

# ETL – PROJETO FINAL

Especialização de Analytics





# Índice

1	Introdução .....	3
2	Arquitetura .....	4
2.1	Design escolhido .....	4
2.1.1	Data Source .....	4
2.1.2	Data Ingestion.....	4
2.1.3	Data Engineering.....	5
2.1.4	Data Visualization .....	9
2.1.5	Componentes da seleção e decisão da Arquitetura.....	12
2.2	Data Flow Design .....	12
	Figura 1 - Modelo inicial .....	3
	Figura 2 - Arquitetura do projeto de Analytics.....	4
	Figura 3 - Importação das bibliotecas no Databricks .....	5
	Figura 4 - Modelo Dimensional BikeCapStore .....	5
	Figura 5 - Dim_Staffs final .....	6
	Figura 6 - Dim_Customers final .....	6
	Figura 7 - Dim_Products final.....	6
	Figura 8 - Dim_Stores final .....	7
	Figura 9 - Criação da Dimensão Data .....	7
	Figura 10 - Dim_Date final .....	8
	Figura 11 - Função da transformação das datas para chaves .....	8
	Figura 12 - Fact_Orders final.....	8
	Figura 13 - Limpeza do DBFS e exportação dos dataframes para tabelas .....	9
	Figura 14 - Ligação direta do Databricks para o PowerBI .....	10
	Figura 15 - Pedido de especificações do cluster pelo PowerBI .....	10
	Figura 16 - Especificações do cluster .....	11
	Figura 17 - Modelo Dimensional importado no PowerBI.....	11
	Figura 18 - Data Flow do projeto .....	12
	Tabela 1 – Inputs, caminhos e endpoints do ETL.....	13

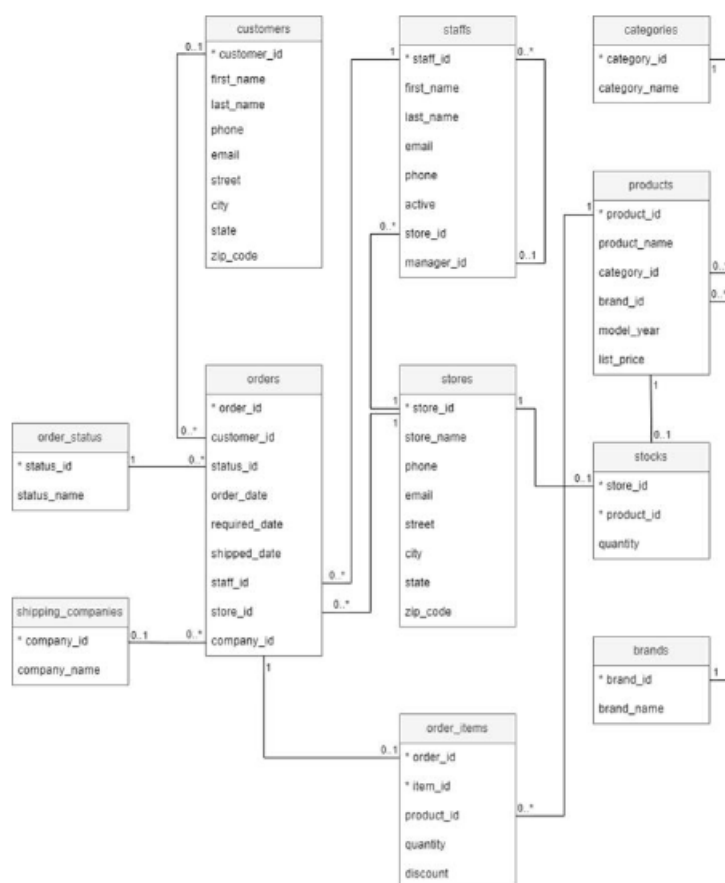


# 1 Introdução

Este relatório pretende fornecer toda a informação técnica utilizada durante o desenvolvimento do projeto de ETL para a BikeCapStore, de forma a possibilitar a sua reprodução por outros membros da equipa.

Para a elaboração do projeto final da Especialização de Analytics, os dados fornecidos foram trabalhados no Dbeaver, e posteriormente exportados para o repositório do GitHub. De seguida foi feita a ingestão dos ficheiros para o Databricks, onde foi construído o modelo dimensional, através de várias operações de transformação de dados. No final desta tarefa, as novas tabelas foram exportadas para o Power BI.

Apresenta-se o input original na Figura 1.





## 2 Arquitetura

### 2.1 Design escolhido

A Figura 2 representa a arquitetura do projeto de Analytics.

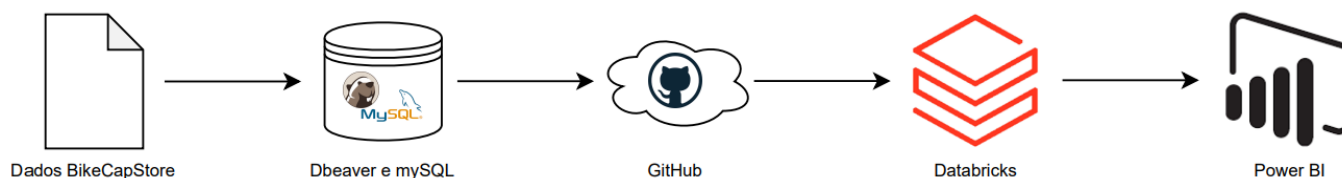


Figura 2 - Arquitetura do projeto de Analytics

#### 2.1.1 Data Source

O GitHub é responsável por manter os ficheiros em Cloud para diferentes equipas terem acesso aos dados.

Neste projeto foram utilizados dados de uma única fonte, o repositório GitHub, onde estão ficheiros do tipo csv, na seguinte localização: <https://github.com/XLucas27X/bikecapstore>

#### 2.1.2 Data Ingestion

O Databricks foi utilizada como ferramenta de ETL para ingerir os ficheiros do GitHub, através do comando "wget" do Power Shell.

Para a ingestão é utilizada o raw link de cada ficheiro.

%sh

```
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/brands.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/categories.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/customers.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/order_items.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/orders.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/products.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/staffs.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/stocks.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/stores.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/shipping_companies.csv
wget https://raw.githubusercontent.com/XLucas27X/bikecapstore/main/order_status.csv
```

O notebook pode ser consultado no seguinte link (é necessário manter sempre o cluster ativo):

<https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bfcf/3904951712713107/315028503103403/245661924386905/latest.html>



### 2.1.3 Data Engineering

O processo inicia-se com a importação das bibliotecas necessárias de modo a facilitar a transformação dos dados. Neste projeto utilizaram-se as linguagens Python, SQL e Linux.

```
3 from pyspark.sql.functions import *
4 from pyspark.sql import functions as F
5 from pyspark.sql.types import *
6 import pandas as pd
```

Figura 3 - Importação das bibliotecas no Databricks

Foi desenvolvida uma função que automatiza a criação de dataframes a partir dos csv importados. Nos parâmetros utilizados assume-se a primeira linha como "TRUE", inferimos o schema de cada ficheiro e utilizou-se a vírgula como separador. O formato de leitura é csv e os ficheiros foram carregados a partir do filesystem do databricks: <file:/databricks/driver/>

Após a criação dos dataframes, procedemos ao design do modelo:

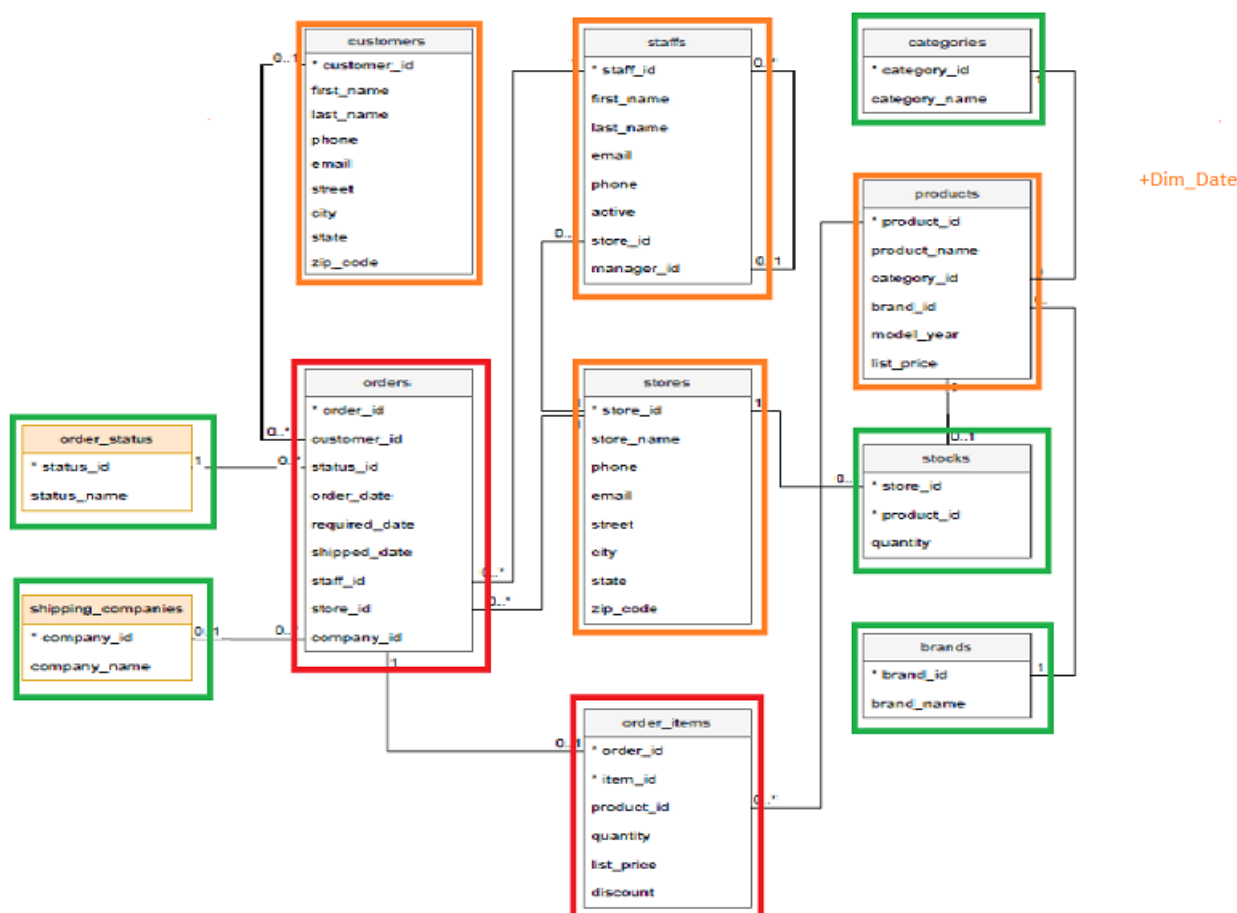


Figura 4 - Modelo Dimensional BikeCapStore



As tabelas foram primeiramente classificadas consoante o grau de granularidade, e foram criadas as diferentes dimensões do modelo e a facts table, de modo a formar o star schema.

**Vermelho** → Facts Table

**Laranja** → Dimensões

**Verde** → Tabelas que vão ser integradas conforme abaixo

**Dim\_Staffs:** Eliminação da coluna com o id desnecessário (store\_id) para o modelo final.

	staff_id ▲	first_name ▲	last_name ▲	email ▲	phone ▲	active ▲	manager_id ▲
1	1	Fabiola	Jackson	fabiola.jackson@bikes.shop	(831) 555-5554	1	NULL
2	2	Mireya	Copeland	mireya.copeland@bikes.shop	(831) 555-5555	1	1
3	3	Genna	Serrano	genna.serrano@bikes.shop	(831) 555-5556	1	2
4	4	Virgie	Wiggins	virgie.wiggins@bikes.shop	(831) 555-5557	1	2

Figura 5 - Dim\_Staffs final

**Dim\_Customers:** Foi mantida a tabela original.

	customer_id ▲	first_name ▲	last_name ▲	phone ▲	email ▲	street ▲	city ▲	state ▲	zip_code ▲
1	1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127
2	2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553

Figura 6 - Dim\_Customers final

**Dim\_Products:**

- Join das categories e brands com os products para criar a dimensao products
- Eliminação dos id's repetidos e desnecessários (brands\_id e category\_id), resultantes do join das tabelas.

	product_id ▲	product_name ▲	brand_id ▲	category_id ▲	model_year ▲	list_price ▲
1	1	Trek 820 - 2016	9	6	2016	379.99
2	2	Ritchey Timberwolf Frameset - 2016	5	6	2016	749.99
3	3	Surly Wednesday Frameset - 2016	8	6	2016	999.99
4	4	Trek Fuel EX 8 29 - 2016	9	6	2016	2899.99

Figura 7 - Dim\_Products final



**Dim\_Stores:** Foi mantida a tabela stores original, descartando a tabela stocks, uma vez que a mesma não continha informação relevante para responder aos KPI's solicitados.

	store_id ▲	store_name ▲	phone ▲	email ▲	street ▲	city ▲	state ▲	zip_code ▲
1	1	Santa Cruz Bikes	(831) 476-4321	santacruz@bikes.shop	3700 Portola Drive	Santa Cruz	CA	95060
2	2	Baldwin Bikes	(516) 379-8888	baldwin@bikes.shop	4200 Chestnut Lane	Baldwin	NY	11432
3	3	Rowlett Bikes	(972) 530-5555	rowlett@bikes.shop	8000 Fairway Avenue	Rowlett	TX	75088

Figura 8 - Dim\_Stores final

## Dim\_Date:

- Criação manual da dimensão data, recorrendo a uma view desenvolvida em Spark SQL, com as seguintes colunas:
  - a) calendarDate (data completa)
  - b) Day\_Name (nome do dia da semana)
  - c) DayOfMonth (número do dia do mês)
  - d) DayOfWeek (número do dia da semana)
  - e) DayOfYear (número do dia do ano)
  - f) IsWeekDay (se é dia útil ou não)
  - g) Month (número do mês)
  - h) Month\_Name (nome do mês)
  - i) QuarterOfYear (número do trimestre)
  - j) Year (ano)
  - k) SK\_date (chave primária)

```
2 beginDate = '2016-01-01'
3 endDate = '2024-12-31'
4 #dd-MM-yyyy
5
6 spark.sql(f"select explode(sequence(to_date('{beginDate}'), to_date('{endDate}'), interval 1 day)) as calendarDate").createOrReplaceTempView('dates')
```

Command took 0.41 seconds -- by lucas.reis.neves@hotmail.com at 12/29/2022, 12:36:24 PM on My Cluster

Cmd 14

```
1 %sql
2 create temporary view Dim_Date
3 as
4 select
5     day(calendarDate)+ month(calendarDate) * 100+year(calendarDate) * 10000 as SK_date,
6     calendarDate,
7     year(calendarDate) AS Year,
8     date_format(calendarDate, 'MMMM') as Month_Name,
9     month(calendarDate) as Month,
10    date_format(calendarDate, 'EEEE') as Day_Name,
11    weekday(calendarDate) + 1 as DayOfWeek,
12    case
13        when weekday(calendarDate) < 5 then 'Y'
14        else 'N'
15    end as IsWeekDay,
16    dayofmonth(calendarDate) as DayOfMonth,
17    dayofyear(calendarDate) as DayOfYear,
18    quarter(calendarDate) as QuarterOfYear
19
20 from
21     dates
22 order by
23     calendarDate
```

Figura 9 - Criação da Dimensão Data



	SK_date ▲	calendarDate ▲	Year ▲	Month_Name ▲	Month ▲	Day_Name ▲	DayOfWeek ▲	IsWeekDay ▲	DayOfMonth ▲	DayOfYear ▲	QuarterOfYear ▲
1	01012016	01-01-2016	2016	January	1	Friday	5	Y	1	1	1
2	02012016	02-01-2016	2016	January	1	Saturday	6	N	2	2	1
3	03012016	03-01-2016	2016	January	1	Sunday	7	N	3	3	1
4	04012016	04-01-2016	2016	January	1	Monday	1	Y	4	4	1

Figura 10 - Dim\_Date final

## Fact\_Orders:

- Join das tabelas orders, orders\_status e shipping\_companies para criar a facts table;
- Eliminação dos id's repetidos e desnecessários (status\_id e company\_id), que resultaram do join das tabelas;
- Aplicação de uma função que foi desenvolvida para alterar o formato default do Databricks para o do PBI (yyyyMMdd para ddMMyyyy);
- Transformação das datas em chaves (da facts table), de forma a ligar com a dimensão data através da SK\_date (i.e., foram retirados os "-" e zeros (00:00:00) das datas).

```

1  #Reversing datekeys to the desired format (ddMMyyyy)
2  def reverse_datekey(spark_dataframe, datekey,colno,dash=False):
3      sklist=[]
4      if dash==True:
5          for index, i in spark_dataframe.toPandas().iterrows():
6              aux=str(i[1])
7              sklist.append(aux[8:]+ "-" +aux[5:-3]+ "-" +aux[:4])
8      else:
9          for index, i in spark_dataframe.toPandas().iterrows():
10             aux=str(i[colno])
11             sklist.append(aux[6:]+aux[4:-2]+aux[:4])
12
13
14     new_pandas_df = spark_dataframe.toPandas()
15     new_pandas_df[datekey] = sklist
16
17     return spark.createDataFrame(new_pandas_df)

```

Figura 11 - Função da transformação das datas para chaves

	order_id ▲	customer_id ▲	status_id ▲	order_date ▲	required_date ▲	shipped_date ▲	store_id ▲	staff_id ▲	company_id ▲
1	1	259	4	2016-01-01T00:00:00.000+0000	2016-01-03T00:00:00.000+0000	2016-01-03	1	2	4
2	2	1212	4	2016-01-01T00:00:00.000+0000	2016-01-04T00:00:00.000+0000	2016-01-03	2	6	4
3	3	523	4	2016-01-02T00:00:00.000+0000	2016-01-05T00:00:00.000+0000	2016-01-03	2	7	4
4	4	175	4	2016-01-03T00:00:00.000+0000	2016-01-04T00:00:00.000+0000	2016-01-05	1	3	5

Figura 12 - Fact\_Orders final





Por fim, efetuou-se a limpeza do DBFS e a exportação dos dataframes para tabelas no Databricks.

```
Cmd 24
1 %fs rm -r dbfs:/user

res0: Boolean = true

Command took 15.95 seconds -- by lucas.reis.neves@hotmail.com at 1:

Cmd 25
1 #EXPORTS to filesystem
2 products_df.write.saveAsTable("dim_products")
3 orders_df.write.saveAsTable("fact_orders")
4 dimDate_df.write.saveAsTable("dim_date")
5 customers.write.saveAsTable("dim_customers")
6 staffs.write.saveAsTable("dim_staffs")
7 stores.write.saveAsTable("dim_stores")

▶ (24) Spark Jobs

💡 1
```

Figura 13 - Limpeza do DBFS e exportação dos dataframes para tabelas

No que concerne ao processamento dos dados, foi utilizado o motor Spark através de um cluster do Databricks. O cluster tem um runtime 10.4, active memory 15GB e active cores 2.

#### Spark config:

spark.databricks.rocksDB.FileManager.useCommitService - false

spark.databricks.delta.preview.enabled - true

#### Environment variables:

PYSPARK\_PYTHON=/databricks/python3/bin/python3

### 2.1.4 Data Visualization

O PowerBI é o último componente da arquitetura, utilizado como ferramenta para a visualização. O PowerBI permite uma ligação directa com o Databricks.

Os dataframes criados encontram-se no Databricks filesystem, no seguinte diretório:

[dbfs:/user/hive/warehouse](#)



Para processar esta ligação é necessário abrir o PBI, fazer a importação dos dados pelo Get Data, escolher o Azure Databricks e conectar.

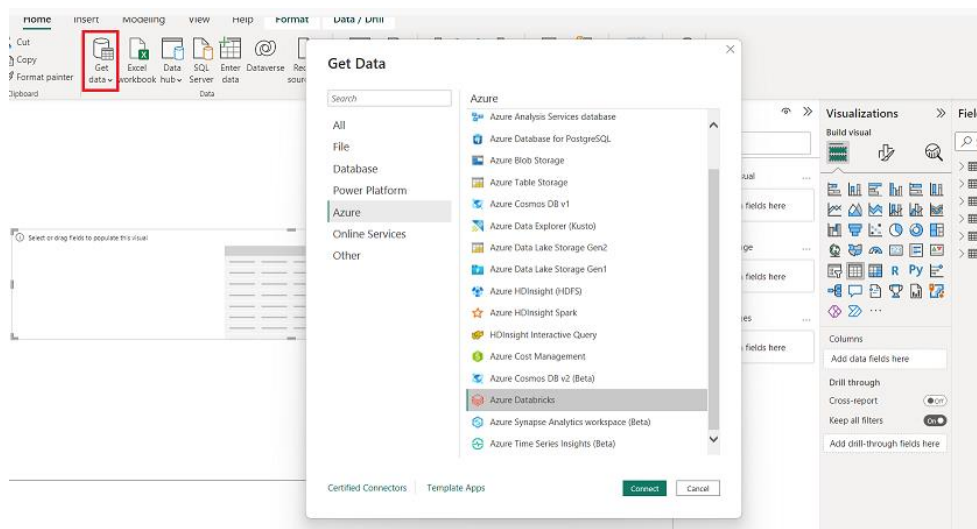


Figura 14 - Ligação direta do Databricks para o PowerBI

De seguida colocar as especificações solicitadas -Server Hostname e e HTTP- (No Databricks, Ir a Compute, entrar no cluster utilizado e escolher a opção JDBC/ODBC). Posteriormente, utilizar as credenciais de acesso ou token de autenticação do Databricks. Selecionar as tabelas a carregar para o modelo.

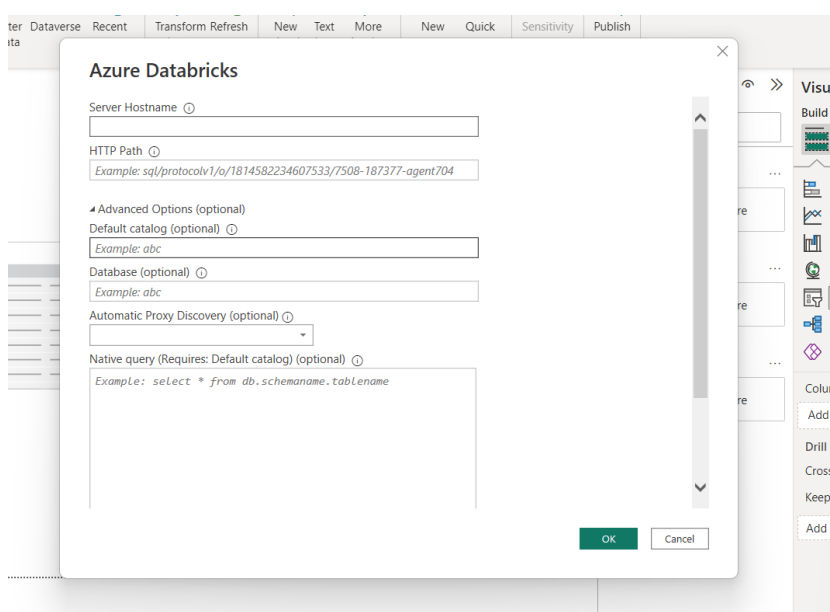


Figura 15 - Pedido de especificações do cluster pelo PowerBI



Clusters / My Cluster

### My Cluster ✓

Configuration   Notebooks (0)   Libraries   Event log   Spark UI   Driver logs   Metrics   Apps   Spark cluster UI -

10.4 LTS (includes Apache Spark 3.2.1, Scala 2.12)

Driver type  
Community Optimized   15.3 GB Memory, 2 Cores, 1 DBU

Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Instances   Spark   **JDBC/ODBC**   Permissions

Server Hostname  
**community.cloud.databricks.com**

Port  
443

Protocol  
HTTPS

HTTP Path  
**sql/protocolv1/o/3904951712713107/0102-105935-e13px82e**

Figura 16 - Especificações do cluster

Da importação resulta o seguinte modelo:

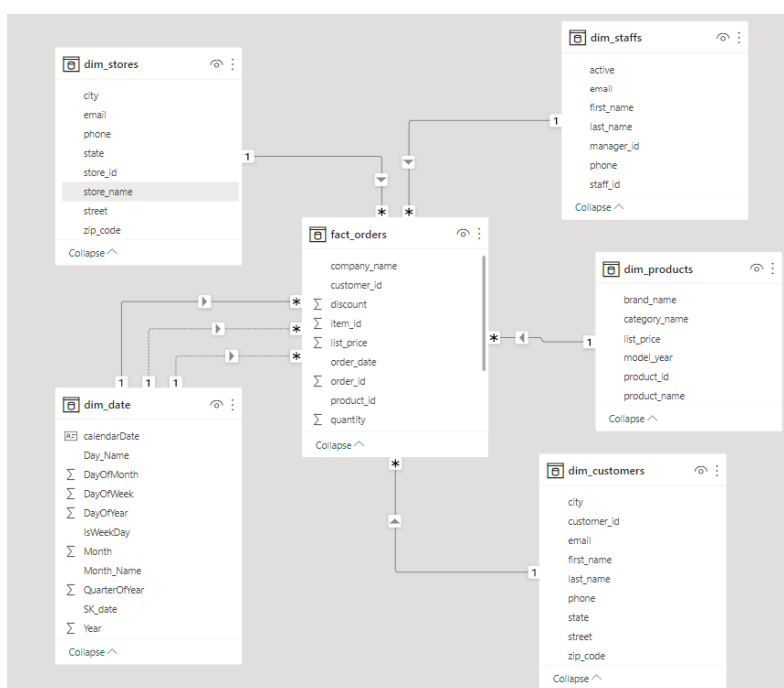


Figura 17 - Modelo Dimensional importado no PowerBI

As ligações entre as dimensões e facts table surgem automaticamente com a importação, com exceção da dim\_date, onde é necessário ligar a SK\_date com as 3 chaves de datas da facts table.



## 2.1.5 Componentes da seleção e decisão da Arquitetura

- GitHub: permite uma fácil colaboração de equipas de desenvolvimento, o tracking do histórico das alterações e o armazenamento dos dados em Cloud, de forma gratuita.

Documentação oficial: <https://docs.github.com/en/get-started>

- Databricks: fornece uma ingestão simplificada de diversas e múltiplas fontes, possui uma abordagem declarativa para construir pipelines e automatizar o ETL. Possibilita ainda uma ligação direta com o Power BI, permite a utilização de diferentes linguagens (e.g. Pyspark e SQL) e tem escalabilidade para Big Data.

Documentação oficial: <https://docs.databricks.com/introduction/index.html>

- Power BI: tecnologia líder de mercado como ferramenta de visualização, que permite a transformação de dados em informação coerente, visualmente envolventes e interativos. Acrescenta-se a facilidade de integração com as outras tecnologias escolhidas e o tipo de ficheiros (csv) utilizados.

Documentação oficial: <https://learn.microsoft.com/en-us/power-bi/>  
<https://powerbi.microsoft.com/en-us/why-power-bi/>

## 2.2 Data Flow Design

Na figura 12 apresenta-se o Data Flow esperado após toda a integração.

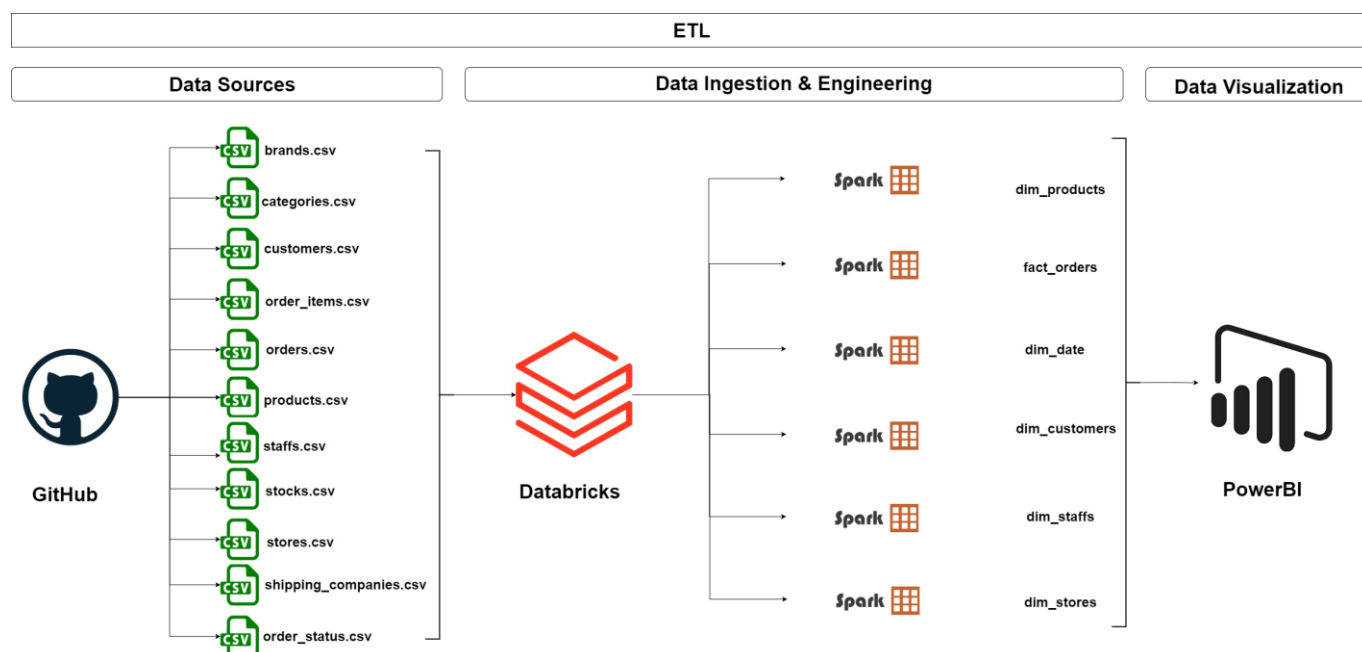





Figura 18 - Data Flow do projeto



Na Tabela 1 apresentam-se os inputs, caminhos e endpoints do ETL.

Tabela 1 - Inputs, caminhos e endpoints do ETL

 <b>GitHub</b>	 <b>databricks</b>	 <b>Power BI</b>	
<b>Inputs</b>	<b>Databricks File Path</b>	<b>Dataframes Path</b>	<b>Endpoint</b>
brands.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	PowerBI
categories.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
customers.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
order_items.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
orders.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
products.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
staffs.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
stocks.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
stores.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
shipping_companies.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	
order_status.csv	file:/databricks/driver/	dbfs:/user/hive/warehouse	

## Sobre Capgemini

A Capgemini é uma líder global em parcerias com empresas para transformar e gerir os seus negócios, aproveitando o poder da tecnologia. O Grupo é guiado diariamente pelo objetivo de libertar a energia humana através da tecnologia para um futuro inclusivo e sustentável. É uma organização responsável e diversificada com mais de 270.000 profissionais em quase 50 países. Com uma forte herança de 50 anos e profunda experiência no setor, a Capgemini tem a confiança dos seus clientes para responder a toda a amplitude das suas necessidades de negócio, desde estratégia e desenho até operações, alimentada pelo mundo inovador e em rápida evolução de *cloud*, dados, IA, conectividade, software, engenharia digital e plataformas. O Grupo reportou em 2020 uma receita global de € 16 mil milhões.

Get the Future You Want | [www.capgemini.com](http://www.capgemini.com)



Este documento contém informações que podem ser privilegiadas ou confidenciais e a propriedade é do Grupo Capgemini.

**Choose an item.** Copyright © 2021 Capgemini. Todos os direitos reservados.