

Exercício Prático - Trabalhando com JSON e Nested Types no Hive

Atividade 01 - Escolher um dataset com dados no formato JSON, armazená-los em uma pasta no HDFS, isto é, ler um arquivo JSON contendo dados aninhados (nested) e criar uam tablea Hive com STRUCT e ARRAY, e executar consultas para manipular os dados.

The screenshot shows the Apache Ambari File Browser interface. On the left, there's a sidebar with various datasets like bronze_links, bronze_movies, etc. In the center, a file browser window is open, showing a file named 'vendas_nested.json'. The file content is displayed as a JSON array:

```
[{"id_venda": 1, "cliente": {"id": 101, "nome": "Ana"}, "produtos": ["Notebook", "Mouse"], "total": 5100.00}, {"id_venda": 2, "cliente": {"id": 102, "nome": "João"}, "produtos": ["Teclado"], "total": 200.00}, {"id_venda": 3, "cliente": {"id": 103, "nome": "Carlos"}, "produtos": ["Monitor", "Cadeira Gamer", "Webcam"], "total": 2350.50}, {"id_venda": 4, "cliente": {"id": 104, "nome": "Mariana"}, "produtos": ["Smartphone", "Carregador Portátil"], "total": 3400.75}]
```

Aqui, cria a tabela **Hive** com **STRUCT** e **ARRAY**

The screenshot shows the Apache Ambari Hive Query Editor. On the left, there's a sidebar with various datasets. In the center, a query editor window is open, showing the creation of a table 'vendas_json' using the following SQL code:

```
-- Atividade Prática: Trabalhando com JSON e Nested Types no Hive
CREATE EXTERNAL TABLE vendas_json (
    id_venda INT,
    cliente STRUCT<id:INT, nome:STRING>,
    produtos ARRAY<STRING>,
    total DOUBLE
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/admin/vendas';
```

Below the query, a success message is displayed: "Success."

Exemplo que consulta

Query

Search data and saved documents...

default Tables (20) + ⌂

bronze_links
bronze_movies
bronze_ratings
bronze_tags
clients
flightdelays
flights_weather
gold_filme_genero_detalhado
gold_genero_analise_popularidade
pedidos
sfo_weather
sfo_weather_txt
silver_links
silver_movies
silver_ratings
silver_tags
values_tmp_table_1
values_tmp_table_2
vendas_json
weather_partitioned

```

67 -- Consultas
68
69 SELECT
70   id_venda,
71   cliente.nome AS nome_cliente,
72   produto
73 FROM vendas_json
74 LATERAL VIEW explode(produtos) exploded_table AS produto;
75
76
77 SELECT id_venda, cliente.nome, size(produtos) AS qtd_produtos
78 FROM vendas_json
79 WHERE size(produtos) > 1;
80
81 SELECT cliente.nome, total
82 FROM vendas_json
83 WHERE size(produtos) = 0;
84

```

Query History Saved Queries Query Builder Results (8)

	id_venda	nome_cliente	produto
1	1	Ana	Notebook
2	1	Ana	Mouse
3	2	João	Teclado
4	3	Carlos	Monitor
5	3	Carlos	Cadeira Gamer
6	3	Carlos	Webcam

Query History Saved Queries Query Builder Results (3)

	id_venda	nome	qtd_produtos
1	1	Ana	2
2	3	Carlos	3
3	4	Mariana	2

✓ Done. 0 results.

Query History Saved Queries Query Builder

poucos segundos atrás ➔	SELECT cliente.nome, total FROM vendas_json WHERE size(produtos) = 0
poucos segundos atrás ✓	SELECT id_venda, cliente.nome, size(produtos) AS qtd_produtos FROM vendas_json WHERE size(produtos) > 1

Query History Saved Queries Query Builder Results (4)

	id_venda	id_cliente	nome_cliente	total
1	1	101	Ana	5100
2	2	102	João	200
3	3	103	Carlos	2350.5
4	4	104	Mariana	3400.75