[Mini Projeto] Pipeline de Dados e Análise com SQL no BigQuery

> 💡 **Atenção!**
> Vocês terão perguntas nos exercícios abaixo que devem ser documentadas e repondidas no arquivo README.md que devem subir no repositório Github do grupo.

▸ Conteúdo Guiado

1. Introdução: A Missão da Livraria DevSaber

Contexto para o professor transmitir:

"Olá, pessoal! Hoje, vamos deixar de ser apenas estudantes de SQL para nos tornarmos analistas de dados em uma missão real. Nossa cliente é a 'Livraria DevSaber', uma nova loja online que fez suas primeiras vendas e, até agora, anotou tudo em uma planilha. Isso é um começo, mas para crescer, eles precisam de *insights*. Nossa missão é transformar essa planilha em um *mini data warehouse* inteligente no Google BigQuery. Vamos construir todo o pipeline de dados: desde criar a estrutura, carregar os dados, até extrair as respostas que ajudarão a livraria a entender seus negócios."

Perguntas para a turma:

- Por que uma planilha não é ideal para uma empresa que quer analisar suas vendas a fundo?
- Que tipo de perguntas vocês acham que o dono da livraria gostaria de responder com esses dados?

---

2. Estruturando o Armazenamento: `CREATE TABLE` no BigQuery

Contexto para o professor transmitir:

"O primeiro passo em qualquer projeto de dados é construir a 'casa' onde os dados vão morar. No BigQuery, fazemos isso com o `CREATE TABLE`. Mas, diferente dos bancos de dados tradicionais, o BigQuery tem suas próprias regras de arquitetura. Ele é construído para ser incrivelmente rápido com volumes de dados gigantescos, e isso muda um pouco a forma como definimos nossas tabelas."

Exemplo: O código para criar a tabela `Clientes`

```
- No BigQuery, usamos tipos de dados como STRING e INT64. CREATE OR REPLACE TABLE `seu-projeto.seu_dataset.Clientes` ( ID_Cliente INT64, Nome_Cliente STRING, Email_Cliente STRING, Estado_Cliente STRING );
```

Comentários:

- Usamos `CREATE OR REPLACE TABLE` para que nosso script possa ser executado várias vezes sem erros.
- Note os tipos de dados: `VARCHAR` vira `STRING`, e `INT` vira `INT64`.
- **Paradigma Sem Chaves:** Onde estão a `PRIMARY KEY` e a `FOREIGN KEY`? O BigQuery **não as utiliza**! Os relacionamentos são lógicos e nós vamos defini-los mais tarde, na hora da consulta, usando o `JOIN`.

Perguntas para a turma:

- Com base nos dados brutos, quais outras duas tabelas precisamos criar? Que colunas e tipos de dados elas teriam?
- Se o BigQuery não tem chaves estrangeiras, como garantimos que um `ID_Cliente` na tabela de vendas realmente existe na tabela de clientes? (Resposta: A responsabilidade é nossa, na hora de construir a consulta com o `JOIN`).

---

3. Ingerindo os Dados: `INSERT INTO`

Contexto para o professor transmitir:

"Com as tabelas criadas, é hora de fazer a mudança: vamos carregar os dados da nossa 'planilha' para dentro do BigQuery. Para o volume de dados do nosso projeto, o comando `INSERT INTO` é perfeito. Ele nos permite inserir os registros linha a linha, de forma clara e controlada."

Exemplo: Inserindo os primeiros clientes

```
- Inserimos apenas clientes únicos para evitar duplicidade. INSERT INTO
`seu-projeto.seu_dataset.Clientes` (ID_Cliente, Nome_Cliente,
Email_Cliente, Estado_Cliente) VALUES (1, 'Ana Silva', 'ana.s@email.com',
'SP'), (2, 'Bruno Costa', 'b.costa@email.com', 'RJ');
```

Comentários:

- A sintaxe é a mesma que já conhecemos do SQL padrão.
- Separamos os dados em tabelas normalizadas. Aqui, inserimos apenas os clientes únicos. Na tabela `Produtos`, faremos o mesmo. A tabela `Vendas` irá conectar tudo.

Perguntas para a turma:

- Por que é uma boa prática inserir os clientes e produtos em suas próprias tabelas antes de inserir os dados de vendas?
- Em um cenário com milhões de vendas por dia, o `INSERT INTO` seria a melhor abordagem?

---

4. Análise de Dados: Fazendo Perguntas com `SELECT` e `JOIN`

Contexto para o professor transmitir:

"Esta é a parte mais poderosa e divertida! Com os dados estruturados, agora podemos fazer perguntas de negócio e obter respostas imediatas. Usaremos tudo o que aprendemos: `SELECT` para escolher o que queremos ver, `WHERE` para filtrar, `JOIN` para conectar nossas tabelas e `GROUP BY` para agregar e resumir informações."

Exemplo: Respondendo à Pergunta 3 do projeto*Pergunta: Listar todas as vendas, mostrando o nome do cliente, o nome do produto e a data da venda.*

```
SELECT C.Nome_Cliente, P.Nome_Produto, V.Data_Venda FROM `seu-projeto.seu_dataset.Vendas` AS V JOIN `seu-projeto.seu_dataset.Clientes` AS C ON V.ID_Cliente = C.ID_Cliente JOIN `seu-projeto.seu_dataset.Produtos` AS P ON V.ID_Produto = P.ID_Produto ORDER BY V.Data_Venda;
```

Comentários:

- Aqui, o relacionamento lógico que não existia no `CREATE TABLE` é finalmente criado pelo `JOIN`.
- Conectamos as três tabelas para construir uma resposta que seria impossível de obter com elas isoladas.

---

5. Automação e Reuso: Criando uma `VIEW`

Contexto para o professor transmitir:

"A última consulta que fizemos, com três `JOINs`, é muito útil. Provavelmente, a equipe da livraria vai querer vê-la todos os dias. Vamos reescrevê-la toda vez? Não! Vamos automatizar. Uma `VIEW` no BigQuery é como salvar uma consulta complexa com um nome simples. Ela se torna uma 'tabela virtual' que podemos consultar de forma muito mais fácil."

Exemplo: Criando a `VIEW` do relatório detalhado

```
CREATE OR REPLACE VIEW `seu-projeto.seu_dataset.v_relatorio_vendas_detalhado` AS SELECT V.ID_Venda, V.Data_Venda, C.Nome_Cliente, P.Nome_Produto, V.Quantidade, (V.Quantidade * P.Preco_Produto) AS Valor_Total FROM `seu-projeto.seu_dataset.Vendas` AS V JOIN `seu-projeto.seu_dataset.Clientes` AS C ON V.ID_Cliente = C.ID_Cliente JOIN `seu-projeto.seu_dataset.Produtos` AS P ON V.ID_Produto = P.ID_Produto;
```

Comentários:

- Agora, em vez de escrever toda a consulta de `JOIN` novamente, podemos simplesmente fazer: `SELECT * FROM v_relatorio_vendas_detalhado WHERE Nome_Cliente = 'Ana Silva';`
- A `VIEW` simplifica o acesso aos dados e garante que todos na empresa usem a mesma lógica de cálculo.

Perguntas para a turma:

- Qual é a principal vantagem de usar uma `VIEW` em vez de simplesmente salvar o código em um arquivo de texto?
- Se o preço de um produto mudar na tabela `Produtos`, o `Valor_Total` na `VIEW` será atualizado automaticamente na próxima vez que a consultarmos?

---

6. Conclusão: Seu Primeiro Projeto de Portfólio!

Contexto para o professor transmitir:

"Parabéns! Vocês acabaram de construir um pipeline de dados completo e funcional no BigQuery. Vocês passaram por todas as etapas de um analista de dados: entenderam o problema, modelaram os dados, ingeriram as informações e, o mais importante, extraíram valor com análises. O próximo passo é organizar esses três scripts (`CREATE`, `INSERT`, `ANALYSIS`) e um bom `README.md` em um repositório no GitHub. Este não é apenas um exercício de aula; é o primeiro projeto real do seu portfólio."

4. Exercícios

1. Contexto e Cenário

A "Livraria DevSaber", uma loja online, registrou suas primeiras vendas e precisa da sua ajuda para estruturar e analisar esses dados. Sua missão é criar um pequeno *data warehouse* no Google BigQuery para permitir que a empresa responda a perguntas de negócio importantes sobre seus clientes e produtos.

2. Missão do Projeto

Você deve criar um conjunto de scripts SQL para:

1. **Definir o Schema**: Criar as tabelas `Clientes`, `Produtos` e `Vendas`.
2. **Ingerir os Dados**: Inserir os dados brutos fornecidos nas tabelas.
3. **Analisar os Dados**: Escrever consultas SQL para responder a perguntas de negócio.
4. **Criar uma View**: Construir uma `VIEW` para simplificar análises futuras.

3. Dados de Origem

Os dados brutos fornecidos pela empresa são:

id_venda	nome_cliente	email_cliente	estado_cliente	nome_produto	catego
1	Ana Silva	ana.s@email.com	SP	Fundamentos de SQL	Dados
2	Bruno Costa	b.costa@email.com	RJ	Duna	Ficção
3	Carla Dias	carla.d@email.com	SP	Python para Dados	Progra
4	Ana Silva	ana.s@email.com	SP	Duna	Ficção
5	Daniel Souza	daniel.s@email.com	MG	Fundamentos de SQL	Dados
6	Bruno Costa	b.costa@email.com	RJ	O Guia do Mochileiro	Ficção

Arquivo 1: `01_create_tables_bigquery.sql`

```
-- Este script cria a estrutura do schema no Google BigQuery. -- Tabelas são criadas com `CREATE OR REPLACE TABLE` para permitir a execução repetida do script. -- Lembre-se de substituir `seu-projeto.seu_dataset` pelo nome do seu projeto e dataset. -- Tabela de Clientes -- Armazena informações únicas de cada cliente. -- No BigQuery, chaves primárias não são impostas, mas ID_Cliente serve como identificador lógico. CREATE OR REPLACE TABLE `seu-projeto.seu_dataset.Clientes` ( ID_Cliente INT64, Nome_Cliente STRING, Email_Cliente STRING, Estado_Cliente STRING ); -- Tabela de Produtos -- Armazena informações únicas de cada produto. CREATE OR REPLACE TABLE `seu-projeto.seu_dataset.Produtos` ( ID_Produto INT64, Nome_Produto STRING, Categoria_Produto STRING, Preco_Produto NUMERIC ); -- Tabela de Vendas -- Tabela de fatos que relaciona clientes e produtos, registrando cada transação. -- As relações com Clientes e Produtos são lógicas, mantidas pelos campos de ID. CREATE OR REPLACE TABLE `seu-projeto.seu_dataset.Vendas` ( ID_Venda INT64, ID_Cliente INT64, ID_Produto INT64, Data_Venda DATE, Quantidade INT64 );
```

Arquivo 2: `02_insert_data_bigquery.sql`

```
-- Este script popula as tabelas criadas no BigQuery. -- A cláusula VALUES é usada para inserir múltiplas linhas de uma vez. -- Inserção de dados na tabela Clientes (sem duplicatas) INSERT INTO `seu-projeto.seu_dataset.Clientes` (ID_Cliente, Nome_Cliente, Email_Cliente, Estado_Cliente) VALUES (1, 'Ana Silva', 'ana.s@email.com', 'SP'), (2, 'Bruno Costa', 'b.costa@email.com', 'RJ'), (3, 'Carla Dias', 'carla.d@email.com', 'SP'), (4, 'Daniel Souza', 'daniel.s@email.com', 'MG'); -- Inserção de dados na tabela Produtos (sem duplicatas) INSERT INTO `seu-projeto.seu_dataset.Produtos` (ID_Produto, Nome_Produto, Categoria_Produto, Preco_Produto) VALUES (101, 'Fundamentos de SQL', 'Dados', 60.00), (102, 'Duna', 'Ficção Científica', 80.50), (103, 'Python para Dados', 'Programação', 75.00), (104, 'O Guia do Mochileiro', 'Ficção Científica', 42.00); -- Inserção de dados na tabela Vendas INSERT INTO `seu-projeto.seu_dataset.Vendas` (ID_Venda, ID_Cliente, ID_Produto, Data_Venda, Quantidade) VALUES (1, 1, 101, '2024-01-15', 1), (2, 2, 102, '2024-01-18', 1), (3, 3, 103, '2024-02-02', 2), (4, 1, 102, '2024-02-10', 1), (5, 4, 101, '2024-02-20', 1), (6, 2, 104, '2024-03-05', 1);
```

Arquivo 3: `03_analysis_and_view_bigquery.sql`

```
-- Este script contém as consultas para análise e a criação da VIEW. -- Lembre-se de substituir `seu-projeto.seu_dataset` pelo nome do seu projeto e dataset. -- ANÁLISE DE DADOS -- -- Pergunta 1: Qual o nome dos clientes que moram no estado de 'SP'? SELECT Nome_Cliente FROM `seu-projeto.seu_dataset.Clientes` WHERE Estado_Cliente = 'SP'; -- Pergunta 2: Quais produtos pertencem à categoria 'Ficção Científica'? SELECT
```