Objetivo

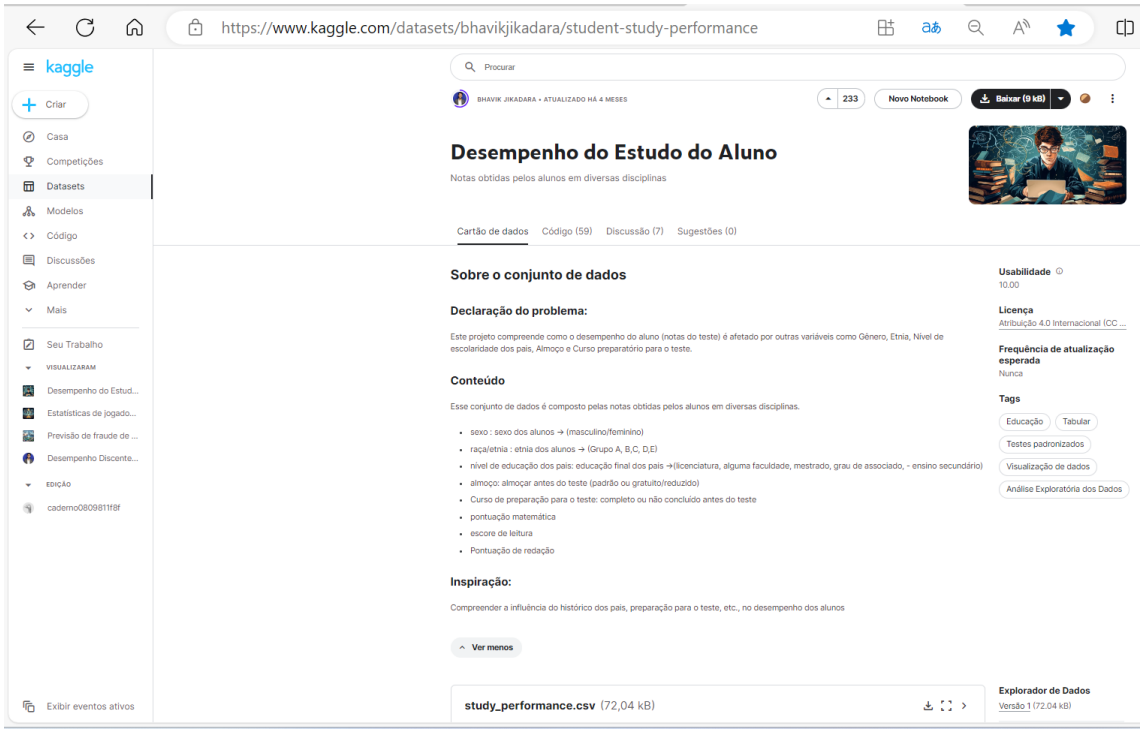
A partir de dados de um estudo sobre o desempenho de alunos em testes, analisar:

1. Quantos alunos tiraram nota acima de 80 em Matemática que tiveram almoço padrão?
2. Qual o percentual de alunos do estudo que tiraram nota acima de 80 em Matemática que tiveram almoço padrão?

Detalhamento

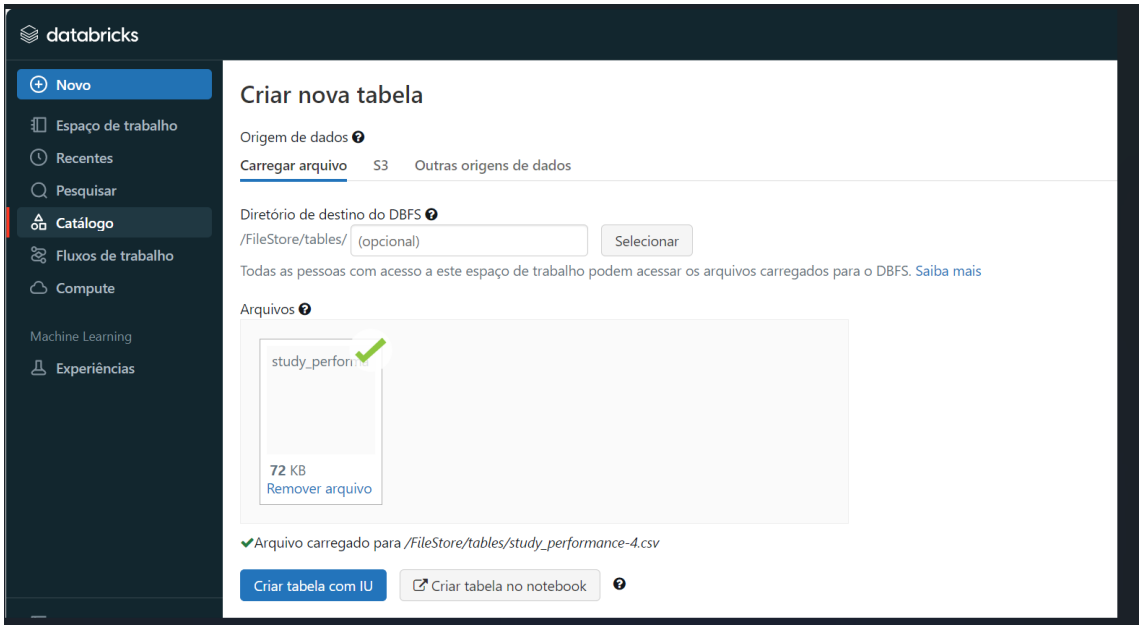
Busca pelos dados

[Student Study Performance (kaggle.com)]



Coleta

Dados baixados para máquina local e inseridos manualmente no DBFS (Databricks File System).



Modelagem

Foi feito um Data Lake adotando um modelo flat por conceito representado por uma tabela única buscando ter flexibilidade, performance nas consultas e simplicidade, adequado ao volume de dados do arquivo.

Catálogo de Dados

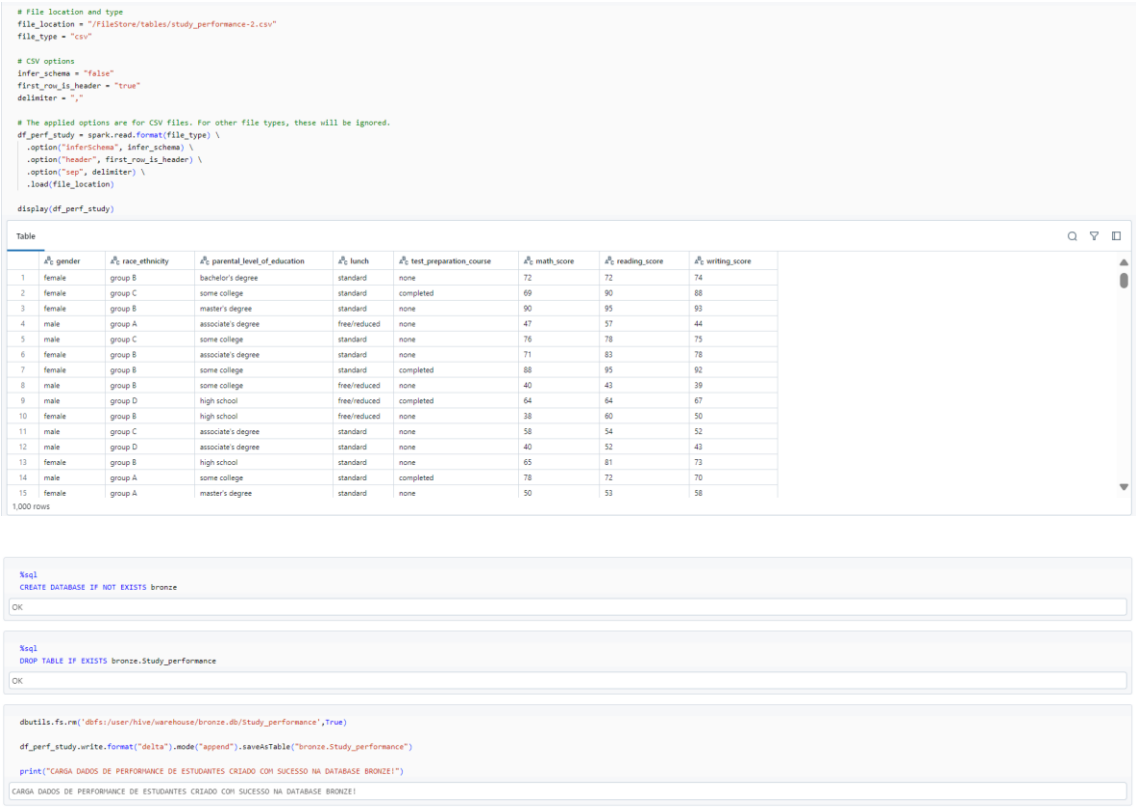
- gender: Sexo dos alunos. Domínio -> (Male, female) *(Masculino, Feminino)*
- race_ethnicity: Etnia dos alunos. Domínio -> (Group A, B, C, D, E)
- parental_level_of_education: Nível de educação dos pais. Domínio -> (bachelor's degree, some college, master's degree, associate's degree, high school) *(bacharelado, alguma faculdade, mestrado, diploma concedido após um curso de 2 anos, ensino médio)*
- lunch: Tipo de almoço antes do teste. Domínio -> (standard, free/reduced) *(padrão, gratuito/baixo custo)*
- test_preparation_course: Curso de preparação para o teste. Domínio -> (completed, none) *(completo, incompleto)*
- math_score: Nota no teste de matemática. Valor mínimo: 0 e Valor Máximo: 100
- reading_score: Nota no teste de leitura. Valor mínimo: 0 e Valor Máximo: 100
- writing_score: Nota no teste de redação. Valor mínimo: 0 e Valor Máximo: 100

Os dados são provenientes do estudo realizado para compreender como o desempenho do aluno nos testes de matemática, leitura e redação pode ser afetado por variáveis como gênero, etnia, nível de escolaridade dos pais, tipo do almoço e curso preparatório para o teste e foram baixados do Kaggle, não sendo necessária nenhuma técnica para compor o conjunto de dados.

Carga

O ETL foi realizado através de um notebook executado no Databricks Community Edition utilizando comandos Python e SQL.

Na etapa 1, foram realizados os comandos para extrair (*Extract*) os dados da fonte no DBFS colocando em um dataset e a partir dele criando um banco de dados BRONZE com os dados originais do arquivo.



```
# File location and type
file_location = "/FileStore/tables/study_performance-2.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df_perf_study = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df_perf_study)
```

Table

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_score	reading_score	writing_score
1	female	group B	bachelor's degree	standard	none	72	72	74
2	female	group C	some college	standard	completed	69	90	88
3	female	group B	master's degree	standard	none	90	95	93
4	male	group A	associate's degree	free/reduced	none	47	57	44
5	male	group C	some college	standard	none	76	78	75
6	female	group B	associate's degree	standard	none	71	83	78
7	female	group B	some college	standard	completed	88	95	92
8	male	group B	some college	free/reduced	none	40	43	39
9	male	group D	high school	free/reduced	completed	64	64	67
10	female	group B	high school	free/reduced	none	38	60	50
11	male	group C	associate's degree	standard	none	58	54	52
12	male	group D	associate's degree	standard	none	40	52	43
13	female	group B	high school	standard	none	65	81	73
14	male	group A	some college	standard	completed	78	72	70
15	female	group A	master's degree	standard	none	50	53	58

1,000 rows

```
%sql
CREATE DATABASE IF NOT EXISTS bronze
```

OK

```
%sql
DROP TABLE IF EXISTS bronze.Study_performance
```

OK

```
dbutils.fs.rm('dbfs:/user/hive/warehouse/bronze.db/Study_performance',True)

df_perf_study.write.format("delta").mode("append").saveAsTable("bronze.Study_performance")

print("CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE BRONZE!")
```

CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE BRONZE!

bronze.study_performance | Atualizar

My Cluster

Detalhes Histórico

Atualizar

Descrição:
Criada em: 2024-07-07 13:56:45
Última modificação: 2024-07-07 13:56:47
Colunas de partição:
Número de arquivos: 1
Tamanho: 7.64 kB

Esquema:

	col_name	data_type	comment
1	gender	string	null
2	race_ethnicity	string	null
3	parental_level_of_educati...	string	null
4	lunch	string	null
5	test_preparation_course	string	null
6	math_score	string	null
7	reading_score	string	null
8	writing_score	string	null

Na etapa 2, foram realizados os comandos para realizarmos a etapa de transformação (*Transform*) dos dados, convertendo os campos “math_score”,

```
## carregando os dados do DATABASE bronze para tratamento dos dados de performance dos estudantes

df_perf_study_bronze_sql = spark.sql('''select * from bronze.Study_performance''')
```

```
from pyspark.sql.functions import col

df_perf_study_silver = df_perf_study_bronze_sql.withColumn("math_score", col("math_score").cast("int")) \
                       .withColumn("reading_score", col("reading_score").cast("int")) \
                       .withColumn("writing_score", col("writing_score").cast("int"))
```

```
dbutils.fs.rm('dbfs:/user/hive/warehouse/silver.db/Study_performance',True)

df_perf_study_silver.write.format("delta").mode("append").saveAsTable("silver.Study_performance")

print("CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE SILVER!")
```

```
CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE SILVER!
```

silver.study_performance | Atualizar

My Cluster

Detalhes Histórico

Atualizar

Descrição:
Criada em: 2024-07-07 13:56:52
Última modificação: 2024-07-07 13:56:54
Colunas de partição:
Número de arquivos: 1
Tamanho: 7.68 kB

Esquema:

	col_name	data_type	comment
1	gender	string	null
2	race_ethnicity	string	null
3	parental_level_of_educati...	string	null
4	lunch	string	null
5	test_preparation_course	string	null
6	math_score	int	null
7	reading_score	int	null
8	writing_score	int	null

Na etapa 3, foram realizados os comandos para realização da etapa de carregamento (*Load*) dos dados transformados no banco de dados GOLD.

```
%sql
DROP TABLE IF EXISTS gold.Study_performance
```

```
OK
```

```
df_perf_study_gold_sql = spark.sql('''select * from silver.Study_performance''')
```

```
dbutils.fs.rm('dbfs:/user/hive/warehouse/gold.db/Study_performance',True)

df_perf_study_gold_sql.write.format("delta").mode("append").saveAsTable("gold.Study_performance")

print("CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE GOLD!")
```

```
CARGA DADOS DE PERFORMANCE DE ESTUDANTES CRIADO COM SUCESSO NA DATABASE GOLD!
```

gold.study_performance | Atualizar

My Cluster

Detalhes | Histórico

Atualizar

Descrição:
Criada em: 2024-07-07 13:56:58
Última modificação: 2024-07-07 13:57:01
Colunas de partição:
Número de arquivos: 1
Tamanho: 7.68 kB

Esquema:

	col_name	data_type	comment
1	gender	string	null
2	race_ethnicity	string	null
3	parental_level_of_educati...	string	null
4	lunch	string	null
5	test_preparation_course	string	null
6	math_score	int	null
7	reading_score	int	null
8	writing_score	int	null

Análise

Qualidade dos Dados

Analisando os atributos do conjunto de dados, foi observado que estes já haviam sido curados e bem tratados antes de serem disponibilizados. Não há valores nulos e os valores dos campos do tipo texto estão de acordo com o domínio definido no catálogo de dados, bem como os campos do tipo int possuem valores inteiros dentro dos valores mínimo e máximo definidos.

Solução do Problema

Foi possível responder todas as perguntas definidas preliminarmente nos objetivos através de consultas SQL realizadas com a utilização da biblioteca pyspark do Python.

```
from pyspark.sql.functions import count

# Filtragem e contagem dos alunos que tiraram nota acima de 80 em Matemática e tiveram almoço padrão
num_students_gold = df_perf_study_gold_sql.filter((col("math_score") > 80) & (col("lunch") == "standard")) \
                                          .agg(count("*").alias("num_students"))

# Exibição do resultado
num_students_gold.show()
```

```
+------------+
|num_students|
+------------+
|         156|
+------------+
```

```
from pyspark.sql.functions import col

# Filtrar alunos com nota de Matemática acima de 80 e almoço padrão
students_filtered = df_perf_study_gold_sql.filter((col('math_score') > 80) & (col('lunch') == 'standard'))
```

```
# Contar o número de alunos que satisfazem os critérios
total_students = df_perf_study_gold_sql.count()
students_above_80 = students_filtered.count()

# Calcular o percentual
percent_above_80 = (students_above_80 / total_students) * 100

# Arredondar para duas casas decimais
percent_above_80 = round(percent_above_80, 2)

print(f"Percentual de alunos com nota acima de 80 em Matemática e almoço padrão: {percent_above_80}%")
```

```
Percentual de alunos com nota acima de 80 em Matemática e almoço padrão: 15.6%
```

O notebook contendo o código executado no Databricks Community Editon está disponível em [Edmout/Engenharia-da-Dados-MVP: Engenharia da Dados MVP (github.com)]