Projeto final - Analista Big Data

Introdução

Neste projeto trabalhei com dados da empresa **Tokio School Viagens** especificamente com informação do tráfego no aeroporto de São Francisco, EE.UU. O objetivo deste trabalho foi aplicar diferentes ferramentas e técnicas de tratamento de dados, para achar padrões, tirar insights e conseguir entender o nosso conjunto de dados.

Carga e visualização dos dados

Para começar, decidi trabalhar com a biblioteca **Pandas** para carregar o ficheiro e obter informação dos dados, porque considero-á fácil de utilizar.
Com uma visualização de dados mais clara que PySpark, e tem comandos básicos que ajudam a conhecer os dados e a sua estrutura, sem ter que aplicar muito código, ganhando tempo.

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal
0	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1
1	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1
2	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1
3	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1
4	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1

**Parte da tabela demonstrando como devolve os dados a biblioteca Pandas*

Com ajuda do comando *unique()* conheci os valores únicos em colunas que achei necessário para a descrição das mesmas.

```
raw_data["GEO Summary"].unique()
```

```
array(['Domestic', 'International'], dtype=object)
```

```
raw_data["Price Category Code"].unique()
```

```
array(['Low Fare', 'Other'], dtype=object)
```

```
raw_data["Boarding Area"].unique()
```

```
array(['B', 'G', 'A', 'E', 'C', 'F', 'Other', 'D'], dtype=object)
```

Após visualizar o conteúdo dos nossos dados, prossigo a fazer uma descrição da estrutura e categoria de cada coluna.

Nome do campo	Tipo de dado
Activity Period	Numérico discreto. Descreve em qual mês e ano foi feito o voo. Dá informação sobre temporalidade, para poder realizar uma análise através do tempo.
Operating Airline	Categórico nominal. Nome da companhia responsável pelo transporte dos passageiros. Permite categorizar os nossos dados.
Operating Airline AITA Code	Categórico nominal. É o acrónimo da companhia aérea.
Published Airline	Categórico nominal. Nome da empresa que faz a gestão comercial do voo.
Published Airline AITA Code	Categórico nominal. Acrónimo da empresa responsável pela gestão comercial.
GEO Summary	Categórico binário. Domestic | International Descreve o tipo geográfico do voo.
GEO Region	Categórico nominal. Devolve a área geográfica do voo.
Activity Type Code	Categórico nominal. Informa o estado do voo, como se posou, se decolou, ou se continua em andamento.
Price Category Code	Categórico binário. Low Fare | Other Define qual é o tipo económico de voo, em relação à se for de ‘baixo custo’ ou outro tipo.
Terminal	Categórico nominal. Informação que diz a localização do avião na hora de saída e de chegada.
Boarding Area	Categórico ordinal. Existe uma ordem alfabética na classificação das áreas de abordagem, é da-nos informação de onde os passageiros ingressaram ao avião.
Passenger Count	Numérico discreto. Conseguimos ver a quantidade suposta de passageiros para o voo.

Nome do campo	Tipo de dado
Adjusted Activity Type Code	Categórico nominal. Estado atualizado do voo.
Adjusted Passenger Count	Numérico discreto. Quantidade atualizada de passageiros do voo.
Year	Numérico discreto. Ano ligado ao período de funcionamento.
Month	Numérico discreto. Mês ligado ao período de funcionamento.

Processamento de dados

Antes de inserir os dados em qualquer Base de Dados, é importante fazer um tratamento e limpeza desses dados.
Com o comando *info()* observei que existem duas colunas com valores nulos, para certificar-me apliquei a função *isnull()* que devolve um df com valores booleanos em cada registo, e em conjunto com *sum()* conheci somente a contagem de valores nulos para cada coluna

```
# visualização valores nulos
raw_data.isnull().sum()
```

```
Activity Period                0
Operating Airline              0
Operating Airline IATA Code    54
Published Airline              0
Published Airline IATA Code    54
GEO Summary                    0
GEO Region                     0
Activity Type Code             0
Price Category Code            0
Terminal                       0
Boarding Area                  0
Passenger Count                0
Adjusted Activity Type Code    0
Adjusted Passenger Count       0
Year                           0
Month                          0
dtype: int64
```

Conhecendo quais são as colunas com valores nulos, apliquei uma pequena análise para perceber o porquê poderiam estar vazios, e com tratá-los.

- 'IATA Code' é um código internacional para as companhias aéreas de voos comerciais. O que sugere que as empresas envolvidas nesses registos tenham sido ligadas a atividades auxiliares aos aviões, passageiros e cargas.

- Visualizei quais são os registos com essas colunas vazias para assim entender melhor a falta desses valores, e como influência isso.

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region
148	200508	Boeing Company	NaN	Boeing Company	NaN	Domestic	US
6814	201005	Servisair	NaN	Servisair	NaN	Domestic	US
6815	201005	Servisair	NaN	Servisair	NaN	Domestic	US
6925	201006	Pacific Aviation	NaN	Pacific Aviation	NaN	International	Europe
6926	201006	Pacific Aviation	NaN	Pacific Aviation	NaN	International	Europe
7173	201008	Servisair	NaN	Servisair	NaN	Domestic	US

**Alguns resultados onde se observam as companhias e colunas com valores NaN*

- Após fazer uma pesquisa das companhias mencionadas nesses registos, soube que são empresas que prestam serviços secundários as operações, como carga, limpeza de aeronaves, abastecimento de combustíveis, entre outros. [1]
- Analisei também o número de passageiros num período dessas companhias, e notei valores baixos em comparação com as outras empresas, com código IATA.

Após a análise, decidi preencher os espaços vazios com o valor 'NC' em referência às empresas não comerciais, e seguindo a estrutura da coluna. Para manter todos os registos no DataFrame.

Antes de preencher os valores, foi preciso comprovar se existem já registos com esse código, então fiz a consulta correspondente e não devolveu nenhum resultado.

```
raw_data[raw_data["Operating Airline IATA Code"] == "NC"]
```

Preenchi os espaços vazios e criei uma variável (df) para trabalhar com ela e não com os dados crus.
Como tenho a certeza que os únicos valores nulos existentes são os que analisei anteriormente, utilizei o comando *fillna()* por ser mais simples.
Mas se for o caso de ter que preencher com valores diferentes, ou colunas diferentes, utilizaria outra técnica, definindo os diferentes valores e aonde fazer essas transformações.

```
# preencho valores nulos
df = raw_data.fillna("NC")
df.isnull().sum()
```

```
Activity Period                0
Operating Airline              0
Operating Airline IATA Code    0
Published Airline              0
Published Airline IATA Code    0
GEO Summary                    0
GEO Region                     0
Activity Type Code             0
Price Category Code            0
Terminal                       0
Boarding Area                  0
Passenger Count                0
Adjusted Activity Type Code    0
Adjusted Passenger Count       0
Year                           0
Month                          0
dtype: int64
```

**Contagem de valore nulos após transformação*

Outra transformação que achei importante para melhorar o trabalho futuro com os dados, foi modificar os valores da coluna 'Month' que estão com nome do mês, para o formato numérico, já que será mais fácil agrupar por trimestre, por exemplo, ou dar uma ordem sazonal.
Para isso apliquei o método *replace()* da biblioteca Pandas para trocar os meses pelo seu número correspondente. Logo transformo a coluna ao tipo *int*. [2]

```
df["Month Number"] = df["Month"].replace({"January": 1, "February" : 2,
                                          "March": 3, "April":4,
                                          "May":5, "June": 6, "July": 7,
                                          "August": 8, "September": 9,
                                          "October": 10, "November": 11,
                                          "December": 12
                                            })
df["Month Number"] = df["Month Number"].astype("int")
```

Para finalizar a transformação apaguei a coluna original 'Month' e troquei o nome da nova coluna para 'Month' e assim, manter a estrutura original da tabela.

```
df.drop(columns = "Month", inplace = True)
df.rename(columns = {"Month Number" : "Month"}, inplace = True)
```

```
df["Month"].head()
```

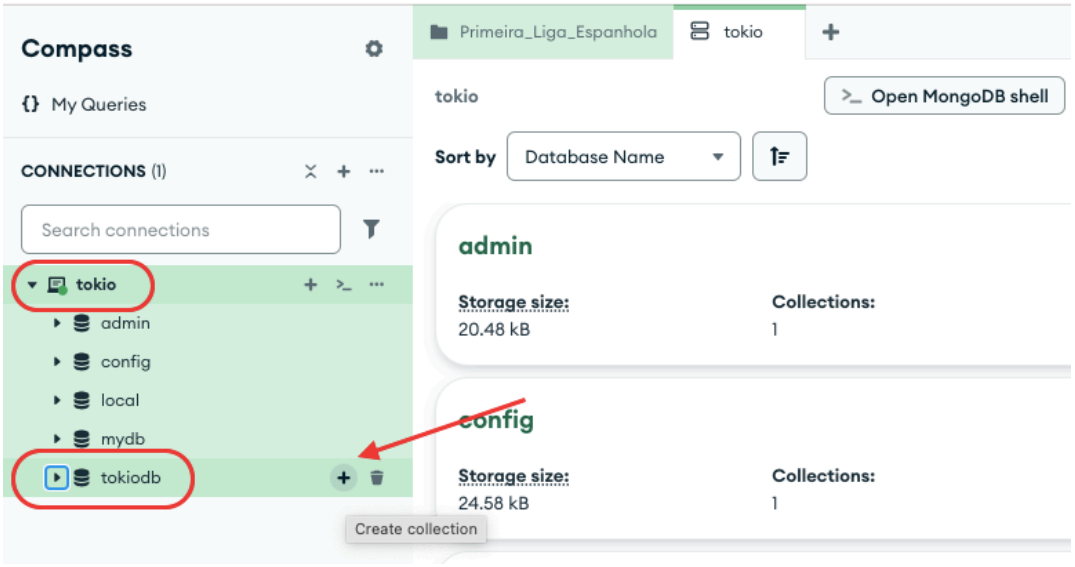
```
0    7
1    7
2    7
3    7
4    7
Name: Month, dtype: int64
```

Inserção de dados na Base de Dados - MongoDB

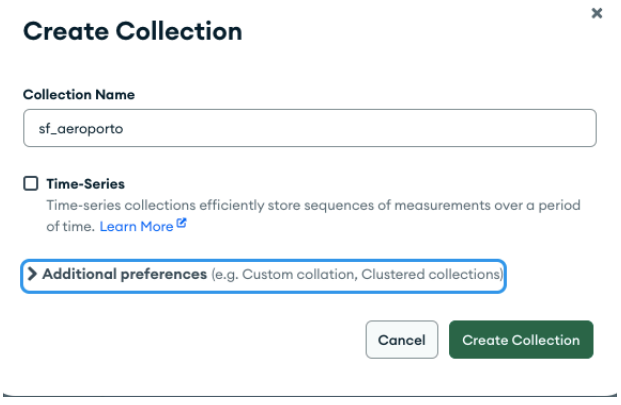
Decidi inserir os dados numa Base de Dados orientada a documentos, porque considero ser muito versátil, com uma fácil escalabilidade no sentido de acrescentar novas variáveis, sem ter que modificar a estrutura da collection.
Em parte também foi uma decisão pessoal de querer aprender como trabalhar em conjunto Python e Mongo, o que me levou a conhecer a biblioteca *pymongo*, que permite criar conexões com a DB.

Criação de 'collection'

Ao utilizar o aplicativo *MongoDB Compass* foi simples criar uma collection dentro da minha DB já existente chamada "tokiodb", foi clicar em alguns botões e escrever o nome, a seguir deixo uma breve explicação e fotos com o processo de criação.



Como descreve a foto, a minha conexão é chamada **"tokio"** e dentro dela tenho as minhas DB, uma delas é **"tokiodb"**, posso o rato por cima e monstra esse símbolo **+** aonde se clicar, crio uma collection.



Para finalizar defino o nome "**sf_aeroporto"** e aperto onde diz **"Create Collection"**.

Inserção de dados na DB

Para inserir os dados tenho que primeiro transformar ao formato JSON que são os tipos de dados com os que trabalha Mongo, para lográ-lo importei a biblioteca *json* e o seu módulo *loads*, em conjunto com Pandas consegui o meu objetivo.
Criando um arquivo do tipo JSON com comando *to_json()* e a seguir criei uma variável *parsed* que contem todos os dados no formato indicado. [3]

```
df_json = df.to_json(orient = "records")
parsed = loads(df_json)
```

Com os dados transformados criei uma função a modo de ser mais simples o processo no momento de inserir dados. Só é preciso o *URI* ou endereço de conexão com o MongoDB Compass, e os documentos em formato JSON a inserir. [4]

```
def insert_many(uri, docs):
    client = MongoClient(uri)
    try:
        database = client.get_database("tokiodb")
        collection = database.get_collection("sf_aeroporto")
        collection.insert_many(docs)
        client.close()
        print("Dados inseridos com sucesso")
    except Exception as e:
        print(f"Erro durante a conexão: {e}")
```

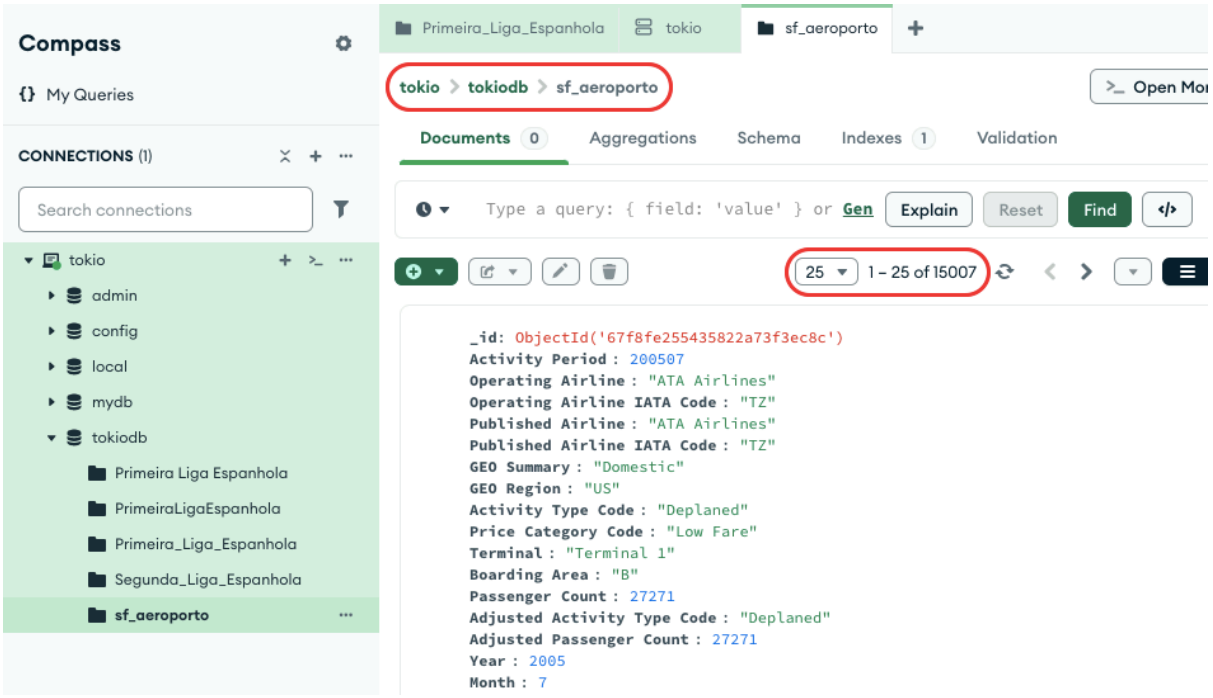
Apliquei a função e os dados foram inseridos.

```
uri = "mongodb://localhost:27017/"

insert_many(uri, parsed)
```

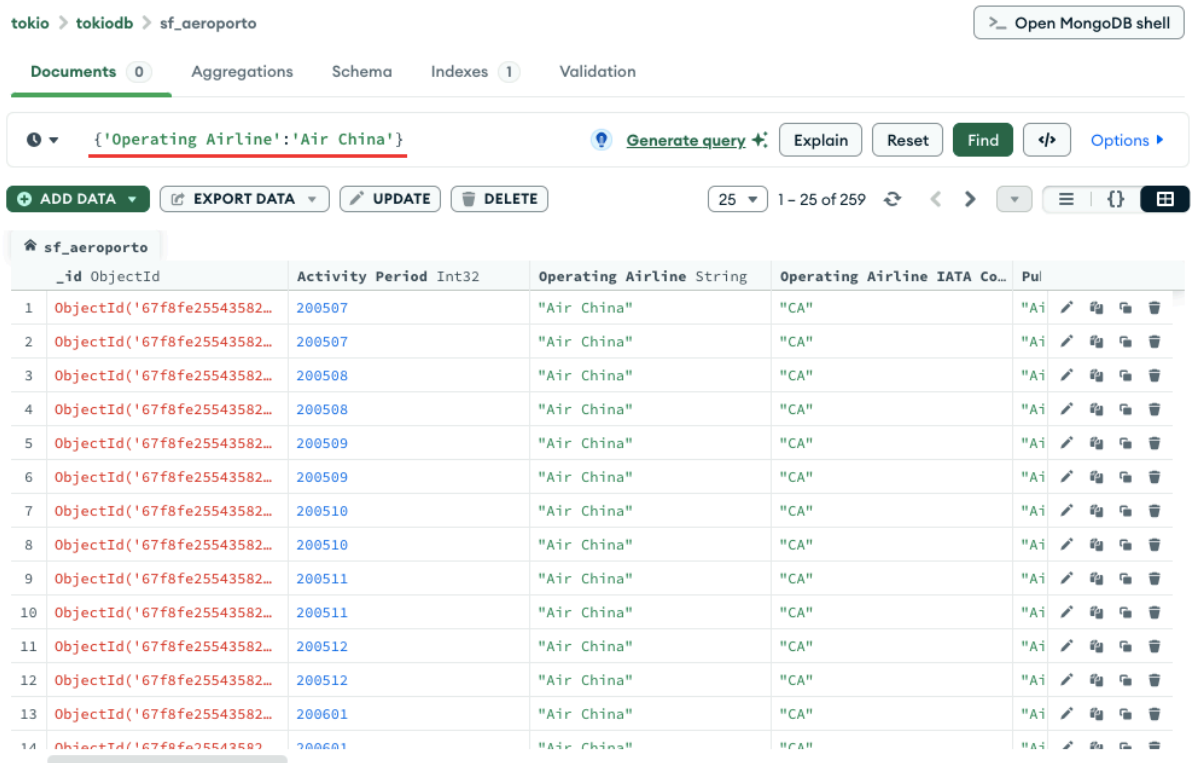
```
Dados inseridos com sucesso
```

Como observo na imagem abaixo, foram salvos os 15007 registos na minha coleção. Podendo continuar com as querys solicitadas.



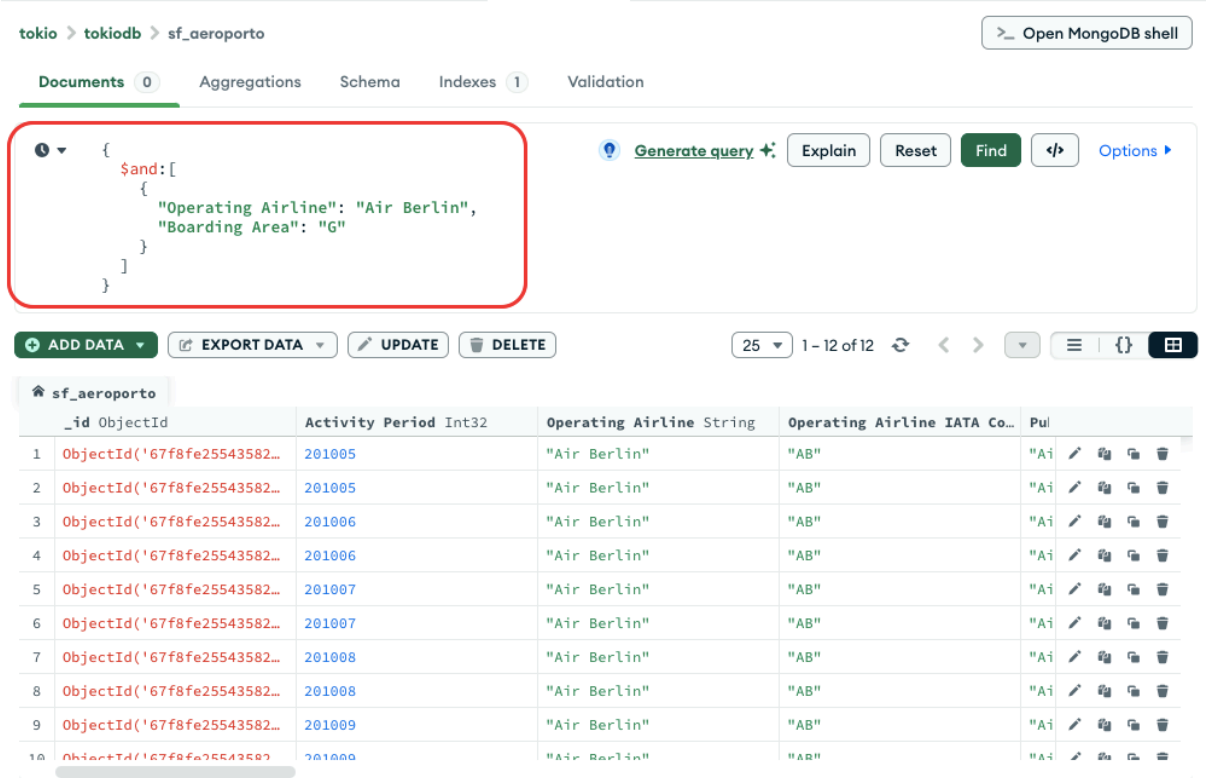
Consultas na minha DB

A) O MongoDB Compass permite realizar querys desde a interfase de uma maneira simples, como se deixa ver na imagem a seguir, temos um campo (sublinhado com vermelho) onde posso escrever a minha consulta, neste caso foi só obter os registos da companhia 'Air China'. Devolvendo 259 registos.



	_id ObjectId	Activity Period Int32	Operating Airline String	Operating Airline IATA Co…	Pul
1	ObjectId('67f8fe25543582…	200507	"Air China"	"CA"	"Ai
2	ObjectId('67f8fe25543582…	200507	"Air China"	"CA"	"Ai
3	ObjectId('67f8fe25543582…	200508	"Air China"	"CA"	"Ai
4	ObjectId('67f8fe25543582…	200508	"Air China"	"CA"	"Ai
5	ObjectId('67f8fe25543582…	200509	"Air China"	"CA"	"Ai
6	ObjectId('67f8fe25543582…	200509	"Air China"	"CA"	"Ai
7	ObjectId('67f8fe25543582…	200510	"Air China"	"CA"	"Ai
8	ObjectId('67f8fe25543582…	200510	"Air China"	"CA"	"Ai
9	ObjectId('67f8fe25543582…	200511	"Air China"	"CA"	"Ai
10	ObjectId('67f8fe25543582…	200511	"Air China"	"CA"	"Ai
11	ObjectId('67f8fe25543582…	200512	"Air China"	"CA"	"Ai
12	ObjectId('67f8fe25543582…	200512	"Air China"	"CA"	"Ai
13	ObjectId('67f8fe25543582…	200601	"Air China"	"CA"	"Ai
14	ObjectId('67f8fe25543582…	200601	"Air China"	"CA"	"Ai

B) Agora para conhecer os voos da companhia ‘Air Berlin’ que sairam pela porta de embarque ‘G’ apliquei a condição *$and* e devolveu um total de 12 registos.



Trabalhando no ambiente PySpark

Primeiro passo foi importar as bibliotecas pertinentes do módulo *pyspark.sql* e a seguir criei a sessão de **PySpark**.
Aproveitei o df transformado, com o que trabalhei até cá e criei o DataFrame de Spark com ele.[5]

Contagem de companhias

Para conhecer quantas companhias existem no ficheiro utilizei o comando *count_distinct()* que devolve contagem de valores únicos numa coluna. Como no ficheiro existem dois tipos de companhias, fiz a contagem das duas colunas. [6]

```
# número de companhias existentes no df
df_sp.select(f.count_distinct("Operating Airline")\
             .alias("Companhias operacionais"),
             f.count_distinct("Published Airline")\
             .alias("Companhias comerciais"))\
             .show()
```

```
+----------------------+--------------------+
|Companhias operacionais|Companhias comerciais|
+----------------------+--------------------+
|                    77|                  68|
+----------------------+--------------------+
```

Aplicando esse comando consegui saber que existem 77 companhias responsáveis pelas operações e 68 companhias a cargo da parte comercial.
Quer dizer que existem companhias que dependem de outras, ou são sub empresas que só ficam responsáveis pela parte operacional.
Ou pode ser, também, que não tem a estrutura suficiente para fazer-se cargo das operações e comercializar o seu serviço.
Para saber mais sobre o assunto seria pertinente fazer uma pesquisa mais detalhada dessas empresas em particular.

Média de passageiros em cada voo

Após analisar os dados que tenho disponíveis no ficheiro, não achei nenhuma referência a quantidade de voos num período, ou forma de calcular isso de uma forma certa. Pelo que não consegui calcular a média de passageiros em cada voo.
Mas posso, sim, calcular a média dentro de cada período e agrupado por companhia, o que achei mais apropriado a fazer, para isso agrupei os dados por período e por companhia com *groupBy()*, e a seguir calculei o valor médio sem decimais com *round(avg())*.

```
# média de passageiros por período e companhias
df_avg = df_sp.groupBy(["Activity Period", "Operating Airline"])\
                        .agg(f.round(f.avg("Adjusted Passenger Count"))\
                        .alias("Avg Passenger")
                            )\
                        .orderBy("Activity Period", ascending = False)
df_avg.show(5)
```

Salvei os resultados num ficheiro .csv com o comando *write() e defini as opções como overwrite* para fazer uma cópia sempre que for corrido o código, e não criar um ficheiro novo cada atualização dos dados

```
+---------------+-----------------+-------------+
|Activity Period| Operating Airline|Avg Passenger|
+---------------+-----------------+-------------+
|         201603|    Japan Airlines|       6286.0|
|         201603|         Emirates |      12632.0|
|         201603|Singapore Airlines|      12680.0|
|         201603|        Aeromexico|      10795.0|
|         201603| Hawaiian Airlines|      15525.0|
+---------------+-----------------+-------------+
```

**Média de passageiros por companhia dentro de cada período*

Apagar registos duplicados

Para apagar os registos onde se repete a região, mantendo os que tem o maior número de passageiros, utilizei o comando *dropDuplicates()* o qual para realizar este retiro de duplicados, a função deixa os primeiro valores únicos que aparecem na coluna e continua apagando os restantes, fazendo um mapeio de cima para baixo. [7]
Conhecendo essa propriedade, dei uma ordem numérica de maior até o menor, sobre a coluna *Adjusted Passenger Count*, e apliquei a função *dropDuplicates()* sobre a coluna *GEO Region*.

```
# apagando duplicados pela região
geo_regiao = df_sp.orderBy("Adjusted Passenger Count", ascending = False)\
                .dropDuplicates(["GEO Region"])
```

Com a mesma lógica que apliquei antes no comando *write()* vou utilizar o mesmo comando para salvar os resultados num ficheiro .csv

Operating Airline	GEO Region	Adjusted Passenger Count
United Airlines	US	659837
United Airlines -...	Asia	86398
United Airlines	Europe	48136
Air Canada	Canada	39798
United Airlines	Mexico	29206
Emirates	Middle East	14769
Air New Zealand	Australia / Oceania	12973
TACA	Central America	8970
LAN Peru	South America	3685

**Tabela sem regiões duplicadas*

Análise Descritiva dos dados com PySpark

Na minha análise, concentrei-me em calcular as médias e desvio padrão de passageiros, agrupando por diferentes caraterísticas, e tirando uma conclusão disso. Também investiguei a distribuição dos dados para conseguir obter uma melhor análise do aeroporto de São Francisco e os seus dados.

Este código foi o que utilizei para calcular as médias e desvio padrão de quase todas as categorias abaixo apresentadas, mudei as propriedades que iria analisar em cada caso.

```
# média de passageiros agrupados por companhias
df_sp.groupBy("Operating Airline")\
    .agg(
        f.round(f.avg("Adjusted Passenger Count")).alias("Media de passageiros"),
        f.round(f.stddev("Adjusted Passenger Count")).alias("Desvio padrão"),
        f.max("Adjusted Passenger Count").alias("Max"),
        f.min("Adjusted Passenger Count").alias("Min")
    )\
    .orderBy("Media de passageiros", ascending = False)\
    .show(15)
```

Onde agrupei pela caraterística a estudar, criei colunas com os cálculos da média *avg()*, desvio padrão *stddev()*, máximo *max()* e mínimo *min()*, ordenando pela média dos passageiros para visualizar os que tem maiores resultados no topo.

Resultados agrupados por companhia aérea

Operating Airline	Media de passageiros	Desvio padrão	Max	Min
American Airlines	127164.0	22044.0	167225	36425
Southwest Airlines	81223.0	60311.0	159846	2
Virgin America	74405.0	68540.0	194636	103
United Airlines	72827.0	111353.0	659837	2
Delta Air Lines	68515.0	52420.0	186774	82
US Airways	55318.0	17369.0	89861	20944
United Airlines -...	49366.0	101159.0	443656	2
SkyWest Airlines	37084.0	47114.0	167699	14
JetBlue Airways	35261.0	15782.0	64326	82
Northwest Airlines	26206.0	23201.0	70641	216
Compass Airlines	23360.0	13641.0	43069	86
Lufthansa German ...	19302.0	3158.0	26240	11651
Air Canada	18252.0	8036.0	39798	4915
Frontier Airlines	17788.0	4894.0	35735	4105
British Airways	17625.0	2490.0	22772	10520

only showing top 15 rows

Este cálculo devolveu alguns resultados onde o desvio padrão é maior que a média e por muita diferença, para entender melhor, acrescentei as colunas do valor máximo e do valor mínimo.
Assim, observando melhor, posso notar que existe uma amplitude de valores muito grande nesses registos onde o desvio é maior que a média.

Concluindo que essas companhias, tem uma variação muito grande em diferentes períodos. Estas diferenças em quantidade de passageiros, sugere ser por uma sazonalidade, onde tem períodos do ano onde tem um aumento significativo.
Ou também porque tem alguns meses com preços mais baratos, ofertas e promoções, que faz com que as vendas subam mais do normal, já que existem companhias ‘baixo custo’ onde é normal ter voos com poucos passageiros na baixa temporada.

Resultados agrupados por região

GEO Region	Media passageiros	Desvio padrão	Max	Min	Mediana
US	58486.0	84860.0	659837	1	24061.0
Asia	13509.0	16148.0	86398	2	8011.0
Europe	12779.0	8602.0	48136	13	11023.0
Canada	9804.0	7806.0	39798	2	8807.5
Middle East	8659.0	2733.0	14769	82	9244.5
Mexico	7251.0	5274.0	29206	2	6505.0
Australia / Oceania	6495.0	2650.0	12973	57	6889.0
Central America	4947.0	1221.0	8970	234	4862.0
South America	2786.0	397.0	3685	1623	2787.0

A mediada devolve o valor na posição media da coluna, é calculada agrupando todos os valores de menor a maior, e devolve o valor na posição **n/2**. O que nos dá uma ideia de como é a dispersão do conjunto de dados.

Posso observar que os voos com destino ‘US’ são os que presentam uma dispersão bem maior que as outras, não só pelo desvio dos valores, se não também no valor máximo e na mediana.
Analisando a mediana posso entender um pouco melhor se a dispersão é devido a valores outlier muito altos, ou baixos. Se a mediana é muito menor que a média, sei que existem poucos valores atípicos muito altos fazendo aumentar o desvio padrão.

Sabendo isso, vou concluir com que os voos dentro dos US, domésticos, tem uma grande dispersão devido a que existem períodos onde tem uns valores de passageiros atípicos. Penso que está ligado a ofertas, promoções, eventos importantes.
Enquanto as outras regiões, além de algumas, ter uma dispersão amplia, a mediana está perto das médias, pelo que posso dizer que os voos internacionais, tem uma frequência mais estável, e a sua variação pode ter mais a ver com sazonalidade que com campanhas de marketing das empresas.

Voos domésticos agrupados por mês.

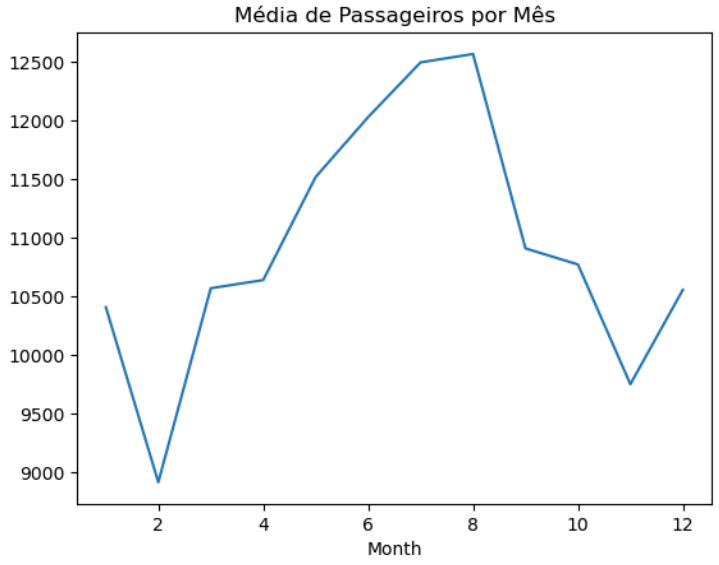
Month	Media passageiros	Desvio padrão	Max	Min
1	50977.0	74825.0	482915	1
2	49414.0	69302.0	351096	2
3	60007.0	83464.0	392302	7
4	57304.0	81233.0	397287	8
5	60135.0	85401.0	404581	4
6	64291.0	89146.0	433650	5
7	65662.0	95842.0	625885	1
8	64299.0	95609.0	659837	6
9	56724.0	84837.0	548374	6
10	59622.0	88111.0	573619	4
11	55669.0	81310.0	522199	1
12	57761.0	83880.0	556208	2

Para aprofundar na análise dos voos domésticos, observo que sim existe uma tendência de sazonalidade, onde nos meses de Julho e Agosto (verão), sobem a média, a máxima e o desvio padrão forte, em comparação aos outros meses.
Chegando a conclusão de que durante o verão existe uma dispersão de valores maior baseado na contagem de passageiros.

Resultados agrupados por tipo de voo

GEO Summary	Media passageiros	Desvio padrão	Max	Min	Mediana
Domestic	58486.0	84860.0	659837	1	24061.0
International	10982.0	11460.0	86398	2	7874.5

Comparando os voos domésticos e internacionais, posso concluir que a média de passageiros nas ligações internacionais, existe uma certa homogeneidade ao redor da média, mas ainda assim existem períodos onde aumenta a quantidade de clientes.
Isso pode indicar que existe um tipo de sazonalidade, ligado aos meses do verão ao igual que os voos nacionais. Para isso vou fazer um gráfico desses dados e tentar perceber melhor.



**Voos internacionais*

Olhando no gráfico, existe uma tendência de aumento de passageiros nos meses de verão, concluindo com mais segurança que os voos internacionais, igual que os domésticos, têm uma influência temporal.

Considero que este pico de tráfego está ligado as ofertar turísticas que existem na cidade de São Francisco, como nos EE.UU., em geral, durante o verão.

Distribuição dos dados

Para calcular a distribuição dos dados do conjunto, através da biblioteca PySpark apliquei agrupamentos com *groupBy()*, contagem com *count()*, *filter()* para filtrar em alguns casos, e operações matemáticas normais para obter as frequências relativas.
Estes comandos foram basicamente os que utilizei para estudar as distribuições dos dados.

```
dis_geo = df_sp.groupBy("GEO Summary")\
        .agg(f.count("GEO Summary").alias("Freq Absoluta"),\
             f.round(f.count("GEO Summary") * 100 / 15007 ,2)\
             .alias("Freq Relativa"))

dis_geo.show()
```

Para fazer gráficos com esses resultados utilizei a biblioteca Pandas, porque foi mais simples, para mim, visualizar os resultados de uma forma mais limpa e poder utilizar isso na apresentação. [8]

Distribuição do tipo geográfico de voos

GEO Summary	Freq Absoluta	Freq Relativa
International	9210	61.37
Domestic	5797	38.63

Continuando com a análise descritiva dos dados, querendo conhecer a distribuição dos mesmo, comecei calculando os tipos geográficos de voos, e olhando os resultados existe quase o dobro na frequência de voos internacionais, em comparação com os domésticos. Sugerindo que pode ser pela localização da cidade de San Francisco, sendo assim, um aeroporto de 'entrada' aos EE.UU. onde os voos internacionais fazem algum tipo de escala, em conjunto também com o turismo que a cidade oferece, atraindo muito estrangeiro.

Visando reforçar a análise, criei uma tabela pivot, onde calculo a contagem de registos dentro de cada mês, agrupados por voos do tipo Internacional ou Doméstico.

```
# crio tabela pivot para dividir os dados pelo tipo de voo
pivot = df_sp.groupBy("Month")\
        .pivot("GEO Summary")\
        .agg(f.count("GEO Summary")\
        .alias("Contagem"))\
        .orderBy("Month")

pivot.show()
```

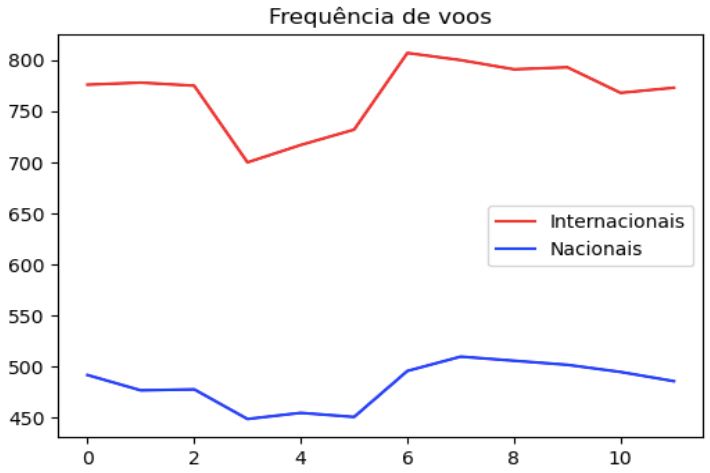
Month	Domestic	International
1	492	776
2	477	778
3	478	775
4	449	700
5	455	717
6	451	732
7	496	807
8	510	800
9	506	791
10	502	793
11	495	768
12	486	773

A seguir criei um gráfico com a biblioteca *Matplotlib* onde posso observar os padrões de frequência desses voos.

```
import matplotlib.pyplot as plt
# gráfico comparativo na frequência de voos
pivot_pd = pivot.toPandas()

y_values = pivot_pd["International"].values.tolist()
y_values2 = pivot_pd["Domestic"].values.tolist()

plt.figure(figsize = (6,4))
plt.plot(y_values, "red", label = "Internacionais")
plt.plot(y_values2, "blue", label ="Nacionais")
plt.title("Frequência de voos")
plt.legend()
plt.show()
```



Visualizando os gráficos posso notar que os padrões são semelhantes entre eles, pelo que posso concluir que no aeroporto de São Franciso são feitas muitas conexões, onde o destino não é essa cidade, sem importar a sazonalidade.

Distribuição dos voos internacionais

GEO Region	Freq Absoluta	Freq Relativa
Asia	3273	21.81
Europe	2089	13.92
Canada	1418	9.45
Mexico	1115	7.43
Australia / Oceania	737	4.91
Central America	274	1.83
Middle East	214	1.43
South America	90	0.6

Considerando os resultados das regiões dos voos internacionais, observo que as rotas que temos com Ásia são significativamente as mais frequentes, com quase 22% de registos no conjunto de dados, seguido da Europa com 14%.
Isso indica que a ideia de o aeroporto de SF ser um aeroporto de escala, é um fator importante nas rotas com Ásia, pela cercania entre este continente e a costa oeste dos EE.UU.

Resumo análise descritiva

Existem períodos com valores atípicos no número de passageiros, principalmente nos voos domésticos. Esses picos são muito altos, o que faz com que o desvio padrão seja maior que a média. Acredito que isso aconteça devido à sazonalidade, já que o tráfego de passageiros aumenta no verão. Além disso, promoções e campanhas de marketing também podem influenciar essa grande variação.

Os dados devolvem que a maioria dos voos registados são internacionais, com um volume maior vindo da Ásia. Isso faz sentido, já que San Francisco está relativamente próxima desse continente e acaba funcionando como uma porta de entrada para os EE.UU.

Mesmo com mais passageiros em voos domésticos, há mais registos de voos internacionais. Isso sugere que o Aeroporto de San Francisco é um aeroporto importante e com muitas conexões, tanto internacionais como nacionais, onde vários voos fazem escala.

Análise de correlação

Para conseguir calcular uma correlação entre variáveis é obrigatório trabalhar com valores numéricos, para resolver isso nos meus dados, fiz a transformação de string para números, com a classe *StringIndexer* do módulo de **Machine Learning** de **PySpark**.

O que faz esta classe é mapear os valores *string* que existem numa coluna é assina um número diferente para cada valor, dependendo da frequência desse valor.
Existindo tantas colunas do tipo *string* e com vários valores em algumas delas, achei esta a maneira mais eficaz e com uma boa escalabilidade.

Antes de aplicar a classe, criei uma tabela sem os códigos IATA, já que são identificadores categóricos sem alguma relação numérica direta, além de devolver correlações enviesadas com as colunas das companhias.
Para não devolver correlações enviesadas, também retirei as colunas de passageiros, e atividade, porque dependem de outras colunas e iria atrapalhar os resultados, ao igual que a coluna Activity Period, que não considerei, para considerar as colunas Year e Month.

Para realizar a transformação defini como vai trabalhar o modelo *StringIndexer* dando uma lista com os valores das colunas para transformar, e uma lista com os nomes das colunas de saída.
Logo treino o modelo com *fit()* utilizando os dados, e por último transformo o meu df *transform()*.

```
input = ["Operating Airline", "Published Airline", "GEO Summary", "GEO Region",
        "Activity Type Code", "Price Category Code",
        "Terminal", "Boarding Area"]
output = ["operator_ind", "published_ind ", "geo_summary_index", "region_index",
        "activity_index", "price_index",
        "terminal_index", "boarding_index"]

# defino o objeto StringIndexer e treino ele com os meus dados
indexer = StringIndexer(inputCols = input, outputCols = output)
si_model = indexer.fit(corr)

# transformo o df para valores numéricos
corr = si_model.transform(corr)
```

Como mencionei antes, este modelo *StringIndexer* vincula um valor dependendo da frequência dessa caraterística. Então para saber qual valor foi vinculado com cada caraterística, fiz um print dos elementos *labelsArray()*, um método do modelo que devolve as caraterísticas por ordem de frequência começando por 0. [9]
Com esta guia vou poder analisar os resultados da minha matriz de correlação.

```
# visualizo frequência de variáveis segundo StringIndexer
for i in si_model.labelsArray:
    print (i)
```

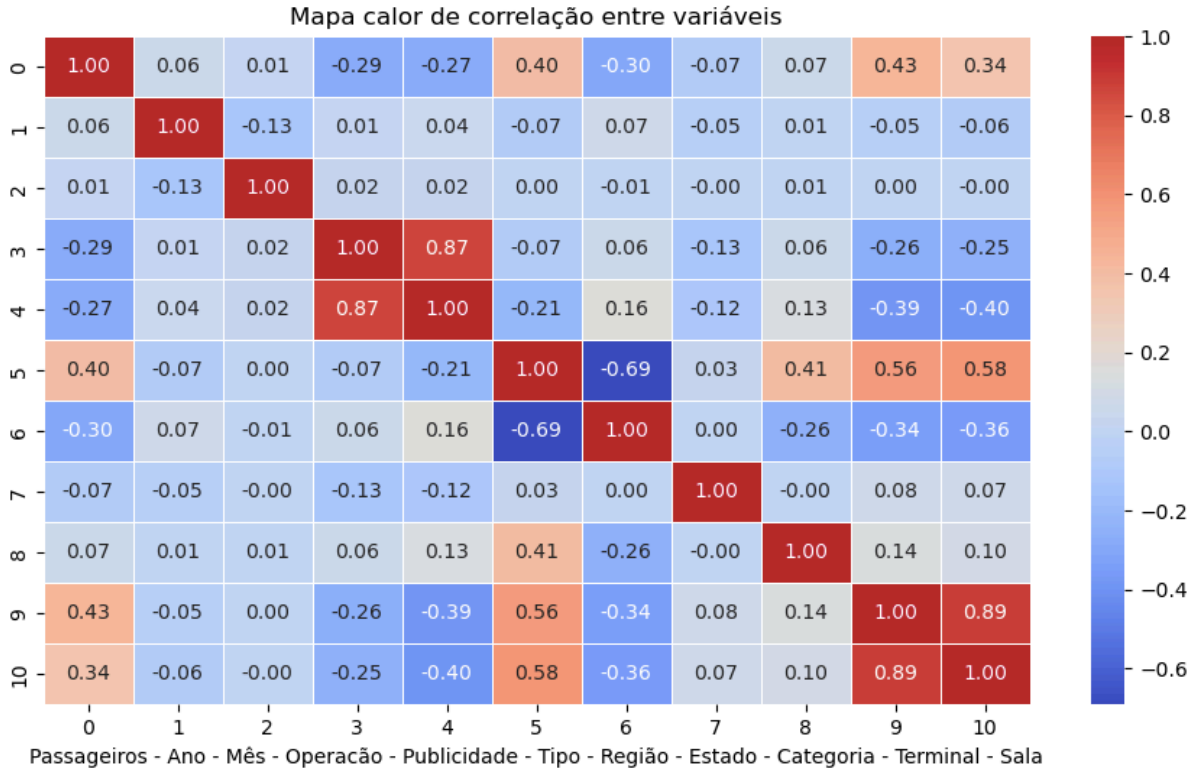
```
('International', 'Domestic')
('US', 'Asia', 'Europe', 'Canada', 'Mexico', 'Australia / Oceania', 'Central America', 'Middle East', 'South America')
('Deplaned', 'Enplaned', 'Thru / Transit')
('Other', 'Low Fare')
('International', 'Terminal 1', 'Terminal 3', 'Terminal 2', 'Other')
('A', 'G', 'B', 'F', 'C', 'E', 'D', 'Other')
```

Matriz de correlação

Para obter uma matriz de correlação, utilizei a biblioteca **Numpy**.
Apaguei as colunas com dados do tipo *string* e transformei o df para um de Pandas, para a biblioteca Numpy poder trabalhar com os dados e calcular as correlações.
Agora para criar a matriz foi preciso o comando *np.corrcoef()* e definir que os valores a calcular estão divididos por colunas e não por linhas.

```
# calculo matriz de correlação
matriz = np.corrcoef(corr_pd, rowvar = False)
```

Achei que fazer um mapa de calor com a matriz criada era a forma mais fácil de interpretar os dados.
Para isso foram necessárias as bibliotecas especializadas em gráficos como **Matplotlib** e **Seaborn**.
Com elas defini os parâmetros para o mapa de calor, como o tamanho, paleta de cores, tamanho dos valores, título e labels do eixo X. [10]



Como vejo no mapa de calor, as variáveis com a correlação positiva mais forte são a terminal e a sala de abordagem (0.89), tendo lógica, já que para manter uma organização os aeroportos separam as áreas por voos internacionais ou nacionais. Então, as terminais e salas de abordagem sempre estão destinadas ao mesmo tipo de voo.

Seguido elas, estão as colunas Operação e Publicidade (0.87), que fazem referência as empresas aéreas, o qual têm sentido existir uma relação positiva tão forte, porque geralmente as empresas que fazem as operações, também fazem a parte comercial, então acabam sendo os mesmos resultados nas colunas. Da para ver também que tem valores similares com as demais variáveis, o que me diz que poderia não considerar uma delas na aplicação do meu modelo de Aprendizagem Automático.

Existe sim uma relação negativa não muito forte (-0.29) com a contagem de passageiros, dando a entender que as companhias que mais vezes aparecem no nosso ficheiro, são as que mais passageiros operam por período. Acho que tem coerência, já que são as empresas nacionais, que fazem mais voos domésticos neste aeroporto.

Analisando a contagem de passageiros, vejo 3 variáveis com uma correlação positiva, uma é o tipo de voo (0.40), representando que se o voo for doméstico, tem maior probabilidade de ter mais número de passageiros. As outras duas variáveis positivas são a terminal e sala de abordagem (0.43, 0.34), ligadas diretamente ao tipo de voo como mencionei acima, pelo que comparte a correlação positiva junto ao tipo
Em contrapartida, a relação dos passageiros com a região é negativa (-0.3), o que nos diz que os períodos com maior quantidade de passageiros ocorre nos voos dentro de EE.UU.

Pelo que observo, a temporalidade (mês e ano) em termos gerais, tendo em consideração todos os dados, não tem alguma influência significativa em nenhuma variável.

A correlação que existe entre o tipo de voo e a região, é negativa e quase (-0.7) demonstrando que os voos internacionais têm uma maior probabilidade de ser de Ásia, Europa e Canada que ser voos provenientes de America do Sul e Middle East.

Olhando a correlação do tipo com a categoria do voo (0.41), os que têm destino doméstico, é mais propenso a ser um voo ‘baixo custo’.
Então o tipo de voo pode nos dar informação de qual é a região de ligação e a categoria do mesmo.

Por último, a variável de ‘Estado’ não brinda informação nenhuma, sendo que brinda informação aleatória, sem ligação a outra variável.

Aplicação do Modelo de Machine Learning

A minha escolha de qual modelo aplicar foi baseada em achar padrões nos dados, analisando quais propriedades exercem maior influência e como essas caraterísticas se relacionam hierarquicamente. Para isso utilizei o modelo **BisectingKMeans** da biblioteca PySpark ML. [11]

Esse é um modelo de aprendizagem automático não supervisionado que faz agrupamentos por semelhança das propriedades, conhecido como **Agrupamento Hierárquico**.
Especificamente apliquei a metodologia de **aglomeração** (bottom-up), gerando clusters progressivamente a partir de grupos individuais até chegar ao número ideal de grupos (K) com maior eficácia.

Para poder medir o rendimento do modelo apliquei a medição **Silhouette Score** que mede a separação entre os clusters, calculando a proximidade dos objetos num mesmo grupo e a distância em relação aos demais agrupamentos.

Pre-processamento dos dados

Para poder treinar um modelo de ML é preciso ter os dados num formato onde possa analisar os mesmos, além de ser do tipo numérico, é preciso criar um vetor com as variáveis que queremos definir para o nosso modelo aprender.

As bibliotecas que utilizei foram todas do módulo **ML** de **PySpark**, sendo então, **BisectingKMeans, ClusteringEvaluator, StringIndexer** e **VectorAssembler.**
Igual que para a matriz de correlação, transformei as colunas do tipo *string* para tipo numérico com *StringIndexer*, para logo a seguir criar o vetor com *VectorAssembler*, onde defini as colunas de entrada, e a coluna de saída *feature*.

```
indexer = StringIndexer(inputCols = inputs, outputCols = outputs)
ind_modelo = indexer.fit(df_sp)

indexer_df = ind_modelo.transform(df_sp)
```

```
assembler = VectorAssembler(inputCols = features, outputCol = "features")
va_df = assembler.transform(indexer_df)
```

Após a minha análise e exploração dos dados, decidi treinar o modelo com os registos de voos internacionais. Porque achei que se treino com todo o universo de dados, vai devolver resultados enviesados pelos valores de passageiros nos voos domésticos.

Com os dados já prontos, procurei o número mais efetivo de *'K'* modelando diferentes algoritmos, modificando o número de K e calculando o *Silhouette Score.* Com a classe *ClusterinEvaluator* posso calcular esta eficácia.

```
# objeto para calcular o Silhouette
evaluator = ClusteringEvaluator(predictionCol = "prediction",
                                featuresCol = "features",
                                metricName = "silhouette")
```

Criei um objeto com os parâmetros para calcular o *Silhouette* e a seguir um *loop for* para treinar os modelos e devolver a eficácia de forma automática, com K entre 3 e 6 (escolha de números aleatória).

```
# loop para conhecer eficácia de diferentes K
for k in range(3, 7):
    bkm = BisectingKMeans().setK(k).setSeed(1)
    modelo = bkm.fit(inter)
    prediction = modelo.transform(inter)
    silhouette = evaluator.evaluate(prediction)
    print(f"K={k}, Silhouette Score={silhouette}")
```

```
K=3, Silhouette Score=0.7961512365681664
K=4, Silhouette Score=0.7912384379590824
K=5, Silhouette Score=0.619050814471425
K=6, Silhouette Score=0.6627470837342898
```

Quanto mais perto do 1 está a eficácia, significa que o modelo agrupou melhor os objetos baseado nas propriedades. Por isso tomei a decisão de aplicar *setK(3)* no modelo, e treinar ele com o conjunto de dados *fit()*, para logo obter os agrupamentos com *transform()*.
A seguir, a função *clusterCenters()* do modelo, devolve os valores dos *centroides* de cada coluna, representando como foram agrupados os registos. Isso devolve três matrizes com os dados de cada coluna na mesma ordem que foi criado o vetor *features*. [12]

```
bkm = BisectingKMeans().setK(3).setSeed(1)
modelo = bkm.fit(inter)
prediction_inter = modelo.transform(inter)
silhouette = evaluator.evaluate(prediction_inter)
print(f"K=3, Silhouette Score={silhouette}")

print("Cluster Centers: ")
centers = modelo.clusterCenters()
for center in centers:
    print(center)
```

```
K=3, Silhouette Score=0.7961512365681664
Cluster Centers:
[2.01055976e+05 1.86450853e+01 1.71771630e+01 0.00000000e+00
 2.78943496e+00 6.02413184e-01 2.54561507e-02 2.01589170e-01
 6.92907593e-01 6.50573867e+00 6.46349985e+03]
[2.01077854e+05 1.40821853e+01 1.34893112e+01 0.00000000e+00
 2.08076010e+00 5.08788599e-01 0.00000000e+00 2.74584323e-01
 1.02280285e+00 6.64275534e+00 1.80616679e+04]
[2.01071715e+05 8.64077670e-01 6.01941748e-01 0.00000000e+00
 1.21359223e+00 4.91909385e-01 3.23624595e-03 9.06148867e-02
 1.18122977e+00 6.66666667e+00 6.21219644e+04]
```

Para visualizar e achar padrões através dos valores dos centroides, apliquei o cálculo da média nas propriedades e criei um dataframe de Pandas para visualizar de uma maneira mais limpa.

prediction	Periodo	Operador	Regiao	Estado	Categoria	Terminal	Embarque	Mês	Media Passageiros
0	201056.0	18.65	2.79	0.60	0.0	0.20	0.69	7.0	6463.0
1	201078.0	14.08	2.08	0.51	0.0	0.27	1.02	7.0	18062.0
2	201072.0	0.86	1.21	0.49	0.0	0.09	1.18	7.0	62122.0

**Tabela com valores dos centroides de cada coluna, agrupado por grupo*

A tabela apresenta os centroides de cada cluster, refletindo as características médias dos voos pertencentes a cada grupo. Analisar esses valores permite compreender os padrões que orientaram ao modelo na criação dos agrupamentos com base nas suas propriedades principais.

- **Cluster 0** ⇒ Representa voos com **menor volume médio de passageiros**, maior diversidade regional e uma distribuição estável das salas de embarque. Isso sugere que esse grupo inclui voos de menor demanda e potencialmente maior variabilidade de companhias operacionais.

- **Cluster 1** ⇒ Caracteriza voos com um **volume intermediário de passageiros**, operados por diversas companhias aéreas e atendendo a múltiplas regiões. A dispersão das portas de embarque sugere que esses voos utilizam diferentes terminais recorrentemente, podendo estar associados a períodos de variação moderada na demanda.

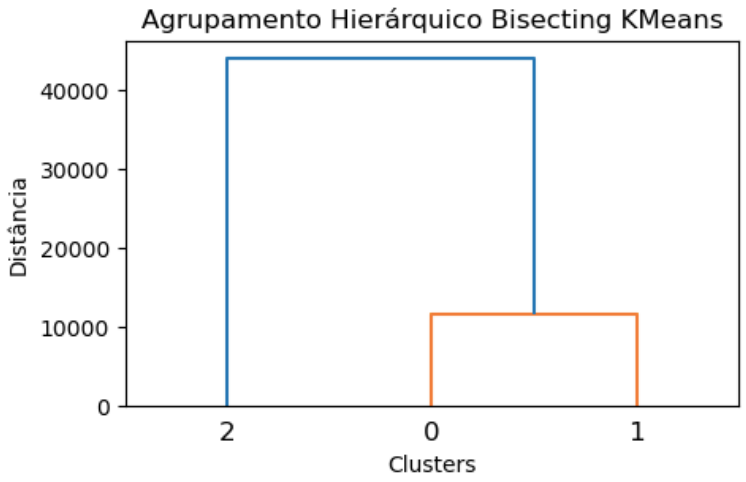
- **Cluster 2** ⇒ Agrupa os voos com **maior número médio de passageiros**, onde as companhias aéreas e as regiões atendidas apresentam um padrão mais concentrado. Isso sugere a possibilidade de rotas estratégicas operadas por empresas específicas, relacionadas a períodos de alta demanda ou operações de grande escala.

Para visualizar a distância dos grupos, e poder dar uma melhor demonstração de como trabalha este tipo de algoritmo na apresentação, criei um gráfico com ajuda da biblioteca *SciPy*. Podendo observar como os clusters 0 e 1 têm características mais próximas que o cluster 2 que está mais longe (dados atípicos).

```
# gráfico dendograma
from scipy.cluster.hierarchy import dendrogram, linkage

linkage = linkage(modelo.clusterCenters(), "single")

plt.figure(figsize =(5,3))
dendrogram(linkage)
plt.title("Agrupamento Hierárquico Bisecting KMeans")
plt.xlabel("Clusters")
plt.ylabel("Distância")
plt.show()
```



Exploração das propriedades

Para conseguir realizar a exploração apliquei comandos como *count(), avg(), stddev(), agg(), groupBy()* e *orderBy()* e logo poder criar gráficos com *Pandas* e *Matplotlib* para ter uma melhor visualização e entendimento dos resultados.
E assim, cruzar informação das diferentes propriedades e determinar as características dos agrupamentos.

1) Calculei a distribuição dos dados fazendo uma contagem dos valores agrupados por cluster. Observei que existem valores bem diferentes em cada um deles.

```
clusters = prediction_inter.groupBy("prediction")\
                .count()\
                .orderBy("prediction")\

clusters.show()
```

```
+----------+-----+
|prediction|count|
+----------+-----+
|         0| 6796|
|         1| 2105|
|         2|  309|
+----------+-----+
```

2) Obtive a média de passageiros e desvio padrão em cada cluster, para conhecer como é a demanda dos voos dentro desses agrupamentos. Em conjunto com a tabela, fiz um gráfico do tipo *boxplot* para interpretar melhor os resultados (gráfico da apresentação). [14]

```
prediction_inter.groupBy("prediction")\
    .agg(f.round(f.avg("Adjusted Passenger Count"))\
         .alias("média passageiros"),
         f.round(f.stddev("Adjusted Passenger Count"),2)\
         .alias("desvio"))\
    .orderBy("prediction").show()
```

```
+----------+-----------------+--------+
|prediction|média passageiros|  desvio|
+----------+-----------------+--------+
|         0|           6463.0| 3032.69|
|         1|          18062.0| 4859.69|
|         2|          62122.0|12056.69|
+----------+-----------------+--------+
```

3) Contei a frequência mensal de voos dentro de cada cluster, agrupando por cluster e mês, para logo desenhar os resultados. Com os valores verifiquei quais são os meses com maior demanda dentro de cada agrupamento.

```
sazonal =prediction_inter.groupBy("prediction", "Month")\
            .agg(f.count("Month").alias("Frequencia"))\
            .orderBy("prediction", "Month")
sazonal.show(5)
```

```
+----------+-----+----------+
|prediction|Month|Frequencia|
+----------+-----+----------+
|         0|    1|       585|
|         0|    2|       626|
|         0|    3|       588|
|         0|    4|       529|
|         0|    5|       509|
+----------+-----+----------+
only showing top 5 rows
```

4) Por último, medi a quantidade de vezes que há conexões com as diferentes regiões dentro de cada agrupamento, para tirar a informação de quais são as rotas mais significativa em cada cluster.

```
regiao = prediction_inter.groupBy("prediction", "GEO Region")\
            .agg(f.count("GEO Region").alias("Contagem"))\
            .orderBy("prediction")

regiao.show(5)
```

prediction	GEO Region	Contagem
0	South America	90
0	Europe	1175
0	Canada	977
0	Mexico	970
0	Asia	2378

only showing top 5 rows

Com todas as informações obtidas do cruzamento das propriedades, e interpretação dos gráficos, estou em condições de tirar conclusões e definir padrões demonstrados pelos resultados do algoritmo.
As minhas conclusões estão expressas com detalhes na *Apresentação* deste projeto.

Bibliografia

[1] Informação de empresas sem código IATA
https://en.wikipedia.org/wiki/Servisair
https://es.wikipedia.org/wiki/Swissport
https://es.wikipedia.org/wiki/Boeing

[2] Referência comando *replace()* Pandas
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html

[3] Aplicação do comando *to_json()* da biblioteca Pandas
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html

[4] Informação sobre conexão de MongoDB com Python e *insert_many()*
https://www.mongodb.com/pt-br/docs/languages/python/pymongo-driver/current/get-started/
https://www.mongodb.com/pt-br/docs/languages/python/pymongo-driver/current/crud/insert/

[5] Quickstart de PySpark, criar um DF partindo do Pandas
https://spark.apache.org/docs/latest/api/python/getting_started/quickstart_df.html

[6] Referência comando *count_distinct* PySpark
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.functions.count_distinct.html?highlight=distinct#pyspark.sql.functions.count_distinct

[7] Comando *dropDuplicates()* PySpark
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.dropDuplicates.html

[8] Utilização de *plot()* da biblioteca Pandas
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html#pandas.DataFrame.plot

[9] Informação sobre *StringIndexerModel.labelsArray de* PySpark
https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StringIndexerModel.html

[10] Link para utilização *heatmap* da biblioteca Seaborn
https://seaborn.pydata.org/generated/seaborn.heatmap.html

[11] Treino modelo *BisectingKMeans*
https://spark.apache.org/docs/latest/ml-clustering.html#bisecting-k-means

[12] Métodos do modelo de agrupamento hierárquico, como o *clusterCenters()*
https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.clustering.BisectingKMeansModel.html

[13] Dendograma com biblioteca SciPy
https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html

[14] *Boxplot* e paleta de cores da biblioteca *Seaborn*
https://seaborn.pydata.org/generated/seaborn.boxplot.html
https://seaborn.pydata.org/tutorial/color_palettes.html

+ Blog com informação sobre visualização com Pandas e Matplotlib
https://medium.com/ironhack/data-visualization-con-pandas-y-matplotlib-7f1358be6764