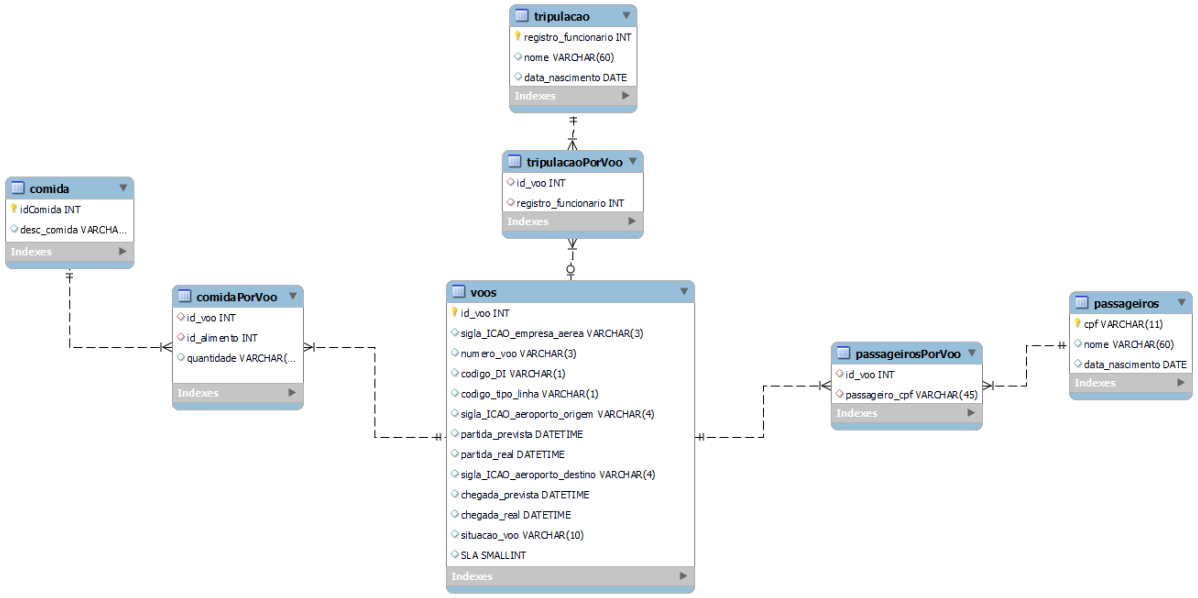
Data Warehouse - Projeto final de Mineração de Dados

Alunos: Felipe Martins Machado Mendes e Caio Righetto Campos

Contextualização:

Somos analistas de dados trabalhando para uma grande empresa de transporte aéreo. A companhia aérea tem vários sistemas para gerenciar as **reservas de voos, manutenção de aeronaves, operações de aeroportos, catering, pessoal e finanças.** A empresa aérea deseja unificar os dados desses sistemas díspares em um único Data Warehouse para melhorar a eficiência das operações, maximizar a lucratividade, aprimorar a experiência do cliente e informar decisões estratégicas.

Ao ouvir mais sobre os objetivos e intenções da empresa com a implementação deste datawarehouse, a solução inicial que pensamos foi a seguinte:



Temos 7 tabelas no total, sendo elas:

Tabela voos:

fonte:
https://www.gov.br/anac/pt-br/assuntos/dados-e-estatisticas/historico-de-voos

id_voo: Número identificador único para cada voo, utilizado como chave primária da tabela.

sigla_ICAO_empresa_aerea: Sigla de 3 letras representando a identificação ICAO da

empresa aérea associada ao voo.

numero_voo: Número de identificação do voo, geralmente composto por 3 letras ou números.

codigo_DI: Código do dígito identificador do voo;

codigo_tipo_linha: Código de um caractere que pode indicar o tipo de linha do voo.

sigla_ICAO_aeroporto_origem: Sigla de 4 letras representando a identificação ICAO do aeroporto de origem do voo.

partida_prevista: Data e hora previstas para a partida do voo.

partida_real: Data e hora real da partida do voo.

sigla_ICAO_aeroporto_destino: Sigla de 4 letras representando a identificação ICAO do aeroporto de destino do voo.

chegada_prevista: Data e hora previstas para a chegada do voo ao destino.

chegada_real: Data e hora real da chegada do voo ao destino.

situacao_voo: Indica a situação do voo.

SLA: Indica o atraso (se houver) para a partida do avião.

Tabela comida:

idComida: Número identificador do alimento.

desc_comida: Texto descrevendo o alimento.

Tabela comidaPorVoo:

id_voo: código de identificação do voo.

id_comida: código de identificação do alimento.

quantidade: número de itens do alimento sendo levado.

Tabela tripulação:

registro_funcionario: código identificador do funcionário.

nome: nome do funcionário.

data_nascimento: data de nascimento do funcionário.

Tabela tripulaçãoPorVoo:

id_voo: código identificador do voo.

registro_funcionario: código identificador do funcionário.

Tabela passageiros:

cpf: cadastro de pessoa física do passageiro, é o identificador de cada um.

nome: nome do passageiro.

data_nascimento: data de nascimento do passageiro.

Tabela passageirosPorVoo:

id_voo: código identificador do voo.

passageiro_cpf: cpf do passageiro no voo.

Conforme foi dito, a **tabela voos** é proveniente dos dados abertos fornecidos pela ANAC (Agência Nacional de Aviação) e contém alguns detalhes que precisam ser tratados.

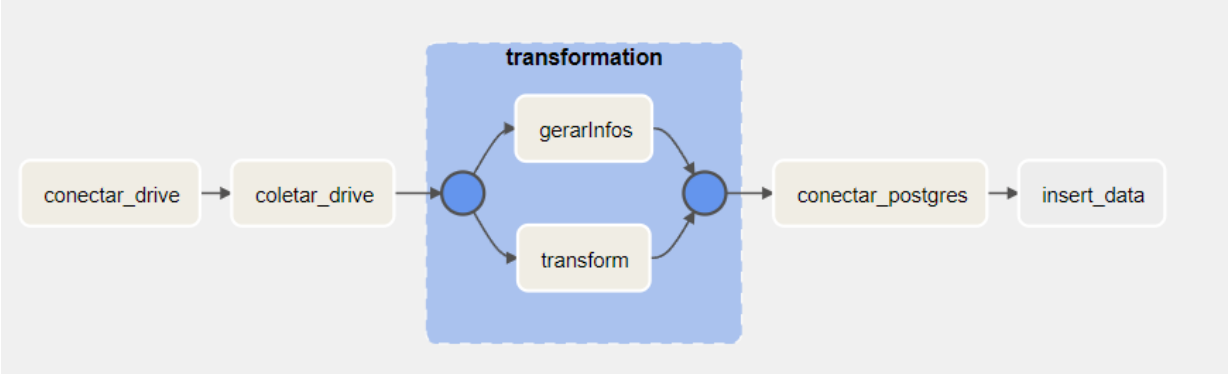
	Sigla ICAO Empresa Aérea	Número Voo	Código DI	Código Tipo Linha	Sigla ICAO Aeroporto Origem	Partida Prevista	Partida Real	Sigla ICAO Aeroporto Destino	Chegada Prevista	Chegada Real	Situação Voo
0	AAL	904	0	I	SBGL	01/10/2023 23:00	01/10/2023 22:49	KMIA	02/10/2023 07:42	02/10/2023 06:47	REALIZADO
1	AAL	905	0	I	KMIA	01/10/2023 23:52	01/10/2023 23:58	SBGL	02/10/2023 08:05	02/10/2023 08:02	REALIZADO

As colunas **Partida Prevista, Partida Real, Chegada Prevista e Chegada Real**

estão preenchidas com datas e horas, o problema é que elas estão como string, se quisermos fazer cálculos envolvendo estas datas, **precisamos converter elas de string para date**.

Além disso, criamos uma coluna adicional chamada **SLA,** que checa se a partida real foi mais tardia do que a partida prevista e se sim, contabiliza o tempo de atraso (será útil no dashboard).

Com essas transformações e o MER em mente, para fazer o processo de **ETL** resolvemos utilizar o **Apache Airflow** e desenvolvemos a seguinte **DAG**:



A dag foi dividida em 6 tasks no total, comentadas abaixo:

conectar_drive: Essa task é responsável por acessar o repositório onde estão os dados brutos.

coletar_drive: Essa task é responsável por coletar os dados brutos.

gerarInfos: Essa task gera as informações para completar as outras tabelas além da que coletamos.

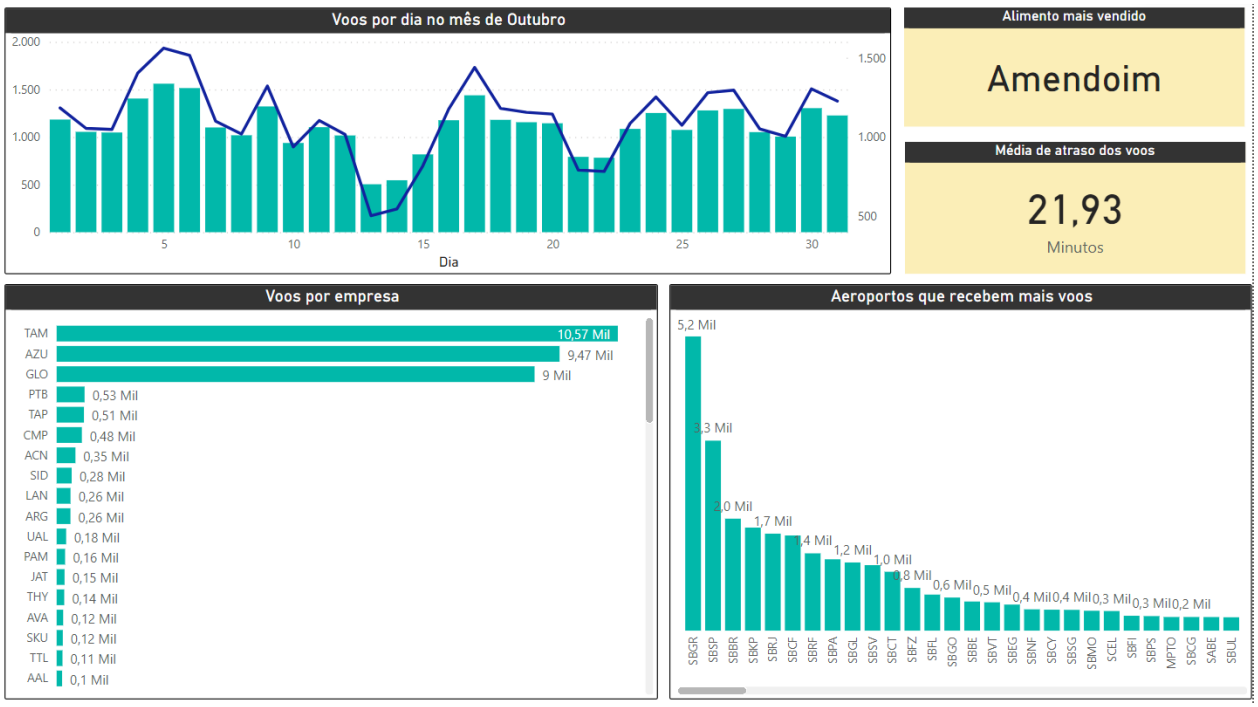
transform: Essa task faz as transformações comentadas acima, criando também a coluna de SLA.

conectar_postgres: Essa task conecta ao bando de dados.

insert_data: Essa task insere os dados nas tabelas.

Com dos dados no banco de dados, agora eles podem ser extraídos e utilizados para várias finalidades, seja para cálculos estatísticos, modelos de machine learning ou, no nosso caso, criação de dashboards.

Fizemos o seguinte dashboard no **Power BI:**



Acreditamos que ele representa parte dos interesses da empresa e se alinha com alguns dos objetivos para alcançar mais lucro e mais performance nas operações da companhia aérea.

Para implementarmos tudo isso em larga escala, seria necessário um local específico para processar os dados, como um **datacenter** ou se a empresa optar por um serviço de **nuvem**.

Com o poder de processamento em mãos, a empresa seria capaz de coletar os dados em tempo real e já encaminha-los para o começo do processo de **ETL.** Fazendo com que assim o dashboard seja atualizado constantemente e decisões estratégicas sejam tomadas.