Projeto Engenheiro de dados Equal BI

Por: Paulo Eduardo Dias

Documentação do Projeto

Parte 1: Processos, Engenharia de Dados e Infraestrutura

Parte 2: Análises Visuais, Relatórios de Qualidade de Dados

Resumo

1. Processos
   1. Pipeline de dados Base SQL
   2. Pipeline de dados Link CSV
2. Engenharia de Dados
   1. Extração e Leitura dos dados Base SQL
   2. Extração e Leitura dos dados Link CSV
   3. Transformação das Bases de dados
   4. Carregamento dos dados no Data Warehouse
3. Infraestrutura
   1. Ambiente utilizado
   2. Modelagem de dados para Power BI
4. Análises Visuais
   1. Análises e Relatórios da tabela Fact\_order\_datails
   2. Análises e Relatórios da tabela Fact\_orders
5. Relatórios de Qualidade de Dados
   1. Relatório de Completude
   2. Relatório de Unicidade
   3. Relatório de Resumo

Final

Resumo

A Equal BI foi contratada por um cliente para desenvolver uma solução de Business Intelligence que integrará dados de diversas fontes, processará esses dados e os disponibilizará para análise em Power BI. O cliente possui dados armazenados em um banco de dados SQL e em arquivos CSV, que são atualizados diariamente.

O objetivo desse projeto é extrair dados atualizados diariamente de um banco de dados SQL e 1 arquivo CSV, realizar as transformações necessárias dos dados, carregar os dados em um data Warehouse, preparar os dados e otimizá-los para a análise no Power BI, implementar relatórios de monitoramento e qualidade de dados.

Parte 1: Processos, Engenharia de Dados e Infraestrutura

1 Processos

A parte do processo detalha as pipelines de cada processo proposto nesse projeto, sendo as pipelines dos dados no banco de dados e no csv.

* 1. Pipeline de dados Base SQL

Recebemos um arquivo de insert para uma base de dados PostgreSQL, nome do arquivo northwind.sql, com esse arquivo usamos a base de dados gratuita ElephantSQL que é um cloud PostgreSQL para armazenar os dados do arquivo. Após os dados armazenados em um banco, criamos uma conexão com essa base de dados através da URL do PostgreSQL em um script Python usando a biblioteca psycopg2, após a conexão criada, usamos ela para fazer a leitura dos dados do banco e transformamos as tabelas em dataFrames usando a biblioteca pandas no Python, realizamos os tratamentos necessários para higienização dos dados, e após transformados, carregamos esses dados de forma incremental com upsert no data Warehouse que criamos no PostgresSQL, arquivo de criação do data Warehouse data\_warehouse.sql, após o upsert dos dados no data Warehouse é gerado os arquivos de monitoramento e qualidade dos dados, as tabelas do banco de dados foram separadas em tabelas fatos e dimensões para ficar apropriado para a leitura do Power BI.

1.1 Pipeline de dados Link SQL

Recebemos um link de um arquivo CSV, nome do arquivo order\_details.csv, criamos uma conexão através do link desse arquivo para fazer a extração diária com um script Python usando a biblioteca gdown, após a conexão criada, usamos ela para fazer a leitura dos dados do csv e transformamos o arquivo em dataFrames usando a biblioteca pandas no Python, realizamos os tratamentos necessários para higienização dos dados, e após transformados, carregamos esses dados de forma incremental com upsert no data Warehouse que criamos no PostgresSQL, arquivo de criação do data Warehouse data\_warehouse.sql, após o upsert dos dados no data Warehouse é gerado os arquivos de monitoramento e qualidade dos dados, foi gerada uma tabela através desse csv e tomamos a decisão de usar ela como uma tabela fato para a leitura do Power BI.

2 Engenharia de Dados

A parte da engenharia detalha melhor cada etapa do processo de forma a compreender melhor cada parte que foi usada no desenvolvimento da solução e mostra com clareza a forma de otimizar cada processo.

2.1 Extração e Leitura dos dados Base SQL

A extração e Leitura dos dados da Base SQL foi feita através de uma Função, essa função esta criada no arquivo extracao\_dados.py disponível no zip da documentação, nesse arquivo temos algumas funções criadas para ajudar na otimização do processo, a função responsável pela Extração e Leitura dos dados chama-se “extract\_postgres\_data“, nela passamos uma variável chamada “table\_name” que vai ser o nome da tabela que esta no banco de dados, a função segue essa sequência, ela armazena a URL de conexão com o postgress numa variável, usa o psycopg2 com a URL para fazer a conexão com o banco de dados, cria uma query com o nome da tabela mencionado acima, e por fim usa o pandas para transformar a query em um dataframe através dessa consulta, assim temos a leitura das tabelas do SQL diretamente no dataframe pandas automatizadas diariamente.

2.2 Extração e Leitura dos dados Link CSV

A extração e Leitura dos dados do Link CSV foi feita através de uma Função, essa função esta criada no arquivo extracao\_dados.py disponível no zip da documentação, nesse arquivo temos algumas funções criadas para ajudar na otimização do processo, a função responsável pela Extração e Leitura dos dados chama-se “extract\_csv\_data\_from\_drive”, nela passamos uma variável chamada “drive\_url” que é o link fornecido para download do arquivo, a função segue essa sequência, obtemos o id do arquivo através da URL usando a variável file\_id, criamos uma URL de download após obter o id do arquivo com a variável download\_url, criamos um caminho temporário para podermos armazenar o arquivo após fazer o download e armazenamos esse caminho na variável output, fazemos o download do arquivo usando a biblioteca gdown passando as variáveis download\_url, output e quiet=False para ver o andamento do download, com o download concluído criamos um dataframe pandas usando o read\_csv com a variável output que é onde o arquivo csv ficou armazenado, assim temos a leitura do link csv de forma automatizada diariamente.

2.3 Transformação das Bases de Dados

A transformação dos dados foi feita através de uma função para todos os dataframes criados, sendo SQL ou csv, essa função esta criada no arquivo extracao\_dados.py disponível no zip da documentação, a função responsável pelo tratamento dos dados chama-se “transform\_data”, nela passamos duas variáveis, “df” e “date\_column\_names”, “df” se refere ao dataframe criado pelo pandas, e “date\_column\_names” se refere a uma ou mais colunas de datas que possa ter no dataframe pandas, o tratamento de dados realizados aqui são os seguintes, remoção de dados duplicados com o comando “drop\_duplicates()”, substituir valores nulos por “0” com o comando fillna(0, inplace=True), usamos o “inplace” para otimização do processamento e da memória em disco e por fim transformamos os campos de data em data com o comando to\_datetime(), passamos os valores do “date\_column\_names” em um loop for e no final retornamos o dataframe tratado com essas 3 transformações, esses tratamentos são feitos para todos os dataframes existente, após esses tratamentos temos um tratamento diferente para um dataframe especifico, o dataframe order\_details sofre alterações individuais em cada campo que possui, para otimizar os dados antes de fazer o upsert dos dados, o tratamento esta detalhado no arquivo leitura\_tratamento\_insert\_dados.ipynb, essa é a parte de tratamento que usamos nesse projeto.

2.4 Carregamento dos dados no Data Warehouse

O carregamento dos dados para o Data Warehouse foi feito através de uma função, essa função esta criada no arquivo extracao\_dados.py disponível no zip da documentação, a função responsável para carregamento dos dados chama-se “load\_data\_to\_postgres”, nela passamos algumas variáveis como “df”, “table\_name”, “unique\_keys” e “column\_key”, “df” é o dataframe após o tratamento, “table\_name” é o nome da tabela que o data Warehouse espera, “unique\_keys” é a chave primaria ou as chaves primarias que as tabelas possuem e “column\_key” é um conjunto de tabelas usadas para validar a duplicidade caso a tabela não possua chave primaria, a função segue essa sequência, montamos a conexão URL como o Data Warehouse que está no PostgreSQL, criamos um cursor para executar os comandos SQL, em um bloco try, pegamos os nomes da colunas e deixamos em uma lista, montamos um insert dos dados com as colunas de cada tabela, verificamos se a tabela possui chave primaria com o bloco if unique\_keys:, se a tabela possuir chaves primarias ele faz o upsert usando o comando ON CONFLICT, caso a tabela não possua chave primaria, ele verifica o campo “column\_key”, pega os nomes das colunas que estão no “column\_key”, faz uma consulta direta no data Warehouse, filtra os dados se baseando nas colunas informada no “column\_key” para não armazenar dados duplicados, se após o filtro houver dados ele faz o upsert somente dos dados novos, caso não tenha dados novos ele não faz a carga incremental, assim é realizado o insert, upsert de todos os dados tratados, sendo csv ou sql, enviado eles diariamente.

3 Infraestrutura

Aqui mostraremos como foi criada a estrutura do projeto, que ferramentas utilizamos durante a modelagem de cada parte do processo.

3.1 Ambiente Utilizado

Usamos o Ambiente Local Windows e a Cloud ElephantSQL, criamos 2 banco de dados no ElephantSQL, um sendo usado como consulta diária para extração dos dados e o outro sendo usado para armazenamento dos dados tratados sendo nosso data Warehouse, a montagem da esteira de desenvolvimento está sendo feita pelo Visual Studio Code, estamos usando também um arquivo bash para armazenar variáveis de ambientes como, link de conexão com o banco de dados, link de download do arquivo csv, para armazenar esses dados de forma dinâmica temos uma arquivo python chamado credencias.py que acessa nosso arquivo bash para buscar as informações do arquivo e deixar disponíveis para acessar dentro dos scripts, bibliotecas necessárias para o funcionamento do projeto, plotly, os, psycopg2, pandas, gdown, tempfile e datetime todas as bibliotecas disponíveis gratuitamente pela web utilizando Python.

3.2 Modelagem de dados para Power BI

Após a analise dos dados, criamos um modelo estrela para análise do Power BI, criamos 2 tabelas fatos e 12 tabelas dimensões, a descrição do dicionário de dados esta no arquivo Modelo estrela BI.drawio.png, assim foi possível tirar insight de negócios de duas fontes de fatos distintas, a fact\_orders que possuem dimensões como dim\_customers, dim\_employees, dim\_employee\_territories, dim\_territories e dim\_region, essas tabelas são usadas para tirar visualizações e relatórios de negocio e temos a outra fato que é a fact\_order\_datails que possuem dimensões como dim\_categories, dim\_products e dim\_suppliers nós dando autonomia para a geração dos relatórios, não usamos o Power BI para fazer a analises dos dados, eles estão sendo feito via script com a biblioteca plotly do Python sendo executadas diretamente no notebook, a conexão com o data Warehouse está sendo feita através da conexão via URL do banco de dado PostgreSQL.

4 Análises Visuais

Aqui demonstraremos algumas análises visuais que criamos a partir dos dados fornecidos, e relatórios que acreditamos serem importante para os negócios da empresa.

4.1 Análises e Relatórios da tabela Fact\_order\_datails

A tabela Fact\_order\_datails, possuem tabelas de dimensões sendo elas as tabelas, dim\_products, dim\_suppliers e a dim\_categories, para analise dos dados fizemos a junção dessa tabelas através das chaves primarias e estrangeiras existentes nelas, sendo as chaves product\_id, supplier\_id e category\_id, após esses joins temos um df com todos os dados da fato e das dimensões, com isso criei um campo calculado com o total de vendas sendo “total\_sales”, a partir desse campo eu criei as seguintes visualizações como total de vendas por produtos, total de venda por categoria, desconto médio por produto, total de vendas por região e total de vendas por fornecedor, todas as analises disponíveis no caminho “\Projeto Esteira BI\Projeto\hml\_config\analises\order\_datails\_visual\”, script onde foi feito as analises chama-se “analise\_dados.ipynb”, não foi criada uma conexão com o Power BI, as analises estão sendo feita via script com o a biblioteca plotly do Python, script disponível com a documentação do projeto, a geração de relatórios está disponível também através do caminho “\Projeto Esteira BI\Projeto\hml\_config\analises\order\_datails\”, criamos 5 relatórios em csv para análise dos dados disponíveis.

4.2 Análises e Relatórios da tabela Fact\_orders

A tabela Fact\_orders, possuem tabelas de dimensões sendo elas as tabelas, dim\_customers, dim\_employees, dim\_employee\_territories, dim\_territories e dim\_region, para análise dos dados fizemos a junção das tabelas através das chaves primaria e estrangeiras existentes nelas, sendo as chaves customer\_id, employee\_id, territory\_id e region\_id, após esses joins temos um DataFrame como todos os dados da fato e das dimensões, com isso criei um campo calculado com o total de vendas sendo “total\_sales” supondo que o campo “freight” que traduzindo fica frete, seria o campo que representa a venda, a partir desse campo eu criei as seguintes visualizações como total de vendas por região, total de vendas por funcionário, total de vendas por cliente, total de vendas por território, total de vendas por data de pedido, todas as analises disponíveis no caminho “\Projeto Esteira BI\Projeto\hml\_config\analises\orders\_visual”, script onde foi feito as analises chama-se “analise\_dados.ipynb”, não foi criada uma conexão com o Power BI, as analises estão sendo feita via script com o a biblioteca plotly do Python, script disponível com a documentação do projeto, a geração de relatórios está disponível também através do caminho “\Projeto Esteira BI\Projeto\hml\_config\analises\ orders \”, criamos 5 relatórios em csv para análise dos dados disponíveis.

5 Relatórios de Qualidade de dados

Aqui falaremos dos relatórios de qualidade de dados, criamos funções para gerar relatórios de dados e descreveremos essas funções e o que elas nos ajudam no dia a dia de processamento.

5.1 Relatório de Completude

O relatório de completude é uma análise que verifica se todos os campos obrigatórios em um conjunto de dados possuem valores preenchidos, ou seja, identifica a presença de valores nulos ou faltantes em um DataFrame. Esse tipo de relatório é crucial para entender a qualidade dos dados e identificar possíveis problemas que possam afetar análises ou decisões baseadas nesses dados, criamos uma função dentro do arquivo monitoramento\_qualidadde\_dados.py que faz essa validação de completude, a função chama-se “generate\_completeness\_report”, passamos duas variáveis nela sendo o “df” e “table\_name”, e a função segue essa sequência, no completeness\_report ele pega o dataframe no “df” passando os comandos isnull() que retorna um valor booleano true para cada linha que estiver nula ou NAN, mean() que calcula a média de cada valor true correspondente no dataframe, e reset\_index() que transforma o índice em uma coluna, convertendo o resultado em um dataframe, a seguir temos o completeness\_report.columns que pega os valores das colunas transformadas acima e renomeia os nomes delas para ['column', 'missing\_percentage'] e por fim o comando completeness\_report['table'] = table\_name que basicamente adiciona o nome da tabela no dataframe criado, assim temos um relatório com os nomes das colunas, o percentual de valores vazios e o nome da tabela a cada linha do relatório, exemplo de relatório disponível no caminho “\Projeto Esteira BI\Projeto\hml\_config\relatorios\” com o nome “completeness\_report\_{data do dia }.csv”, esse relatório é gerado de forma automatizada no final de cada upsert realizado para o data Warehouse.

5.2 Relatório de Unicidade

O relatório de unicidade é uma análise que verifica a presença de registros duplicados em colunas ou combinações de colunas que devem ter valores únicos. Isso é crucial em situações onde a unicidade é essencial, como chaves primárias em bancos de dados ou identificadores únicos em conjuntos de dados, criamos uma função dentro do arquivo monitoramento\_qualidadde\_dados.py que faz essa validação da unicidade de cada dataframe, a função chama-se “generate\_uniqueness\_report”, passamos 3 variáveis nela sendo o “df”, “table\_name” e “unique\_keys”, a função segue essa sequência, primeiro ele verifica se há alguma chave primaria, se não houver ele avisa que não tem chave primaria e gera um df vazio, depois ele cria uma lista vazia que vai armazenar os dados do relatório, após isso tem o loop for para fazer o calculo da unicidade para cada coluna usando a “unique\_keys” como base, ele verifica se a unique\_keys existe no dataframe, calcula o número de valores únicos usando o comando “df[columns].nunique()”, calcula o total de registro do dataframe com o comando “len(df)”, calcula a porcentagem de unicidade com o comando “(unique\_count / total\_count) \* 100” e por fim armazena o valor da porcentagem na lista de dados que foi criada, e por fim transforma essa lista em dataframe onde podemos gerar o relatório csv, exemplo de relatório disponível no caminho “\Projeto Esteira BI\Projeto\hml\_config\relatorios\” com o nome “uniqueness\_report\_{data do dia}.csv”, esse relatório é gerado de forma automatizada no final de cada upsert realizado para o data Warehouse.

5.3 Relatório de Resumo

O relatório de resumo é uma análise que apresenta informações chave de um DataFrame ou tabela de dados de forma concisa e organizada. Geralmente, um relatório de resumo inclui métricas básicas ou metadados importantes que ajudam a entender rapidamente os aspectos fundamentais dos dados, criamos uma função dentro do arquivo monitoramento\_qualidadde\_dados.py que faz esse resumo para cada dataframe, a função chama-se “generate\_summary\_report”, passamos duas variáveis nela sendo o “df” e o “table\_name”, a função segue essa sequência, primeiro ele armazena o “df” e busca a quantidade de linhas que tem dentro dele com o comando “quantity = len(df)”, em seguida ele pega o dia de hoje com a variável current\_date com o comando “datetime.now().strftime('%Y-%m-%d')”, por fim ele cria um dataframe com o resumo das informações, criando os campos ‘table\_name’, ‘quantity’ e ‘date’, onde podemos gerar um relatório em csv, exemplo de relatório disponível no caminho “\Projeto Esteira BI\Projeto\hml\_config\relatorios\” com o nome “summary\_report\_{data do dia}.csv”, esse relatório é gerado de forma automatizada no final de cada upsert realizado para o data Warehouse.

Final

Finalizando o projeto, enviaremos todos os arquivos mencionados neste documento no arquivo Projeto Esteira BI.zip, esse arquivo terá a seguinte estrutura de pastas.

Material do projeto

“Projeto Esteira BI.zip\Projeto Esteira BI\material do projeto\”

Desenvolvimento do projeto

“Projeto Esteira BI.zip\Projeto Esteira BI\Projeto\hml\_config\”

Dicionário de dados e arquivo bash do projeto

“Projeto Esteira BI.zip\Projeto Esteira BI\Projeto\arquivos\”

Análises e Relatórios do projeto

“Projeto Esteira BI.zip \Projeto Esteira BI\Projeto\hml\_config\analises\”

Relatório de Monitoramento e Qualidade de dados

“Projeto Esteira BI.zip \Projeto Esteira BI\Projeto\hml\_config\relatorios\”

Todo o desafio proposto foi realizado, só não houve conexão com o Power BI, por não fazer o uso da ferramenta, os desafios de extração, tratamento, modelagem, ingestão e análises dos dados foram realizados, concluímos aqui a documentação deste projeto.