

Pós-Graduação em Ciência de Dados e Analytics - PUC-Rio

MVP - Engenharia de Dados

Aluno: Fabiano Ramos (Setembro / 2023)

Definição do Problema

Motivação

Para este trabalho, utilizamos o conjunto de dados [Airline Dataset](#), disponibilizado a partir do [Kaggle](#).

O conjunto de dados é *artificial*, e oferece informações relacionados a operações aéreas, possibilitando a obtenção de *insights* interessantes sobre a demografia e comportamento dos passageiros. Possui um total de 98619 registros e 15 atributos.

Nossas perguntas-alvo foram as seguintes:

- Quais os aeroportos mais movimentados?
- Quais aeroportos possuem um maior número de atrasos e/ou cancelamentos?
- Qual a demografia dos passageiros?

O dicionário de dados é o seguinte:

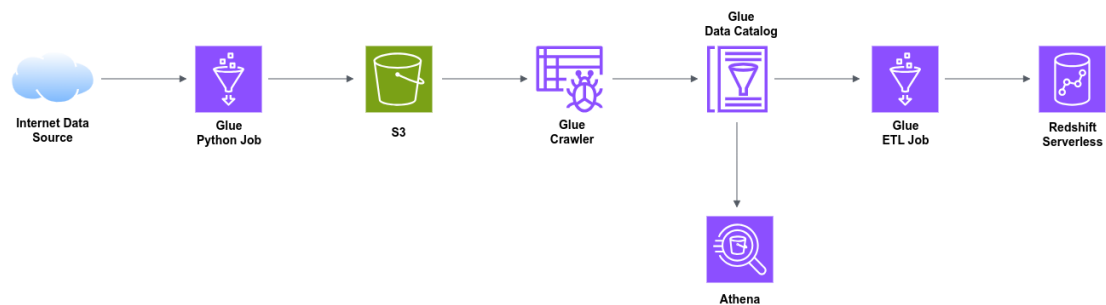
- **Passenger ID** - Identificação única do passageiro.
- **First Name** - Primeiro nome do passageiro.
- **Last Name** - Último nome do passageiro.
- **Gender** - Sexo do passageiro.
- **Age** - Idade do passageiro.
- **Nationality** - Nacionalidade do passageiro.
- **Airport Name** - Nome do aeroporto de partida.
- **Airport Country Code** - Código do país do aeroporto.
- **Country Name** - Nome do país do aeroporto.
- **Airport Continent** - Continente onde o aeroporto está situado.
- **Continents** - Continente envolvido na rota do aeroporto.
- **Departure Date** - Data de decolagem.
- **Arrival Airport** - Aeroporto de destino.
- **Pilot Name** - Nome do piloto.
- **Flight Status** - Situação do voo (cancelado, atrasado, no horário).

Stack de Tecnologia

Decidimos levar a cabo nosso trabalho utilizando a [Amazon Web Services \(AWS\)](#).

Como ferramenta de integração de dados, utilizamos o [AWS Glue](#), um *framework* sem servidor destinado a busca, preparação e integração de dados. Utilizamos também o componente [AWS Glue Data Catalog](#) para preparação do Catálogo de Dados correspondente, o [AWS Athena](#) para consultas diretas ao catálogo de dados (com dados ainda armazenados no S3) e também o [Amazon RedShift Serverless](#), um serviço de *data warehouse*, também sem servidor, a partir do qual realizamos as análises/consultas desejadas.

A figura abaixo mostra em linhas gerais nosso *pipeline* de dados:



Coleta dos Dados

Análise preliminar

Em primeiro lugar, realizamos uma inspeção do *dataset* para verificar sua qualidade e para apurar se operações de transformação se fariam necessárias.

Fizemos uma [cópia](#) do arquivo *csv* original em nosso github para que este trabalho se mantenha auto-contido e íntegro em caso de consulta futura.

```

In [ ]: # carga das bibliotecas
import pandas as pd

# desabilita warnings
import warnings
warnings.filterwarnings('ignore')

# carrega o dataset
dataset = pd.read_csv('https://raw.githubusercontent.com/pragmaerror/puc-

# dimensoes do dataset
print(dataset.shape)

# nulos?
print(dataset.isnull().sum())

# atributos categóricos estão corretos?
print(dataset['Gender'].unique())
print(dataset['Flight Status'].unique())
  
```

```
# ID é mesmo único?
assert(dataset['Passenger ID'].nunique() == dataset.shape[0])
```

```
(98619, 15)
Passenger ID      0
First Name        0
Last Name         0
Gender            0
Age              0
Nationality       0
Airport Name      0
Airport Country Code 0
Country Name      0
Airport Continent 0
Continents        0
Departure Date    0
Arrival Airport   0
Pilot Name        0
Flight Status     0
dtype: int64
['Female' 'Male']
['On Time' 'Delayed' 'Cancelled']
```

Como podemos observar, o conjunto de dados é bem comportado (inclusive os atributos categóricos *Flight Status* e *Gender*) e pode ser carregado diretamente sem necessidade de pré-processamento.

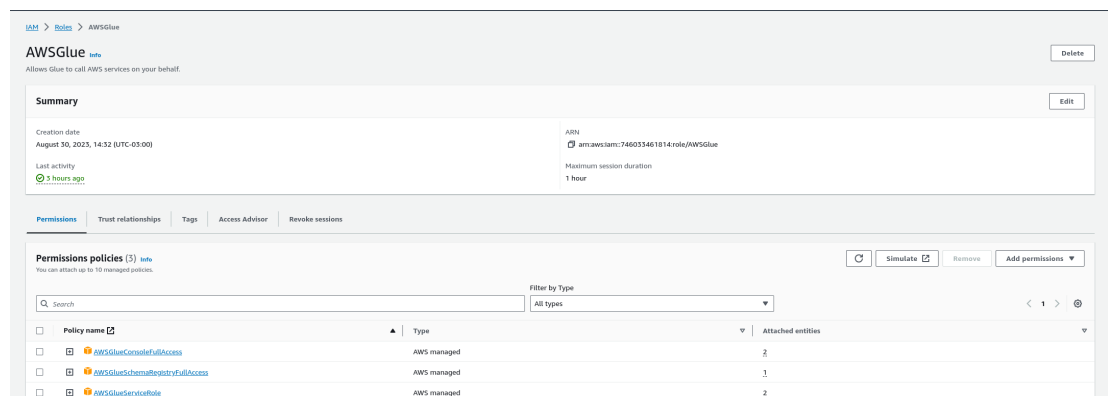
Carga dos Dados para a Nuvem

Para a coleta dos dados, criamos um script ETL python, a partir do AWS Glue.

O script é responsável por buscar os **dados** diretamente do nosso github e salvá-los dentro de um bucket S3 previamente criado.

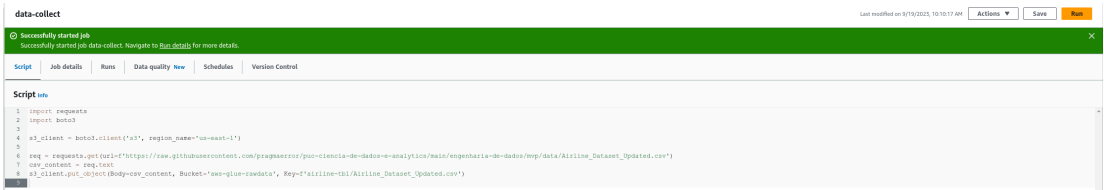
O passo a passo da implantação do script foi o seguinte:

1. Criação de um IAM Role para execução. Criamos um Role chamado *AWSGlue*, composto pelas políticas pré-definidas *AWSGlueServiceRole*, *AWSGlueSchemaRegistryFullAccess* e *AWSGlueConsoleFullAccess*.

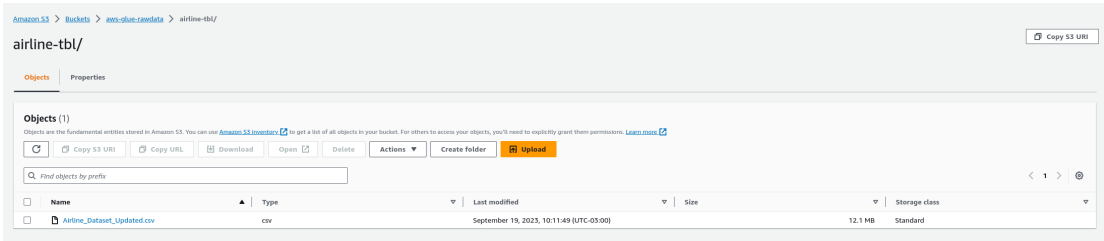


2. Criação do S3 bucket destino. Atribuímos o nome *aws-glue-rawdata*, uma vez que as permissões do *AWSGlueServiceRole* exigem que o nome do bucket possua *aws-glue-* como prefixo.

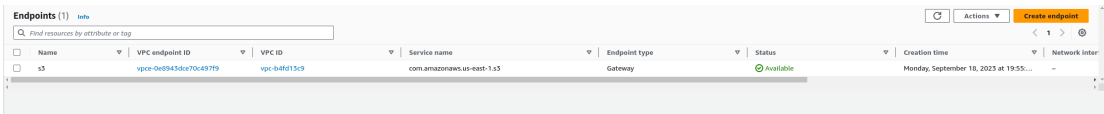
3. Criação do script python propriamente dito, que chamamos de *data-collect*, conforme mostramos a seguir. O [script](#) também pode ser consultado em nosso [github](#).



Através da execução do script, os dados de entrada foram então coletados e armazenados no S3 bucket indicado.



Finalmente, nas configurações de rede, criamos um *endpoint* para que o S3 pudesse ser acessado pelos jobs ETL Glue mais a frente, durante a carga dos dados para o *data warehouse*.

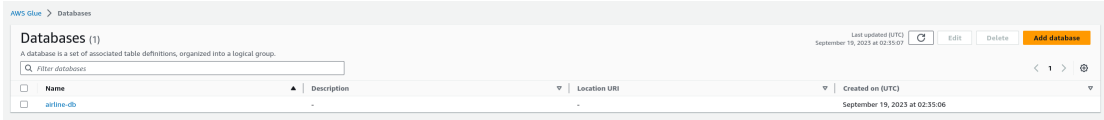


Modelagem

Decidimos então manter a representação dos dados em sua forma *flat*, como uma única tabela, como em um Data Lake.

Prosseguimos então para a definição do Catálogo de Dados associado, usando o [AWS Glue Data Catalog](#). Seguimos então os passos descritos a seguir:

1. Criação da base de dados *airline-db*.



2. Criação da *tabela*, a partir de um *crawler*, com as seguintes configurações:

- Nome do *crawler*: *airline-tbl-crawler*
- Fonte de dados: *s3://aws-glue-rawdata/airline-tbl/*
- IAM role: *AWSGlue*
- Base de dados: *airline-db*

Step 1

Set crawler properties

Step 2

Choose data sources and classifiers

Step 3

Configure security settings

Step 4

Set output and scheduling

Step 5

Review and create

Review and create

Step 1: Set crawler properties

Set crawler properties

Name

airline-tbl-crawler

Description

-

Tags

-

Step 2: Choose data sources and classifiers

Data sources (1)

The list of data sources to be scanned by the crawler

Type

S3

Data source

s3://aws-glue-rawdata/airline-tbl/

Parameters

Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role

AWSGlue

Security configuration

-

Lake Formation configuration

-

Step 4: Set output and scheduling

Set output and scheduling

Database

airline-db

Table prefix - optional

-

Maximum table threshold - optional

-

Schedule

On demand

Cancel

Previous

Create crawler

Ao executar o *crawler* a tabela foi criada e seu esquema automaticamente detectado, conforme figura abaixo. O nome da tabela foi automaticamente gerado a partir do nome da pasta S3 onde o csv estava armazenado, *airline_tbl* (note que o '-' foi substituído por '_').

Tables

A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Tables (1)

View and manage all available tables.

Filter tables

Last updated (UTC)

September 19, 2023 at 13:20:52

Delete

Add tables using crawler

Add table

Name

Database

Location

Classification

Deprecated

View data

Data quality

airline_tbl

airline-db

s3://aws-glue-rawdata/airline-tbl/

CSV

-

Table data

View data quality

Schema

Partitions

Indexes

Schema (15)

View and manage the table schema.

Filter schemas

#

Column name

Data type

Partition key

Comment

1

passenger_id

string

-

-

2

first_name

string

-

-

3

last_name

string

-

-

4

gender

string

-

-

5

age

bigint

-

-

6

nationality

string

-

-

7

airport_name

string

-

-

8

airport_country_code

string

-

-

9

country_name

string

-

-

10

airport_continent

string

-

-

11

continents

string

-

-

12

departure_date

string

-

-

13

arrival_airport

string

-

-

14

pilot_name

string

-

-

15

flight_status

string

-

-

Todos os campos definidos no csv foram corretamente detectados. A seguir, através da opção *Edit schema as JSON*, editamos e ajustamos manualmente o esquema da tabela (nomes de colunas, tipos de dados e comentários), conforme mostramos abaixo. O arquivo [JSON](#) completo pode ser encontrado em nosso github. O ajuste do nome das colunas (remoção de espaços) foi fundamental para que a carga para o *data warehouse* (realizada nos passos seguintes) funcionasse corretamente nos passos seguintes.

Schema

Partitions

Indexes

Schema (15)

View and manage the table schema.

Filter schemas

#

Column name

Data type

Partition key

Comment

1

passenger_id

string

-

Identificação única do passageiro.

2

first_name

string

-

Primeiro nome do passageiro.

3

last_name

string

-

Último nome do passageiro.

4

gender

string

-

Sexo do passageiro (Female, Male).

5

age

int

-

Idade do passageiro.

6

nationality

string

-

Nacionalidade do passageiro.

7

airport_name

string

-

Nome do aeroporto de partida.

8

airport_country_code

string

-

Código do país do aeroporto.

9

country_name

string

-

Nome do país do aeroporto.

10

airport_continent

string

-

Continente onde o aeroporto está situado.

11

continents

string

-

Continente envolvido na rota do aeroporto.

12

departure_date

string

-

Data de decolagem.

13

arrival_airport

string

-

Aeroporto de destino.

14

pilot_name

string

-

Nome do piloto.

15

flight_status

string

-

Situação do voo (On Time, Delayed, Cancelled).

Um vez disponibilizado o catálogo de dados, pode-se efetuar consultas diretamente sobre os dados persistidos no S3 utilizando o componente [AWS Athena](#). Neste momento realizamos então apenas uma consulta básica com o objetivo de verificar que o catálogo de dados foi criado corretamente.

localhost:8888/lab/tree/mvp_fabiano_ramos_3.ipynb

5/12

Para a correta utilização do Athena, um bucket s3 precisou ser configurado para armazenar os resultados das consultas. Criamos então um bucket chamado *aws-glue-athena-workspace* para tal propósito.

A seguir, realizamos uma consulta simples, para verificar a integridade do catálogo de dados:

Amazon Athena > Query editor

EditorRecent queriesSaved queriesSettings

primary

Data

Test Query | XQuery 12 | XQuery 13 | XQuery 14 | X

SELECT * FROM "AwsDataCatalog"."athena-tpc"."athena_test" LIMIT 5;

SQLLn 1, Col 66

Run againExplainCancelClearCreate

Run query results up to 60 minutes ago

Tables and views

Create

Filter tables and views

Tables (1)

athena_ml

Views (0)

Query results

Query status

Completed

Time to query: 183 msRun time: 578 msData scanned: 305.79 KB

Results (5)

CopyDownload results

#	passenger_id	first_name	last_name	gender	age	nationality	airport_name	airport_country_code	country_name	airport_latitude	continent	departure_date	arrival_airport	pilot_name	flight_status
1	ABW8g	Edith	Leggin	Female	62	Japan	Goldfish Airport	US	United States	NAH	North America	6/28/2022	CSF	Edith Leggin	On Time
2	JKXXX	Eliwood	Catt	Male	62	Nicaragua	Kuglakak Airport	CA	Canada	NAH	North America	12/26/2022	YCO	Eliwood Catt	On Time
3	CB22g	Darby	Felgate	Male	67	Russia	Grenoble-lake Airport	FR	France	EU	Europe	1/18/2022	GNB	Darby Felgate	On Time
4	BRS3BV	Dominica	Pyle	Female	71	China	Ottawa / Gatineau Airport	CA	Canada	NAH	North America	9/16/2022	VND	Dominica Pyle	Delayed
5	Nv7Lo	Ray	Perenot	Male	21	China	Gilegia Field	US	United States	NAH	North America	2/25/2022	SEE	Ray Perenot	On Time

Carga dos Dados no Data Warehouse

O passo seguinte foi então carregar os dados no [AWS Redshift Serverless](#).

Realizamos então a configuração inicial do serviço. Criamos um IAM Role apropriado contendo a política *AmazonRedshiftAllCommandsFullAccess*.

Amazon Redshift Serverless > Workgroup configuration > Create workgroup

Step 1Create workgroup

Step 2Choose namespace

Step 3Review and Create

Review and Create

Step 1: Create workgroupEdit

Workgroup

Workgroup is a collection of compute resources from which an endpoint is created. Compute properties include network and security settings.

Workgroup name

mvp-workgroup

Capacity

Set the base capacity used to process your data warehouse workloads. The capacity is measured in Redshift processing units (RPUs). To improve query performance, increase the RPU value.

Base capacity

32

Network and security

Virtual private cloud (VPC)

vpc-b4fd13c9

VPC security group

sg-42859f64

Subnet

subnet-89ba2e87, subnet-1259bf23, subnet-1a538a3b, subnet-7156892e, subnet-4fbc6c29, subnet-63b9db2e,

Step 2: Choose namespace Edit

Namespace
Namespace is a collection of database objects and users. Data properties include database name and password, permissions, and encryption and security.

Target namespace
mvp-namespace

Database name and password

Database name
dev

Admin user credentials
root

Permissions

Default IAM role
arn:aws:iam::746033461814:role/service-role/AmazonRedshift-CommandsAccessRole-20230918T154901

IAM roles
arn:aws:iam::746033461814:role/service-role/AmazonRedshift-CommandsAccessRole-20230918T154901

Encryption and security

AWS KMS encryption
AWS owned key

Audit logging
Off

Cancel Previous Create

Neste momento, voltamos ao Glue Studio para a criação de um job ETL tendo como fonte de dados o Catálogo de Dados previamente criado e, como destino, o Redshift. Como pré-requisito, foi necessário configurar uma conexão com o RedShift, conforme mostrado abaixo.

[AWS Glue](#) > [Connectors](#) > redshift-connection

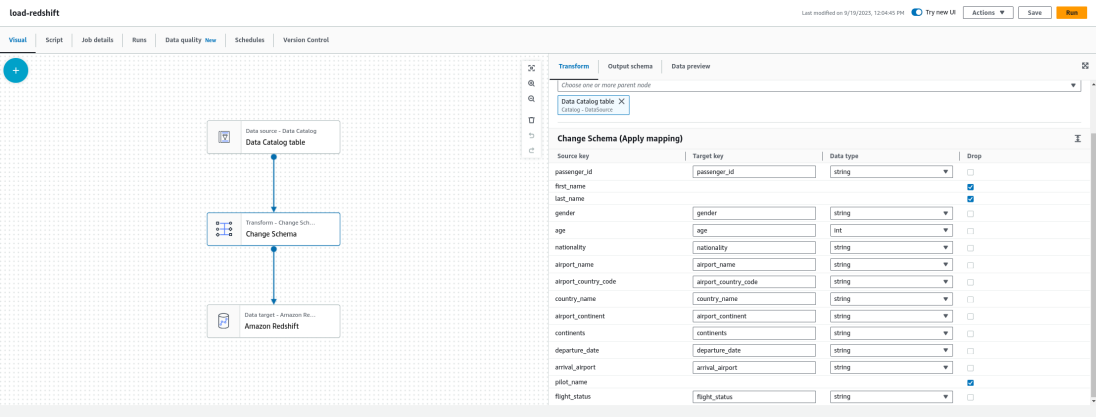
redshift-connection Edit Delete Create job

Connection details Info

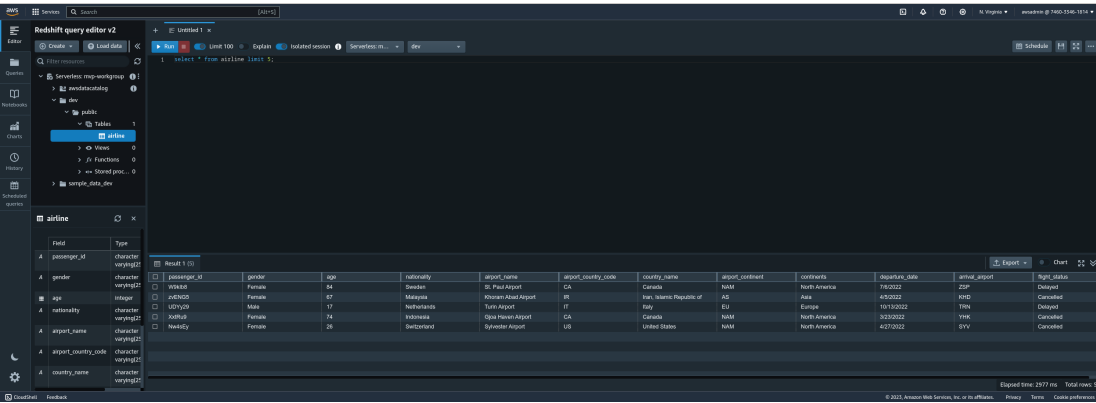
Connector type JDBC	Connection URL jdbc:redshift://mvp-workgroup.746033461814.us-east-1.redshift-serverless.amazonaws.com:5439/dev
Driver class name -	Driver path -
Username root	Require SSL connection false
Subnet subnet-89ba2e87	Security groups sg-42859f64
Description -	Created on 2023-09-19 11:41:02.023000
Last modified 2023-09-19 11:41:02.023000	Class name -

Uma vez criada a conexão, o job ETL pôde então ser criado, conforme mostrado abaixo. Criamos o job visualmente, mas o [fonte completo](#) do script pode ser observado em nosso github. Não aplicamos nenhuma transformação de tipos de dados no esquema,

mas descartamos os campos *First Name*, *Last Name* e *Pilot Name* que são irrelevantes ao problema.



Após a carga, realizamos então uma consulta de testes a partir do console do RedShift, no sentido de validar a carga:



Análise dos Dados

Qualidade dos Dados

Já havíamos realizado a inspeção do *dataset* durante a fase preliminar e não encontramos problemas estruturais.

Notamos, no entanto, que as informações sobre os aeroportos de chegada (código) não são compatíveis (em nível de detalhe) com as dos aeroportos de partida (nome/país), o que limita bastante as oportunidades de análise.

Solução das Questões Propostas

Quais os aeroportos mais movimentados?

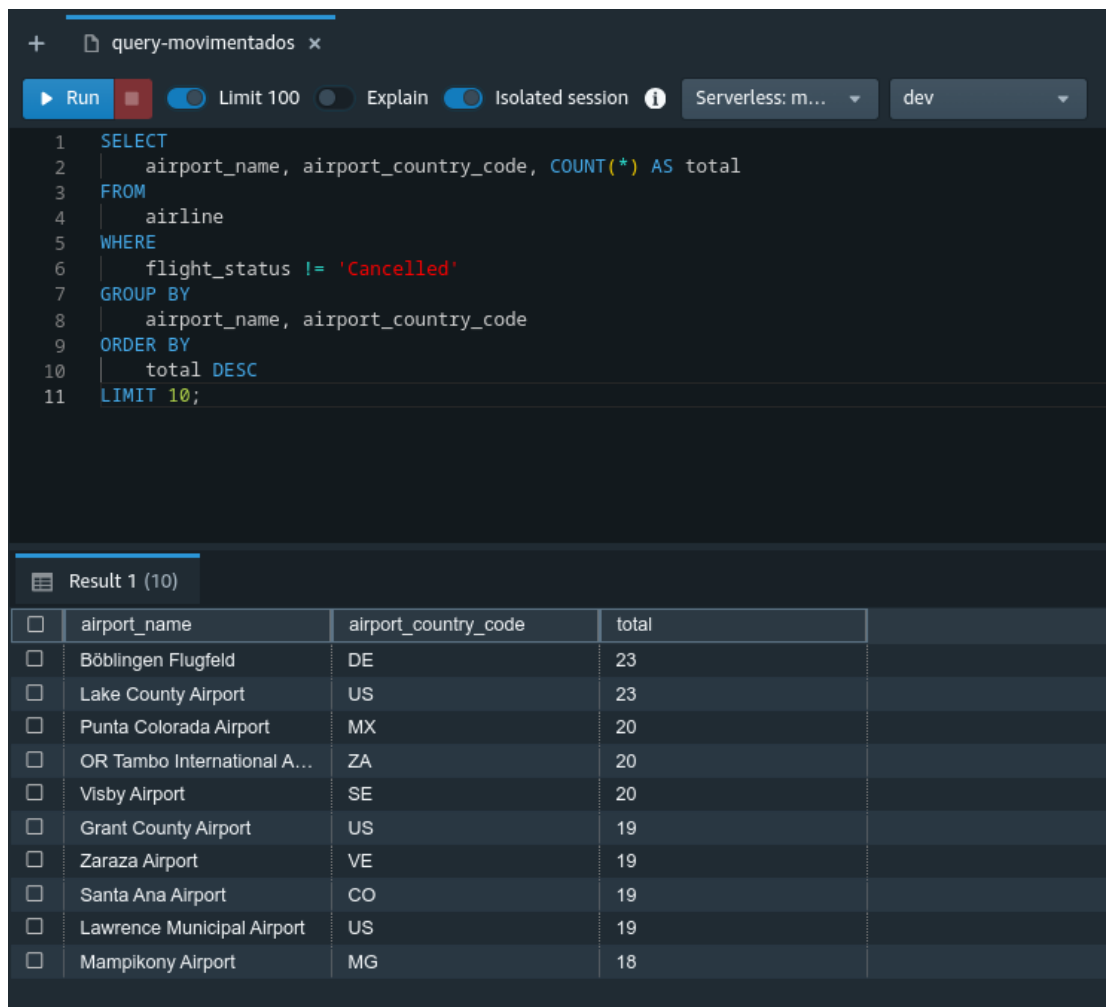
A primeira pergunta que tentamos responder foi determinar os aeroportos mais movimentados.

Realizamos então uma consulta relativamente simples, com o totalizador e deixando de fora os vôos cancelados. Limitamos os resultados a apenas os 10 aeroportos mais

movimentados.

A consulta e sua resposta são mostrados na figura abaixo.

```
In [ ]: SELECT
        airport_name, airport_country_code, COUNT(*) AS total
FROM
        airline
WHERE
        flight_status != 'Cancelled'
GROUP BY
        airport_name, airport_country_code
ORDER BY
        total DESC
LIMIT 10;
```



The screenshot shows a SQL query editor with a dark theme. The query is the same as the one in the previous block. Below the query editor, there is a tab labeled "Result 1 (10)" which displays a table with 10 rows. Each row has a checkbox in the first column, followed by the airport name, country code, and total count. The results are ordered by total count in descending order.

<input type="checkbox"/>	airport_name	airport_country_code	total
<input type="checkbox"/>	Böblingen Flugfeld	DE	23
<input type="checkbox"/>	Lake County Airport	US	23
<input type="checkbox"/>	Punta Colorada Airport	MX	20
<input type="checkbox"/>	OR Tambo International A...	ZA	20
<input type="checkbox"/>	Visby Airport	SE	20
<input type="checkbox"/>	Grant County Airport	US	19
<input type="checkbox"/>	Zaraza Airport	VE	19
<input type="checkbox"/>	Santa Ana Airport	CO	19
<input type="checkbox"/>	Lawrence Municipal Airport	US	19
<input type="checkbox"/>	Mampikony Airport	MG	18

Quais aeroportos possuem um maior número de atrasos e/ou cancelamentos?

A segunda pergunta que tentamos responder foi determinar os aeroportos mais afetados por atrasos e/ou cancelamentos.

Um fator complicador para a consulta foi que tentamos arbitrariamente filtrar apenas os aeroportos com 20 ou mais vôos, já que este tipo de consulta estatística faz pouco sentido para aeroportos muito pequenos. Este pequeno detalhe fez com que

precisássemos realizar uma subconsulta, uma vez que a cláusula WHERE não poderia ser inserida na primeira *query*.

Mais uma vez, limitamos os resultados a apenas os 10 aeroportos mais afetados.

A consulta e sua resposta são mostrados na figura abaixo.

```
In [ ]: WITH flight_problems AS (
        SELECT
            airport_name, airport_country_code,
            SUM(CASE WHEN flight_status = 'Cancelled' OR flight_status = 'Delayed' THEN 1 ELSE 0 END) AS problems,
            COUNT(*) AS total, problems*100.0/total AS percentual
        FROM
            airline
        GROUP BY
            airport_name, airport_country_code
    )
    SELECT
        airport_name, airport_country_code, problems, total, percentual
    FROM
        flight_problems
    WHERE
        total > 20
    ORDER BY
        percentual DESC
    LIMIT 10;
```

query-percentual x

Run Limit 100 Explain Isolated session Serverless: m... dev

```

1 WITH flight_problems AS (
2     SELECT
3         airport_name, airport_country_code,
4         SUM(CASE WHEN flight_status = 'Cancelled' OR flight_status = 'Delayed' THEN 1 ELSE 0 END) AS problems,
5         COUNT(*) AS total, problems*100.0/total AS percentual
6     FROM
7         airline
8     GROUP BY
9         airport_name, airport_country_code
10 )
11 SELECT
12     airport_name, airport_country_code, problems, total, percentual
13 FROM
14     flight_problems
15 WHERE
16     total > 20
17 ORDER BY
18     percentual DESC
19 LIMIT 10;
```

Result 1 (10)

	airport_name	airport_country_code	problems	total	percentual
<input type="checkbox"/>	Baudette International Air...	US	18	21	85.714285714285714
<input type="checkbox"/>	Amchitka Army Airfield	US	21	25	84
<input type="checkbox"/>	Horn Island Airport	AU	18	22	81.818181818181818
<input type="checkbox"/>	Yangzhou Taizhou Airport	CN	22	27	81.481481481481481
<input type="checkbox"/>	Marion Municipal Airport	US	22	27	81.481481481481481
<input type="checkbox"/>	Bradshaw Army Airfield	US	17	21	80.952380952380952
<input type="checkbox"/>	Dalnerechensk Airport	RU	17	21	80.952380952380952
<input type="checkbox"/>	Capital City Airport	US	22	28	78.571428571428571
<input type="checkbox"/>	Lonely Air Station	US	18	23	78.260869565217391
<input type="checkbox"/>	Frederick Regional Airport	US	16	21	76.19047619047619

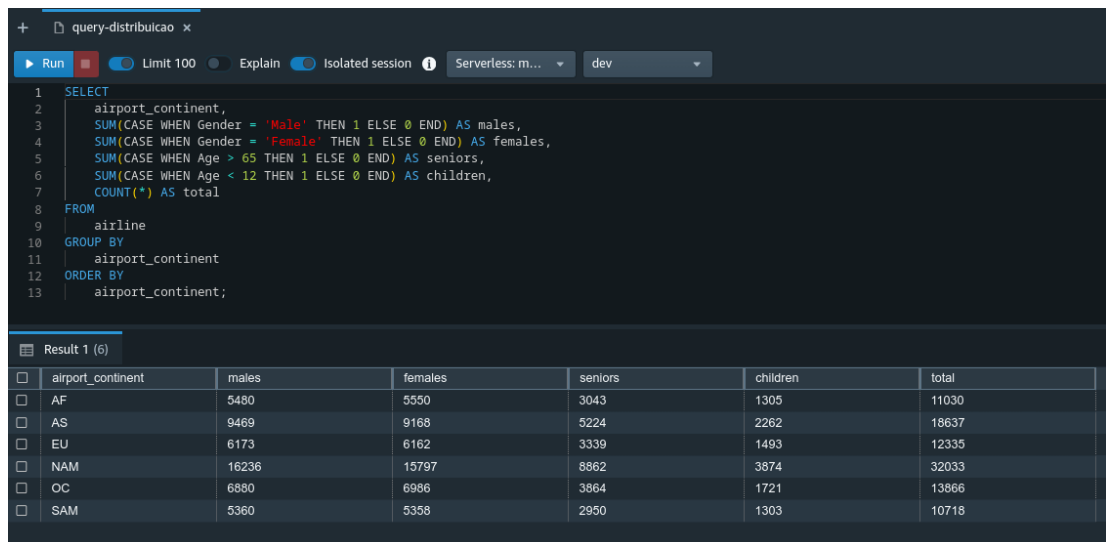
Qual a demografia dos passageiros?

Para a terceira pergunta, tentamos entender a demografia dos passageiros.

Idosos e crianças trazem requisitos diferenciados às empresas aéreas (preço e acomodações) então consideramos esta uma consulta relevante. Incluímos também o sexo dos passageiros, e decidimos agrupar por continente para uma melhor representatividade.

A consulta e sua resposta são mostrados na figura abaixo.

```
In [ ]: SELECT
        airport_continent,
        SUM(CASE WHEN Gender = 'Male' THEN 1 ELSE 0 END) AS males,
        SUM(CASE WHEN Gender = 'Female' THEN 1 ELSE 0 END) AS females,
        SUM(CASE WHEN Age > 65 THEN 1 ELSE 0 END) AS seniors,
        SUM(CASE WHEN Age < 12 THEN 1 ELSE 0 END) AS children,
        COUNT(*) AS total
FROM
    airline
GROUP BY
    airport_continent
ORDER BY
    airport_continent;
```



The screenshot shows a Jupyter Notebook interface with a SQL query editor and a results table. The query is a SELECT statement with columns for airport_continent, males, females, seniors, children, and total. The results are shown in a table with 6 rows and 6 columns.

airport_continent	males	females	seniors	children	total
AF	5480	5550	3043	1305	11030
AS	9469	9168	5224	2262	18637
EU	6173	6162	3339	1493	12335
NAM	16236	15797	8862	3874	32033
OC	6880	6986	3864	1721	13866
SAM	5360	5358	2950	1303	10718

Conclusão & Auto Avaliação

As atividades aqui descritas correspondem ao nosso primeiro contato com a atividade de ETL (e Engenharia de Dados de uma forma mais ampla).

Nosso foco pessoal foi o aprendizado da plataforma AWS, e desta forma tentamos manter as demais variáveis (*dataset* e escopo do trabalho) o mais controladas possível dado o curto prazo de confecção desde projeto. Já tínhamos alguma familiaridade com o GCP (embora em outro contexto) e nos desafiamos a aprender a AWS, o que foi bastante recompensador.

Neste trabalho, fomos capazes de produzir o *pipeline* desejado, desde a análise preliminar dos dados, passando pela sua (simples) modelagem, produção do catálogo de dados correspondente até a sua carga no *data warehouse*, a partir de onde realizamos as consultas desejadas.

Como possibilidades de continuação deste trabalho, podemos destacar em primeiro lugar melhorar a reprodutibilidade da construção do *pipeline*, uma vez que realizamos muitos passos manuais. Entendemos que técnicas de IaC (*infrastructure as code*) seriam indispensáveis em um ambiente de produção. Mesmo dentro do AWS Glue, identificamos a funcionalidade de *workflows*, que supostamente permitem a definição do fluxo de trabalho através de *blueprints*. Seria bem interessante explorar este caminho no futuro.

Outro desenvolvimento que gostaríamos de realizar no futuro é a interação programática (via *python*) com o AWS Glue e com o Redshift.

Além disso, o uso de dados reais como por exemplo os [dados abertos](#) da ANAC possibilitariam mais oportunidades de modelagem, transformação e análise. Chegamos examinar estes dados, mas eles apresentaram demasiados problemas e inexatidões de forma que preferimos evitá-los neste momento, por razões já apresentadas.

Finalmente, de forma geral nossa avaliação desta atividade foi bastante positiva. O escopo bastante amplo, ao mesmo tempo que foi um ponto de complexidade adicional, permitiu aos estudantes direcionar suas implementações de acordo com seus interesses e competências individuais.