	15	Reativado 15+ dias
5	2128	2024-04-04 09:4...	2024-01-01 15:3...	94	Reativado 30+ dias
6	2294	2023-12-24 02:1...	2023-08-24 01:4...	122	Reativado 30+ dias
7	2521	2024-03-25 16:2...	2024-01-23 17:19...	62	Reativado 30+ dias
8	2667	2024-03-24 22:3...	2023-07-07 03:2...	261	Reativado 30+ dias
9	2899	2024-01-22 15:4...	2023-11-12 00:0...	71	Reativado 30+ dias

```sql
WITH PrimeiraCompra AS (
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
),
ComprasComCoorte AS(
    SELECT
        r.CustomerID,
        DATE_TRUNC('month', r.TransactionDate) AS MesCompra,
        p.MesEntrada,
        DATE_DIFF('month', p.MesEntrada, DATE_TRUNC('month', r.TransactionDate)) AS MesRelativo,
        r.TotalAmount
    FROM Retail_Transaction_Clean r
    JOIN PrimeiraCompra p USING (CustomerID)
)
SELECT
    MesEntrada,
    MesRelativo,
    SUM(TotalAmount) AS ReceitaMes,
    SUM(SUM(TotalAmount)) OVER (
        PARTITION BY MesEntrada
        ORDER BY MesRelativo
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS ReceitaAcumulada
FROM ComprasComCoorte
WHERE MesRelativo BETWEEN 0 AND 5
GROUP BY MesEntrada, MesRelativo
ORDER BY MesEntrada, MesRelativo;
```

	MesEntrada date... 2023-04-01 00:0...	MesRelativo int64 0 - 5	ReceitaMes float... 65.05559606 - 2...	ReceitaAcumulada 60840.78785402...
40	2023-10-01 00:0...	5	17336.24756	2043851.957
41	2023-11-01 00:0...	0	1942551.875	1942551.875
42	2023-11-01 00:0...	1	14890.65335	1957442.528
43	2023-11-01 00:0...	2	12887.7104	1970330.239
44	2023-11-01 00:0...	3	17572.09169	1987902.331
45	2023-11-01 00:0...	4	16194.14232	2004096.473
46	2023-11-01 00:0...	5	13818.21775	2017914.691
47	2023-12-01 00:0...	0	2005168.645	2005168.645
48	2023-12-01 00:0...	1	18405.1412	2023573.786
49	2023-12-01 00:0...	2	16643.56602	2040217.352

```sql
WITH PrimeiraCompra AS (
    SELECT
        CustomerID,
        MIN(DATE_TRUNC('month', TransactionDate)) AS MesEntrada
    FROM Retail_Transaction_Clean
    GROUP BY CustomerID
),
ComprasComCoorte AS (
    SELECT
        r.CustomerID,
        DATE_TRUNC('month', r.TransactionDate) AS MesCompra,
        p.MesEntrada,
        DATE_DIFF('month', p.MesEntrada, DATE_TRUNC('month', r.TransactionDate)) AS MesRelativo,
        r.TotalAmount
    FROM Retail_Transaction_Clean r
    JOIN PrimeiraCompra p USING (CustomerID)
),
Resumo AS (
    SELECT
        MesEntrada,
        MesRelativo,
        SUM(TotalAmount) AS ReceitaMes
    FROM ComprasComCoorte
    WHERE MesEntrada IN ('2023-04-01', '2023-10-01')
    GROUP BY MesEntrada, MesRelativo
)
SELECT
    MesEntrada,
    MesRelativo,
    ReceitaMes,
    SUM(ReceitaMes) OVER (
        PARTITION BY MesEntrada
        ORDER BY MesRelativo
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS ReceitaAcumulada
FROM Resumo
ORDER BY MesEntrada, MesRelativo;
```

	MesEntrada date... 2023-04-01 00:0...	MesRelativo int64 0 - 11	ReceitaMes float... 65.05559606 - 1...	ReceitaAcumulada 60840.78785402...
0	2023-04-01 00:0...	0	60840.78785	60840.78785
1	2023-04-01 00:0...	1	619.2800241	61460.06788
2	2023-04-01 00:0...	2	65.05559606	61525.12347
3	2023-04-01 00:0...	4	670.9937636	62196.11724
4	2023-04-01 00:0...	5	419.4533228	62615.57056
5	2023-04-01 00:0...	6	1810.11747	64425.68803
6	2023-04-01 00:0...	7	1019.738408	65445.42644
7	2023-04-01 00:0...	9	320.539709	65765.96615
8	2023-04-01 00:0...	10	615.2700189	66381.23617
9	2023-04-01 00:0...	11	518.213088	66899.44925

17 rows, 4 cols 10 / page Page 1 of 2

```sql
SELECT *
FROM Retail_Transaction_Clean
WHERE StoreLocation ILIKE '%ap.%'
    OR StoreLocation ILIKE '%apt%';
```

	CustomerID int64 100 - 999977	Productid object B 25.3% The 25% 2 others 49.7%	Quantity int64 1 - 9	Price float64 10.00512021 - 99...	TransactionDate d.	PaymentMethod o.	StoreLocation ob...	P
40	908160	B	5	77.911939	2023-05-02 14:1...	PayPal	31900 Stewart P...	E
41	16642	The	1	69.64528917	2024-02-29 03:5...	Cash	2694 Terrance S...	E
42	147851	C	1	64.45863786	2023-09-24 01:2...	Credit Card	9442 Gray Rapid...	C
43	714511	The	3	97.01868272	2023-09-27 19:2...	Debit Card	375 Audrey Isle A...	E
44	143779	The	2	41.2143105	2024-04-07 17:3...	Cash	8659 Clay Street ...	B
45	718553	B	7	44.91639388	2024-03-25 03:4...	Debit Card	4023 Jennifer Pr...	B
46	99487	D	8	51.16619973	2024-02-17 21:4...	Debit Card	7089 Marshall Cli...	B
47	706934	The	3	80.08662747	2024-04-01 03:0...	Debit Card	982 Sharp Cours...	B
48	139063	The	7	52.35890696	2023-12-15 09:12...	PayPal	125 Decker Lake ...	C
49	92407	D	1	92.56164203	2024-04-15 10:5...	Cash	6421 Suzanne Su...	H

22,118 rows, 10 cols 10 / page Page 5 of 2,212

```sql
SELECT
    ROUND(100.0 * COUNT(*) FILTER (
        WHERE StoreLocation ILIKE '%ap.%'
            OR StoreLocation ILIKE '%apt%'
    ) / COUNT(*), 2) AS PercentualApts
FROM Retail_Transaction_Clean;
```

	PercentualApts fl...
0	22.12

1 row, 1 col 10 / page Page 1 of 1

```sql
SELECT
    StoreLocation,
    REGEXP_EXTRACT(StoreLocation, '[ ,]([A-Z]{2})[ ]?[0-9]{5}', 1) AS EstadoAbrev
FROM Retail_Transaction_Clean
LIMIT 20;
```

	StoreLocation ob... 176 Andrew... 5% 11635 Willia... 5% 18 others 90%	EstadoAbrev obj... MO 15% MT 10% 13 others 75%
0	176 Andrew Cliffs...	HI
1	11635 William Wel...	MT
2	910 Mendez Ville ...	MO
3	87522 Sharon Co...	MO
4	0070 Michelle Isl...	VA
5	8492 Jonathan D...	TN
6	USNV Harrell FP...	AA
7	489 Juan Loop A...	WV
8	528 Justin Expre...	SD
9	189 Wright Mews...	MO

20 rows, 2 cols 10 / page Page 1 of 2

```sql
WITH Estados AS (
    SELECT
        StoreLocation,
        REGEXP_EXTRACT(StoreLocation, '[ ,]([A-Z]{2})[ ]?[0-9]{5}', 1) AS EstadoAbrev
    FROM Retail_Transaction_Clean
)
SELECT
    StoreLocation,
    EstadoAbrev,
    CASE
        WHEN EstadoAbrev IN ('PR','GU','VI','AS','MP') THEN 'Insular'
        WHEN EstadoAbrev IN ('FM','MH','PW') THEN 'Associated'
        WHEN EstadoAbrev IN ('AE','AP','AA') THEN 'Military'
        ELSE 'Continental'
    END AS TipoRegiao
FROM Estados
LIMIT 20;
```

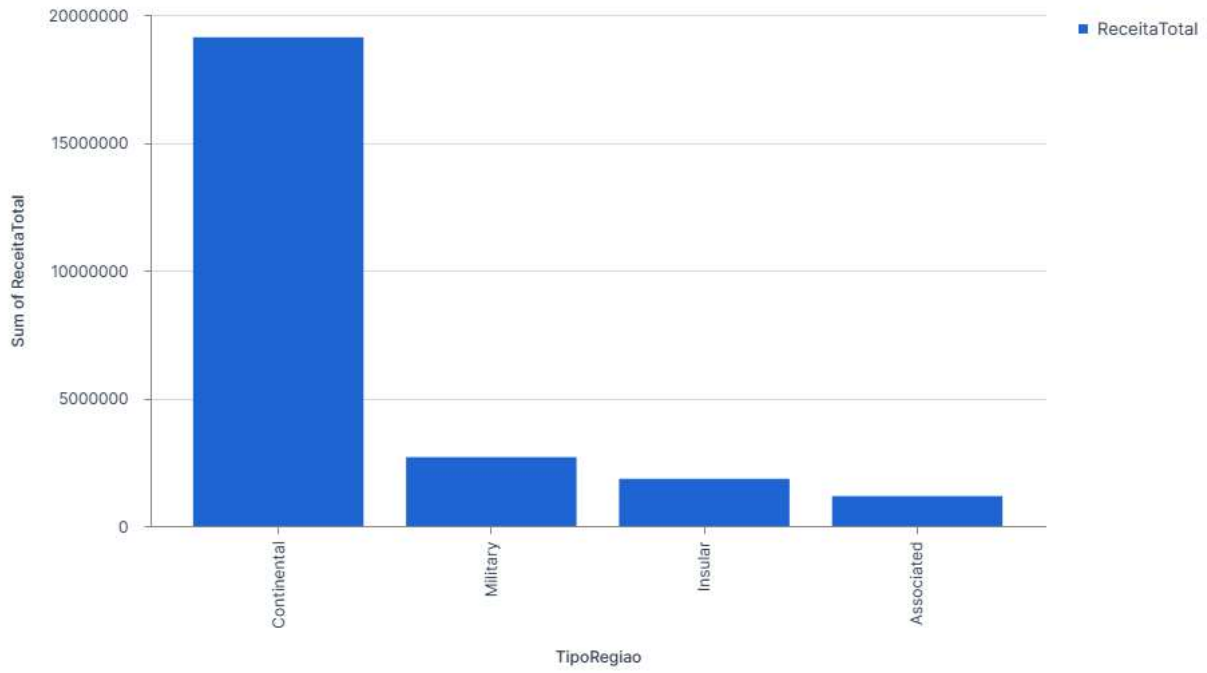
	StoreLocation ob...	EstadoAbrev obj...	TipoRegiao object
	176 Andrew... 5% 11635 Willia... 5% 18 others 90%	MO 15% MT 10% 13 others 75%	Continental 80% Military 10% Associated 10%
0	176 Andrew Cliffs...	HI	Continental
1	11635 William Wel...	MT	Continental
2	910 Mendez Ville ...	MO	Continental
3	87522 Sharon Co...	MO	Continental
4	0070 Michelle Isl...	VA	Continental
5	8492 Jonathan D...	TN	Continental
6	USNV Harrell FP...	AA	Military
7	489 Juan Loop A...	WV	Continental
8	528 Justin Expre...	SD	Continental
9	189 Wright Mews...	MO	Continental

20 rows, 3 cols 10 / page Page 1 of 2

```sql
WITH Estados AS (
    SELECT
        StoreLocation,
        REGEXP_EXTRACT(StoreLocation, '[ ,]([A-Z]{2})[ ]?[0-9]{5}', 1) AS EstadoAbrev
    FROM Retail_Transaction_Clean
),
Classificados AS (
  SELECT
    StoreLocation,
    EstadoAbrev,
    CASE
      WHEN EstadoAbrev IN ('PR','GU','VI','AS','MP') THEN 'Insular'
      WHEN EstadoAbrev IN ('FM','MH','PW')           THEN 'Associated'
      WHEN EstadoAbrev IN ('AE','AP','AA')           THEN 'Military'
      WHEN EstadoAbrev IS NULL                       THEN 'Indefinido'
      ELSE 'Continental'
    END AS TipoRegiao,
    TotalAmount
  FROM Estados e
  JOIN Retail_Transaction_Clean r USING (StoreLocation)
)
SELECT
  TipoRegiao,
  COUNT(*) AS TotalPedidos,
  SUM(TotalAmount) AS ReceitaTotal,
  ROUND(AVG(TotalAmount),2) AS TicketMedio
FROM Classificados
GROUP BY TipoRegiao
ORDER BY ReceitaTotal DESC;
```

	TipoRegiao object	TotalPedidos int64	ReceitaTotal floa...	TicketMedio floa...
0	Continental	77004	19121633.35	248.32
1	Military	10806	2693718.504	249.28
2	Insular	7528	1847357.042	245.4
3	Associated	4662	1170786.606	251.13

4 rows, 4 cols 10 / page Page 1 of 1




```sql
SELECT
    CONCAT(
        'Cliente ', CustomerID,
        ' comprou ', Quantity, ' unidades do produto ', ProductID,
        ' em ', DATE_TRUNC('day', TransactionDate)
    ) AS FrasePedido
FROM Retail_Transaction_Clean
LIMIT 10;
```

	FrasePedido obje... Cliente 109... 10% Cliente 993... 10% 8 others 80%
0	Cliente 109318 c...
1	Cliente 993229 c...
2	Cliente 579675 c...
3	Cliente 799826 c...
4	Cliente 121413 co...
5	Cliente 463050 c...
6	Cliente 888163 c...
7	Cliente 843385 c...
8	Cliente 839609 c...
9	Cliente 184135 c...

10 rows, 1 col 10 / page Page 1 of 1