

Pontifícia Universidade Católica do Rio de Janeiro

MVP ENGENHARIA DE DADOS

Rio de Janeiro

2024

Felipe de Oliveira Pereira
Documento referente ao MVP da disciplina
Engenharia de Dados
Esp. Ciência de Dados e Analytics – PUC RIO

Sumário

| | | |
|----------|---|-----------|
| 1 | Informações sobre o dataset utilizado..... | 4 |
| 1.1 | O dataset | 4 |
| 1.2 | Catálogo do dataset..... | 4 |
| 2 | Questões de negócio..... | 5 |
| 3 | Configuração Clauster..... | 5 |
| 4 | Import..... | 6 |
| 5 | Tratamentos | 7 |
| 6 | Análise dos dados | 8 |
| 6.1 | Cliente que mais gastou na loja..... | 8 |
| 6.2 | Produto mais vendido | 9 |
| 6.3 | Maior compra..... | 10 |
| 6.4 | Cidade com maior valor vendido | 11 |
| 6.5 | Tempo médio de envio | 12 |
| 7 | Autoavaliação..... | 12 |

1. Informações sobre o dataset utilizado

1.1 O dataset

Conjunto de dados do Kaggle referente à 4 anos de vendas de uma superloja do setor de varejo. É uma tabela flat que contém todas as informações de cada pedido realizado no período guardado.

Pode ser acessado através do link '<https://www.kaggle.com/datasets/rohitsahoo/sales-forecasting/data>'.

Além disso, este conjunto de dados conta com uma licença GPL 2. Leia mais sobre este tipo de permissão de uso através do link '<http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>'.

1.2 Catálogo do dataset

Obs: O catálogo abaixo é referente à tabela já tratada disponibilizada ao usuário final para execução das análises, conforme explanado ao longo do documento.

Default.vw_vendas_final Catalog

| Coluna | Tipo | Descrição |
|---------------|---------|--|
| id_pedido | String | Identificador do pedido. Não ha valores repetidos, cada pedido gera um ID específico |
| dt_pedido | Date | Data em que o cliente finalizou a compra do carrinho |
| dt_envio | Date | Date em que o pedido do cliente foi entregue à transportadora |
| id_cliente | String | Identificador do cliente. Não ha valores repetidos, cada CPF gera um ID específico |
| nome_cliente | String | Nome cadastrado para cada cliente da base, pode haver valores iguais |
| cidade | String | Cidade de residencia do cliente, preenchido no ato do cadastro |
| id_produto | String | Identificador do produto. Não ha valores repetidos, cada novo produto cadastrado na base gera um ID específico |
| nome_produto | String | Nome do modelo de cada produto cadastrado para venda |
| valor_produto | Numeric | Preço cheio de cada produto cadastrado vendido |

2. Questões de negócio a serem respondidas após tratamento do dataset

- Qual cliente que mais gastou na loja?
- Qual produto mais vendido
- Qual o pedido de maior valor?
- Qual a cidade com maior valor vendido no período?
- Qual o tempo médio de envio dos produtos?

3. Configuração Cluster

Criado cluster conforme configuração do Databricks Community abaixo.

[Compute](#) > [New compute](#) > [Preview](#)

Felipe Pereira's Cluster

Compute name

Felipe Pereira's Cluster

Databricks runtime version ⓘ

Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)

Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options [↗](#), please upgrade your Databricks subscription. [↗](#)

Instances Spark

Spark config ⓘ

```
spark.databricks.rocksDB.fileManager.useCommitService false
```

Environment variables ⓘ

```
MY_VAR=hello
MY_OTHER_VAR=$MY_VAR world
MY_SECRET_DB_PASSWORD={{secrets/prod/database_password}}
```

4. Import

A base foi importada crua para o databricks e todos os campos 'setados' como string para evitar erros na ingestão de dados.

Schema:

| | Δ_C^B col_name | Δ_C^B data_type | Δ_C^B comment |
|----|-----------------------|------------------------|----------------------|
| 1 | Row ID | string | null |
| 2 | Order ID | string | null |
| 3 | Order Date | string | null |
| 4 | Ship Date | string | null |
| 5 | Ship Mode | string | null |
| 6 | Customer ID | string | null |
| 7 | Customer Name | string | null |
| 8 | Segment | string | null |
| 9 | Country | string | null |
| 10 | City | string | null |
| 11 | State | string | null |
| 12 | Postal Code | string | null |
| 13 | Region | string | null |
| 14 | Product ID | string | null |
| 15 | Category | string | null |

5. Tratamentos

Como a base já é praticamente limpa, foi necessário apenas fazer a seleção de atributos e formatação dos campos de data. O dado tratado foi disponibilizado numa `vw` para o usuário final.



```
CREATE TABLE default.vw_vendas_final as (SELECT `Order ID` as id_pedido,
TO_DATE(CAST(UNIX_TIMESTAMP(`Order Date`, 'dd/MM/yyyy') AS TIMESTAMP)) as dt_pedido,
TO_DATE(CAST(UNIX_TIMESTAMP(`Ship Date`, 'dd/MM/yyyy') AS TIMESTAMP)) as dt_envio,
`Customer ID` as id_cliente,
`Customer Name` as nome_cliente,
City as cidade,
`Product ID` as id_produto,
`Product Name` as nome_produto,
Sales as valor_produto
FROM default.base_vendas)
```

▼ (6) Spark Jobs

- ▶ Job 31 [View](#) (Stages: 1/1)
- ▶ Job 32 [View](#) (Stages: 1/1)
- ▶ Job 34 [View](#) (Stages: 1/1)
- ▶ Job 35 [View](#) (Stages: 1/1, 1 skipped)
- ▶ Job 36 [View](#) (Stages: 1/1, 1 skipped)
- ▶ Job 37 [View](#) (Stages: 1/1, 2 skipped)

Query returned no results

6. Análise dos dados

A partir da `vw_vendas_final` iremos responder as perguntas anteriormente listadas. A `vw` atende suficientemente bem para que cheguemos ao nosso objetivo. Não é necessário mais nenhum tratamento adicional, além dos que foram feitos na construção da `vw`.

6.1 - Cliente que mais gastou na loja: Foi o Sean Miller com \$25.043,05 dólares gastos.

Just now (2s)

```
SELECT id_cliente,
       nome_cliente,
       sum(valor_produto)
FROM default.vw_vendas_final
GROUP BY 1,2
ORDER BY 3 DESC
```

(2) Spark Jobs

- Job 51 [View](#) (Stages: 1/1)
- Job 52 [View](#) (Stages: 1/1, 1 skipped)

| | <code>id_cliente</code> | <code>nome_cliente</code> | <code>1.2 sum(valor_produto)</code> |
|----|-------------------------|---------------------------|-------------------------------------|
| 1 | SM-20320 | Sean Miller | 25043.05 |
| 2 | TC-20980 | Tamara Chand | 19052.217999999997 |
| 3 | RB-19360 | Raymond Buch | 15117.339 |
| 4 | TA-21385 | Tom Ashbrook | 14595.62 |
| 5 | AB-10105 | Adrian Barton | 14473.570999999998 |
| 6 | KL-16645 | Ken Lonsdale | 14175.229 |
| 7 | SC-20095 | Sanjit Chand | 14142.333999999999 |
| 8 | HL-15040 | Hunter Lopez | 12873.297999999999 |
| 9 | SE-20110 | Sanjit Engle | 12209.438000000002 |
| 10 | CC-12370 | Christopher Conant | 12129.072 |
| 11 | TS-21370 | Todd Sumrall | 11891.751 |
| 12 | GT-14710 | Greg Tran | 11820.119999999997 |
| 13 | BM-11140 | Becky Martin | 11789.630000000003 |
| 14 | SV-20365 | Seth Vernon | 11470.949999999997 |
| 15 | CJ-12010 | Caroline Jumper | 11164.974 |

793 rows | 2.10 seconds runtime

6.2 - Produto mais vendido: Foi a Canon imageCLASS 2200 Advanced Copier com \$61.599,82 dólares vendidos.

Just now (2s)

7

```
SELECT id_produto,
       nome_produto,
       sum(valor_produto)
FROM default.vw_vendas_final
GROUP BY 1,2
ORDER BY 3 DESC
```

(2) Spark Jobs

Table

| | A _C id_pro... | A _C nome_produto | 1.2 sum(valor_produto) |
|----|--------------------------|---|------------------------|
| 1 | TEC-CO-10004722 | Canon imageCLASS 2200 Advanced Copier | 61599.824 |
| 2 | OFF-BI-10003527 | Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind | 27453.384 |
| 3 | TEC-MA-10002412 | Cisco TelePresence System EX90 Videoconferencing Unit | 22638.48 |
| 4 | FUR-CH-10002024 | HON 5400 Series Task Chairs for Big and Tall | 21870.576 |
| 5 | OFF-BI-10001359 | GBC DocuBind TL300 Electric Binding System | 19823.479000000003 |
| 6 | OFF-BI-10000545 | GBC Ibimaster 500 Manual ProClick Binding System | 19024.5 |
| 7 | TEC-CO-10001449 | Hewlett Packard LaserJet 3310 Copier | 18839.686 |
| 8 | TEC-MA-10001127 | HP Designjet T520 Inkjet Large Format Printer - 24" Color | 18374.895 |
| 9 | OFF-BI-10004995 | GBC DocuBind P400 Electric Binding System | 17965.068 |
| 10 | OFF-SU-10000151 | High Speed Automatic Electric Letter Opener | 17030.311999999998 |
| 11 | TEC-MA-10000822 | Lexmark MX611dhe Monochrome Laser Printer | 16829.901 |
| 12 | OFF-SU-10002881 | Martin Yale Chadless Opener Electric Letter Opener | 16656.199999999997 |
| 13 | OFF-BI-10001120 | Ibico EPK-21 Electric Binding System | 15875.916000000001 |
| 14 | FUR-BO-10004834 | Riverside Palais Royal Lawyers Bookcase, Royale Cherry Finish | 15610.9656 |
| 15 | TEC-MA-10001047 | 3D Systems Cube Printer, 2nd Generation, Magenta | 14299.89 |

6.3 - Maior compra: A maior compra foi feita por Sean Miller no valor de \$23.661,22 dólares.

Just now (1s)

```
SELECT id_pedido,
       nome_cliente,
       sum(valor_produto)
FROM default.vw_vendas_final
GROUP BY 1,2
ORDER BY 3 DESC
```

(2) Spark Jobs

Table +

| | A ^B _C id_pedido | A ^B _C nome_cliente | 1.2 sum(valor_produto) |
|----|---------------------------------------|--|------------------------|
| 1 | CA-2015-145317 | Sean Miller | 23661.228 |
| 2 | CA-2017-118689 | Tamara Chand | 18336.739999999998 |
| 3 | CA-2018-140151 | Raymond Buch | 14052.48 |
| 4 | CA-2018-127180 | Tom Ashbrook | 13716.458 |
| 5 | CA-2015-139892 | Becky Martin | 10539.896 |
| 6 | CA-2018-166709 | Hunter Lopez | 10499.97 |
| 7 | CA-2015-116904 | Sanjit Chand | 9900.19 |
| 8 | CA-2017-117121 | Adrian Barton | 9892.74 |
| 9 | US-2017-107440 | Bill Shonely | 9135.189999999999 |
| 10 | CA-2017-158841 | Sanjit Engle | 8805.04 |
| 11 | CA-2017-143714 | Christopher Conant | 8539.02 |
| 12 | CA-2015-143917 | Ken Lonsdale | 8319.289999999999 |
| 13 | US-2018-168116 | Grant Thornton | 8167.419999999999 |
| 14 | US-2016-126977 | Peter Fuller | 7678.228 |
| 15 | CA-2018-100111 | Seth Vernon | 7359.918000000001 |

4,922 rows | 1.49 seconds runtime

6.4 - Cidade com maior valor vendido: A cidade com mais vendas é New York com \$252.462,54 dólares vendidos.

Just now (1s)

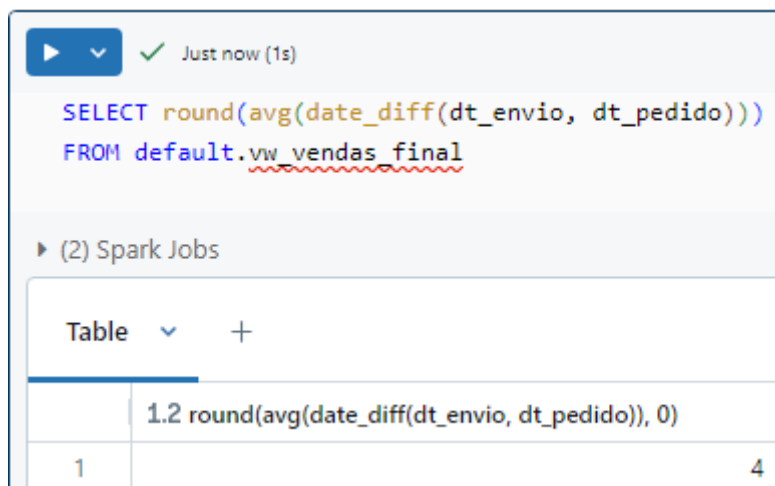
```
SELECT cidade,
| | | sum(valor_produto)
FROM default.vw_vendas_final
GROUP BY 1
ORDER BY 2 DESC
```

(2) Spark Jobs

Table ▾ +

| | ^A _C cidade | 1.2 sum(valor_produto) |
|----|----------------------------------|------------------------|
| 1 | New York City | 252462.547 |
| 2 | Los Angeles | 173420.181 |
| 3 | Seattle | 116106.322 |
| 4 | San Francisco | 109041.11999999999 |
| 5 | Philadelphia | 108841.74900000007 |
| 6 | Houston | 63956.142799999936 |
| 7 | Chicago | 47820.133000000002 |
| 8 | San Diego | 47521.028999999995 |
| 9 | Jacksonville | 44713.183 |
| 10 | Detroit | 42446.944000000002 |
| 11 | Springfield | 41827.81 |
| 12 | Columbus | 38662.562999999995 |
| 13 | Newark | 28448.048999999995 |
| 14 | Columbia | 25283.324 |

6.5 - Tempo médio de envio: O tempo médio de envio é de aproximadamente 4 dias.



The screenshot shows a SQL query execution interface. At the top, there is a play button, a dropdown arrow, a green checkmark, and the text "Just now (1s)". Below this is the SQL query: `SELECT round(avg(date_diff(dt_envio, dt_pedido)))` and `FROM default.vw_vendas_final`. The table name `vw_vendas_final` is underlined with a red wavy line. Below the query, it says "(2) Spark Jobs". Underneath is a table with a header row and one data row. The header row has a column labeled "Table" with a dropdown arrow and a plus sign. The data row has a column with the value "1.2 round(avg(date_diff(dt_envio, dt_pedido)), 0)" and another column with the value "4".

| Table | |
|-------|---|
| 1 | 4 |

7. Autoavaliação

Todos os objetivos delineados no início do trabalho foram atingidos e estou feliz com o resultado, por entender que atendi todas as expectativas. Muito pela preocupação de garantir a entrega do MVP a tempo, usei um dataset e metodologia de ingestão de dados muito simples, por isso como desafio de carreira vou aprofundar um pouco mais a integração com AWS, ferramenta que sempre tive vontade de entender melhor. Além disso, pretendo adentrar mais também em metodologias de Jobs de ingestão e tratamento agendados, para atualizar automaticamente os datasets e vws, essa que imagino ser a forma mais adequada pensando em escalabilidade. Por fim, agradeço a todo o corpo docente responsável pelo curso.