

MVP Engenharia de dados

Análise do tráfego de veículos

Aluno: Kelmo Siqueira dos Anjos
Pós-graduação em Ciência de Dados e Analytics
MVP da Sprint Engenharia de dados



Sumário

- 1- Objetivo
- 2- Base de dados
- 3- Solução escolhida
- 4- Coleta de dados
- 5- Catalogo de dados
- 6- Carga e modelagem de dados
- 7- Análise de dados
- 8- Autoavaliação
- 9- Conclusão

1- Objetivo

O gerenciamento do tráfego de veículos em ambientes urbanos representa um grande desafio para os órgãos de trânsito das cidades. A análise do trânsito a partir de informações de tráfego possibilita um melhor planejamento da utilização das vias públicas, gerenciamento de congestionamentos e auxilia na elaboração de políticas públicas de trânsito.

Este projeto tem como objetivo realizar uma análise das informações sobre o quantitativo e tipos de veículos identificados em uma via pública. Com essas informações será possível realizar um melhor gerenciamento do trânsito em uma cidade e o planejamento de ações, tais como:

- Restrição de tráfego de veículos (Ex: caminhões) em determinados horários;
- Criação de corredores de ônibus;
- Criação de ciclovias, entre outras ações.

Diante desse cenário, surgiram os seguintes questionamentos:

A) Quais são os horários de maior e menor tráfego de veículos no período de avaliação?

B) Quais são os dias da semana de maior e menor tráfego de veículos no período de avaliação?

C) No horário de maior movimento, qual é a quantidade média de cada tipo de veículo?

D) Qual é a quantidade média do tráfego de bicicletas nos horários coletados?

2- Base de dados

O dataset escolhido para o projeto foi o “**Traffic Prediction Dataset**” obtido a partir do site **Kaggle** (<https://www.kaggle.com/datasets/hasibullahaman/traffic-prediction-dataset>), este dataset foi criado a partir de informações que foram obtidas por um modelo de visão computacional que detecta quatro tipos de veículos: carros, bicicletas, ônibus e caminhões.

O dataset possui dados referentes a coleta de informações a cada 15 minutos da quantidade e tipo de veículos em uma via pública, no período de 1 (um) mês. A cada coleta os valores são zerados para iniciar uma nova contagem de veículos.

3- Solução escolhida

Para realizar a análise dos dados do dataset fazendo uso de uma metodologia **ETL** (*Extract Transform Load*) foi escolhida a plataforma de “**Big Data**” em nuvem “**Databricks**”.

Foi utilizada a versão **Databricks Community** que consiste em uma versão gratuita que possibilita o uso de micro-clusters spark, notebooks que proporcionam o uso de diversas linguagens (Ex: Python) e o Apache spark que consiste em um mecanismo de análise de código aberto para processamento de dados em grande escala (Big Data).

A versão **Databricks Community** possui algumas limitações de uso, tais como o uso da máquina virtual (compute) que compõem o cluster que após um período de tempo sem uso a mesma é desligada.

4- Coleta dos dados

Inicialmente foi realizado o download do arquivo “**Traffic.csv**” referente ao dataset “**Traffic prediction dataset**” disponibilizado no site Kaggle, referente a coleta de informações da quantidade e tipo de veículos em uma via pública, no período de 1 (um) mês.

Em seguida, foi realizado o upload do arquivo para a plataforma “**Databricks Community**” na área de **DBFS** (*Databricks File System*) que consiste em um sistema de arquivos distribuído em nuvem que possibilita o armazenamento de arquivos de forma persistente. Ou seja, não sejam apagados mesmos que a máquina virtual utilizada no cluster do Databricks Community seja desligada.

Para isso, foram realizados os seguintes procedimentos:

- Por padrão, o recurso de “**DBFS file Browser**” é desabilitado ao ser criada uma nova conta no Databricks Community. Para habilitar esse recurso, clicar na conta no canto superior direito e selecionar a opção “**Settings**”;
- Selecionar “**Advanced**” e habilitar o recurso “**DBFS file Browser**” (Figura 1);

DBFS File Browser
Enable or disable DBFS File Browser

On 

Figura 1

- Para acessar o “**DBFS file Browser**” clicar no menu à esquerda em “**Catalog**” e selecionar a aba “**DBFS**” (Figura 2);

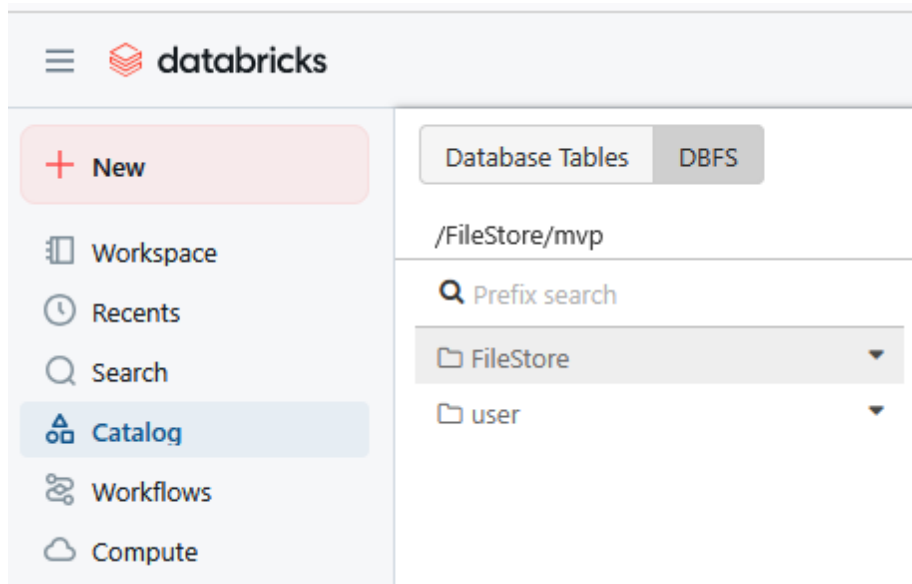


Figura 2

- Por questões de estruturação do armazenamento dos dados, foi criado o um diretório “**mvp**” para armazenar o arquivo “**Traffic.csv**”. Para isso, foi selecionado “**FileStore**” e na tela disponibilizada à direita foi realizado um clique com botão direito e selecionada a opção “**create folder**”;
- Em seguida, selecionar o diretório “**mvp**” e clicar no botão “**Upload**” para transferir o arquivo “**Traffic.csv**” para a estrutura do **DBFS** (Figura 3);

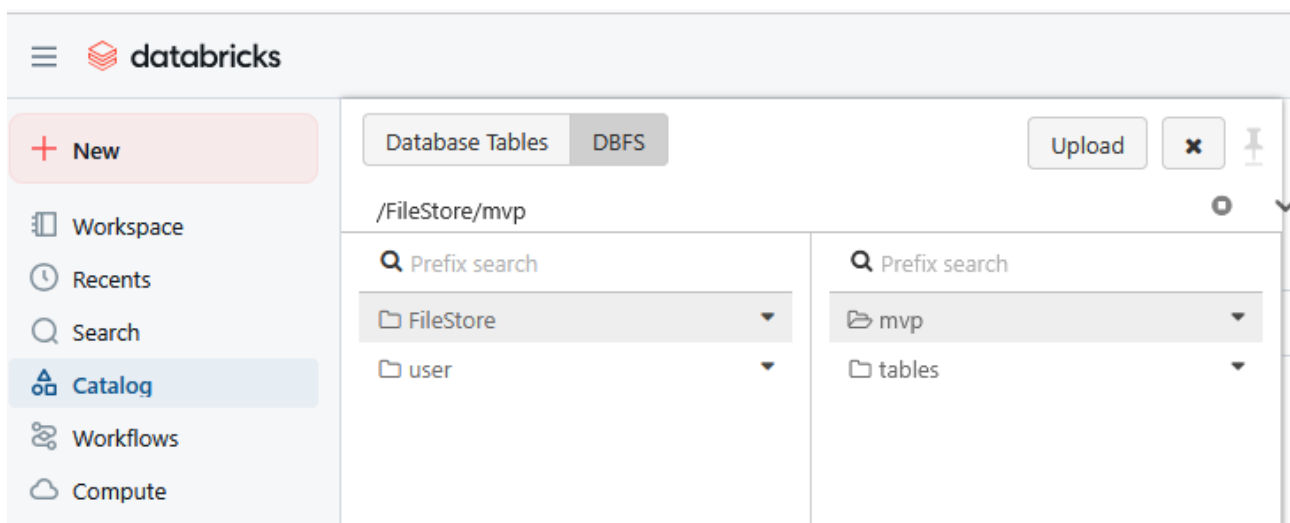


Figura 3

- Na tela “**Upload Data to DBFS**” clicar na área do texto “**Drop files to upload, or click to browse**” para ser direcionado a tela que possibilita selecionar o arquivo que deseja fazer o upload (Figura 4). Para esse projeto foi realizado o upload do arquivo “**Traffic.csv**”;

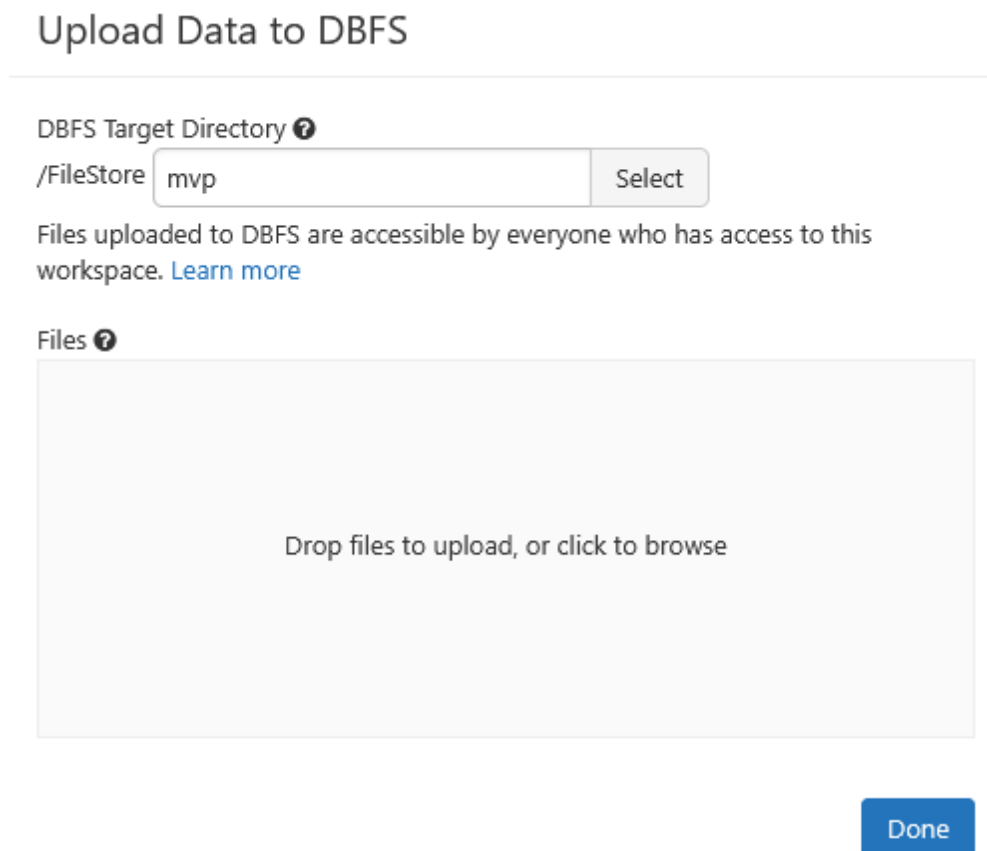


Figura 4

- Ao término do upload o arquivo “**Traffic.csv**” estará disponível no DBFS do Databricks Community (Figura 5);

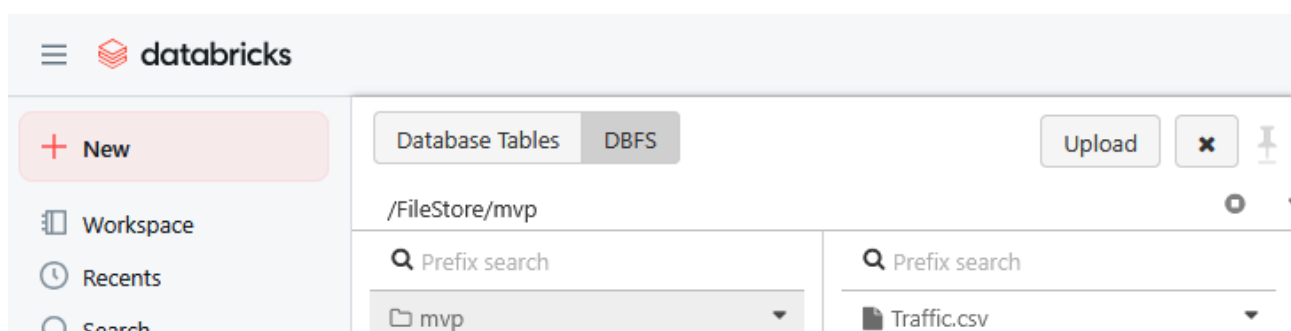


Figura 5

5- Catalogo de dados

O dataset utilizado no projeto apresenta **2976 registros e 9 atributos**, os dados foram disponibilizados no formato CSV e apresenta as seguintes colunas:

- **Time**
 - Consiste em um dado do **tipo string**;
 - Representa a hora da coleta dos dados de tráfego de veículos, apresentando as seguintes características:
 - O dado é composto da hora da coleta e da informação “AM” ou “PM”. Por exemplo: “1:00:00 AM”;
 - Para o atributo “Hora”, os valores representam a hora do dia que foi realizada a coleta dos dados. Como a coleta é realizada a cada 15 minutos, e um dia possui 1440 minutos ocorreram durante o período de um dia 96 coletas (1440 min / 15 min);
 - Durante o período de um mês, os horários das 96 coletas realizadas diariamente foram os mesmos nos dias durante esse período;
- **Date**
 - Consiste em um do **tipo inteiro**;
 - Representa a data (apenas o dia, não tem a informação do mês e ano) da coleta dos dados de tráfego de veículos, contendo o dia do mês que foi realizada a coleta.
 - **Os valores dessa coluna podem estar entre 1 e 31;**
- **Day of the week**
 - Consiste em um dado do **tipo string**;
 - Representa o dia da semana que foi realizada a coleta dos dados de tráfego de veículos.
 - Esse campo poderá assumir os seguintes valores: **"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"**;
- **CarCount**
 - Consiste em um dado do **tipo inteiro**;
 - Representa a quantidade de carros contabilizados desde a última coleta realizada, ou seja, durante o período de 15 minutos;

- **BikeCount**

- Consiste em um dado do **tipo inteiro**;
- Representa a quantidade de bicicletas contabilizados desde a última coleta realizada, ou seja, durante o período de 15 minutos;

- **BusCount**

- Consiste em um dado do **tipo inteiro**;
- Representa a quantidade de ônibus contabilizados desde a última coleta realizada, ou seja, durante o período de 15 minutos;

- **TruckCount**

- Consiste em um dado do **tipo inteiro**;
- Representa a quantidade de caminhões contabilizados desde a última coleta realizada, ou seja, durante o período de 15 minutos;

- **Total:**

- Consiste em um dado do **tipo inteiro**;
- Representa o total de veículos (independente do tipo) que trafegaram desde a última coleta realizada;
- **Corresponde a soma dos dados das colunas CarCount, BikeCount, BusCount e TruckCount;**

- **Traffic Situation**

- Consiste em um dado do **tipo string**;
- Representa a situação do tráfego no momento da coleta, podendo assumir as seguintes informações:
 - **Low**, tráfego baixo de veículos;
 - **Normal**, tráfego normal de veículos;
 - **Heavy**, tráfego mais intenso de veículos;
 - **High**, tráfego alto de veículos;

6- Carga e Modelagem dos dados

Para o processo de modelagem e análise dos dados foi criado um notebook na plataforma “**Databricks Community**” que será executado no micro-cluster disponibilizado na plataforma.

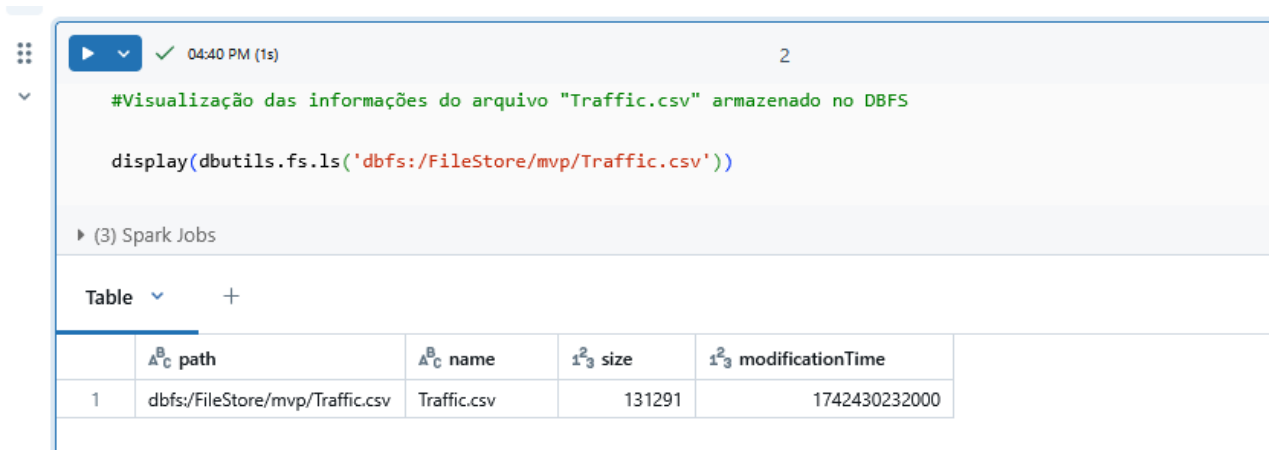
No notebook, inicialmente foi realizada a importação das bibliotecas “**pandas**” e “**numpy**” que irão auxiliar no processo de modelagem dos dados.

```
import pandas as pd
import numpy as np
```

Para verificar a presença do arquivo “**Traffic.csv**” no DBFS do “**Databricks Community**” foi utilizado o comando abaixo para disponibilizar as informações de localização, tamanho e última modificação do arquivo

Sendo possível confirmar que o arquivo se encontra em “**dbfs:/Filestore/mvp/Traffic.csv**” e o seu tamanho (Figura 6).

```
display(dbutils.fs.ls('dbfs:/FileStore/mvp/Traffic.csv'))
```



The screenshot shows a Databricks notebook interface. At the top, there's a status bar indicating the execution time is 04:40 PM (1s) and the job ID is 2. Below the status bar, the code cell contains a comment in Portuguese: "#Visualização das informações do arquivo 'Traffic.csv' armazenado no DBFS" followed by the command `display(dbutils.fs.ls('dbfs:/FileStore/mvp/Traffic.csv'))`. Below the code cell, there's a section for Spark Jobs, showing (3) Spark Jobs. Below that, there's a table view of the output. The table has 5 columns: path, name, size, and modificationTime. The table contains one row with the following data: path: dbfs:/FileStore/mvp/Traffic.csv, name: Traffic.csv, size: 131291, and modificationTime: 1742430232000.

	path	name	size	modificationTime
1	dbfs:/FileStore/mvp/Traffic.csv	Traffic.csv	131291	1742430232000

Figura 6

Para criação de um dataframe “**df_traffic**” contendo as informações do arquivo “**Traffic.csv**” foi utilizado o comando abaixo, sendo possível visualizar o nome das colunas e tipo de dado que estão disponíveis na coluna (Figura 7).

```
df_traffic = spark.read.option('inferSchema', True).csv('dbfs:/FileStore/mvp/Traffic.csv',
header=True)
```


The screenshot shows a Jupyter Notebook cell with the following code:

```
#Criação do dataframe "df_traffic"

df_traffic = spark.read.option('inferSchema', True).csv('dbfs:/FileStore/mvp/Traffic.csv', header=True)
```

Below the code, the output shows the Spark job status and the schema of the created DataFrame:

```
(2) Spark Jobs

df_traffic: pyspark.sql.dataframe.DataFrame
  Time: string
  Date: integer
  Day of the week: string
  CarCount: integer
  BikeCount: integer
  BusCount: integer
  TruckCount: integer
  Total: integer
  Traffic Situation: string
```

Figura 7

O dataset “**Traffic.csv**” apresenta uma estrutura de dados que necessita fazer uso apenas de uma tabela de dados sem a necessidade de relacionamentos, caracterizado um modelo do tipo “**Flat**”.

Em uma abordagem de melhores práticas, nos projetos de Big Data recomenda-se o uso de camadas de armazenamento de dados que são utilizadas para melhorar a qualidade e confiabilidade dos dados, são elas:

- **Bronze**, armazena os dados originais (bruto) sem tratamento;
- **Prata**, armazena dados que passaram por algum refinamento;
- **Gold**, armazena dados prontos para análise e processamento;

Para seguir as melhores práticas em projetos de Big Data, foi criada uma database “**bronze**” para receber as tabelas que conterão os dados sem tratamento do dataframe “**df_traffic**”. Para isso, foram utilizados os seguintes comandos:

```
%sql DROP DATABASE bronze;
%sql CREATE DATABASE bronze;
```

O comando “**DROP DATABASE...**” foi utilizado para caso tenha alguma database existente com o nome “**bronze**” ela seja removida para que se possa iniciar um novo projeto. O comando “**CREATE DATABASE...**” foi utilizado para criar uma nova database “**bronze**” (Figura 8).

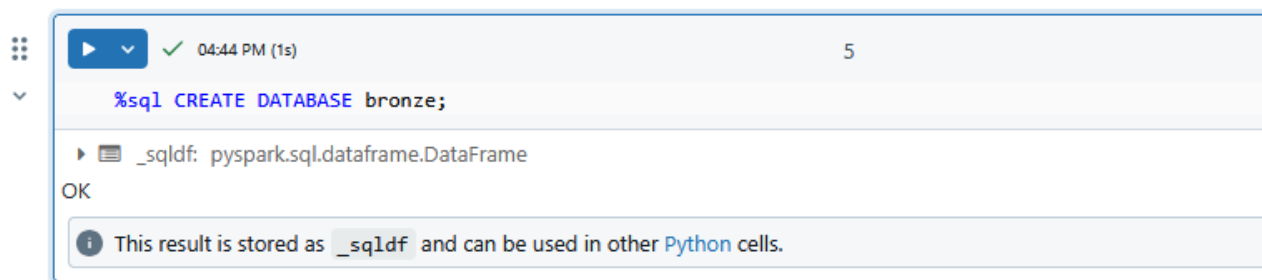


Figura 8

Ao visualizar as informações dos nomes dos campos após a criação do dataframe “df_traffic” foi identificado que as colunas “Day of the week” e “Traffic Situation” possuem espaços em branco. Essa situação gera erro no processo de criação de tabela no SQL, sendo necessário alterar o nome das colunas antes de criar a tabela na database “bronze”.

Para isso, foram utilizados os seguintes comandos para modificar o nome da coluna “Day of the week” para “Dayweek” e o nome da coluna “Traffic Situation” para “Trafficsit” (Figura 9):

```
df_traffic = df_traffic.withColumnRenamed("Day of the week", "Dayweek")
df_traffic = df_traffic.withColumnRenamed("Traffic Situation", "Trafficsit")
```

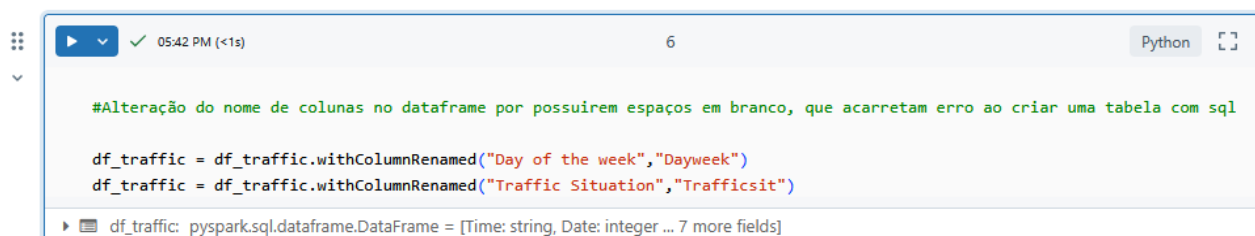


Figura 9

Após os ajustes nos nomes das colunas no dataframe, foi possível criar a tabela “bronze.Traffic” fazendo uso do seguinte comando (Figura 10):

```
df_traffic.write.format("delta").mode("overwrite").saveAsTable('bronze.Traffic')
```

```

▶ 2 minutes ago (5s) 7

#Criação da tabela "bronze.Traffic"

df_traffic.write.format("delta").mode("overwrite").saveAsTable('bronze.Traffic')

▶ (6) Spark Jobs

```

Figura 10

Para visualizar o conteúdo da tabela “**bronze.Traffic**” foi utilizada a seguinte consulta SQL (Figura 11), nesse etapa, não foi identificada a necessidade de ajustes iniciais na estrutura dos dados.

```

spark.sql("""
SELECT *
FROM bronze.Traffic
""").show()

```

```

▶ 05:55 PM (2s) 8

#Visualização do conteúdo da tabela "bronze.Traffic"

spark.sql("""
SELECT *
FROM bronze.Traffic
""").show()

▶ (2) Spark Jobs

```

Time	Date	Dayweek	CarCount	BikeCount	BusCount	TruckCount	Total	TrafficSit
12:00:00 AM	10	Tuesday	31	0	4	4	39	low
12:15:00 AM	10	Tuesday	49	0	3	3	55	low
12:30:00 AM	10	Tuesday	46	0	3	6	55	low
12:45:00 AM	10	Tuesday	51	0	2	5	58	low
1:00:00 AM	10	Tuesday	57	6	15	16	94	normal
1:15:00 AM	10	Tuesday	44	0	5	4	53	low

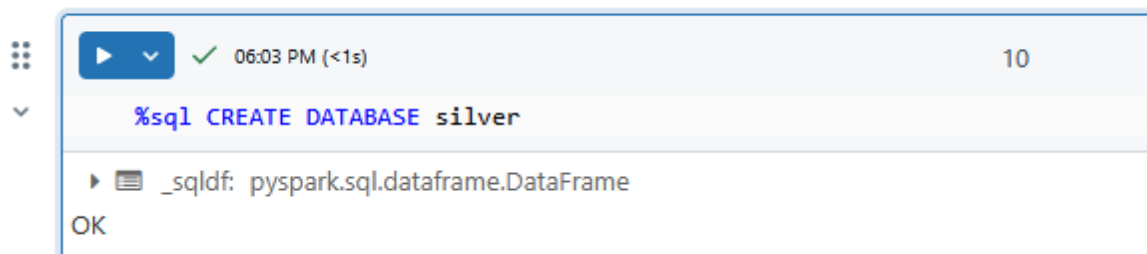
Figura 11

Seguindo as melhores práticas em projetos de Big Data, foi criada uma database “**silver**” para conter as tabelas com os dados que tiveram algum tratamento do dataframe “**df_traffic**”. Para isso, foram utilizados os seguintes comandos (Figura 12):

```

%sql DROP DATABASE silver CASCADE
%sql CREATE DATABASE silver

```



The screenshot shows a Databricks console interface. At the top, there is a status bar with a play button, a checkmark, the time '06:03 PM (<1s)', and the number '10'. Below this, the command '%sql CREATE DATABASE silver' is displayed. Underneath the command, the output shows '_sqlctx: pyspark.sql.dataframe.DataFrame'. At the bottom, there is an 'OK' button.

```
%sql CREATE DATABASE silver
```

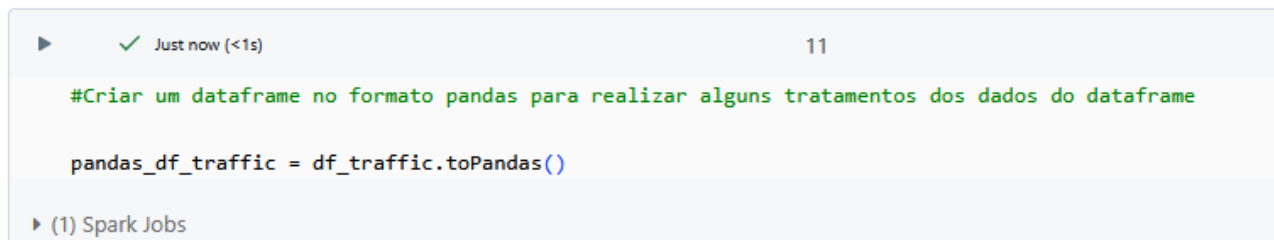
_sqlctx: pyspark.sql.dataframe.DataFrame

OK

Figura 12

Para realização de alguns tratamentos dos dados no dataframe “**df_traffic**”, será necessário criar o dataframe “**pandas_df_traffic**” no formato “**pandas**”. Para isso, foi utilizado o seguinte comando (Figura 13):

```
pandas_df_traffic = df_traffic.toPandas()
```



The screenshot shows a Databricks console interface. At the top, there is a status bar with a play button, a checkmark, the time 'Just now (<1s)', and the number '11'. Below this, the command '#Criar um dataframe no formato pandas para realizar alguns tratamentos dos dados do dataframe' is displayed. Underneath the command, the output shows 'pandas_df_traffic = df_traffic.toPandas()'. At the bottom, there is a '(1) Spark Jobs' button.

```
#Criar um dataframe no formato pandas para realizar alguns tratamentos dos dados do dataframe
```

```
pandas_df_traffic = df_traffic.toPandas()
```

(1) Spark Jobs

Figura 13

Para verificar a existência de valores inválidos nas colunas do dataframe “**pandas_df_traffic**” foi utilizado o comando abaixo. Onde pode ser observado que foram identificadas 0 (zero) ocorrências dessa situação nas colunas (Figura 14):

```
pandas_df_traffic.isna().sum()
```

```
▶ ✓ 2 minutes ago (<1s) 12

#Verificar a existência de valores inválidos nas colunas do dataframe

pandas_df_traffic.isna().sum()

Out[30]: Time      0
Date      0
Dayweek    0
CarCount   0
BikeCount  0
BusCount   0
TruckCount 0
Total      0
Trafficsit 0
dtype: int64
```

Figura 14

Para verificar a existência de valores nulos (ou ausência de valor) nas colunas do dataframe “**pandas_df_traffic**” foi utilizado o comando abaixo. Onde pode ser observado que foram identificadas 0 (zero) ocorrências dessa situação nas colunas (Figura 15).

```
pandas_df_traffic.isnull().sum()
```

```
⋮ ▶ ✓ 06:33 PM (<1s) 13
▼

#Verificar a existência de valores nulos (ou ausência de valor) nas colunas do dataframe

pandas_df_traffic.isnull().sum()

Out[31]: Time      0
Date      0
Dayweek    0
CarCount   0
BikeCount  0
BusCount   0
TruckCount 0
Total      0
Trafficsit 0
dtype: int64
```

Figura 15

Para avaliar a necessidade de novos ajustes nos dados, foi criada a tabela “**silver.Traffic**” fazendo uso do seguinte comando (Figura 16):

```
df_traffic.write.format("delta").mode("overwrite").saveAsTable('silver.Traffic')
```



Figura 16

Para visualizar o conteúdo da tabela “**silver.Traffic**” foi utilizada a seguinte consulta SQL (Figura 17), nesse etapa, não foi identificada a necessidade de ajustes iniciais na estrutura dos dados:

```
spark.sql("""
SELECT *
FROM silver.Traffic
""").show()
```

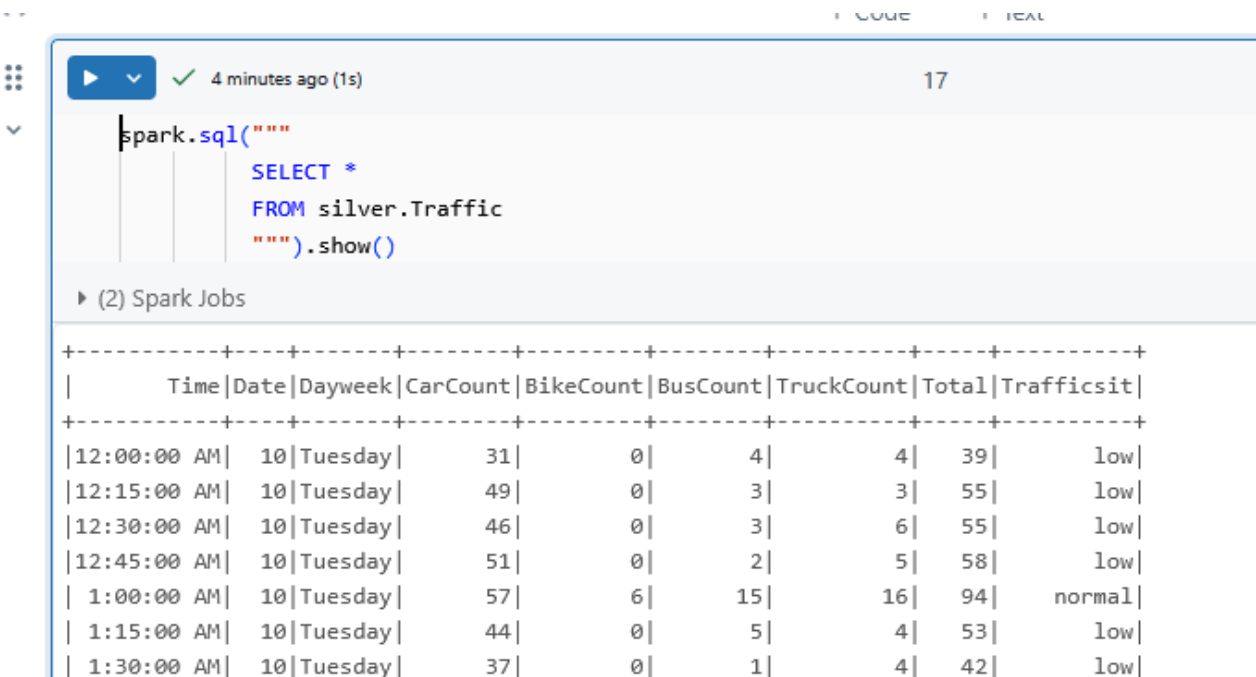


Figura 17

Após análise dos objetivos referentes aos questionamentos que serão identificados, foi identificado que as respostas dos mesmos não teria necessidade da coluna “**Trafficsit**”. Devido a isso, essa coluna será retirada do dataframe para diminuir o volume de informações que será analisado.

Para isso, será necessário criar o dataframe “**pandas_df_traffic_silver**” no formato “**pandas**”. Sendo utilizado o seguinte comando (Figura 18):

```
pandas_df_traffic_silver = df_traffic.toPandas()
```

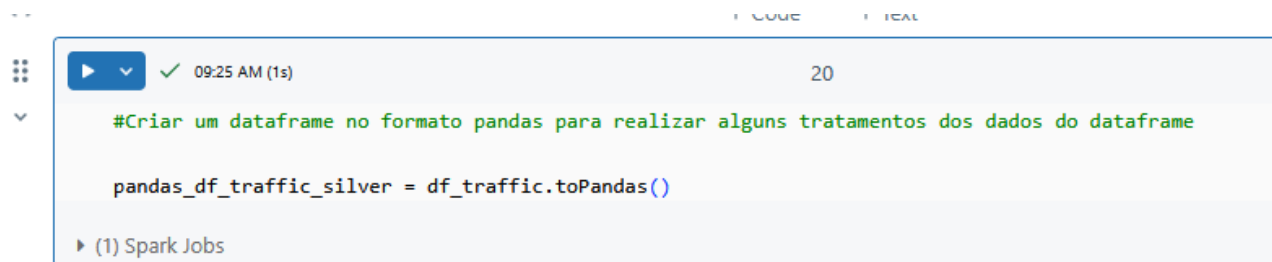


Figura 18

Em seguida, foi realizada a remoção do coluna “**Trafficsit**” a partir do seguinte comando (Figura 19).

```
pandas_df_traffic_silver = pandas_df_traffic_silver.drop(columns=['Trafficsit'])
```

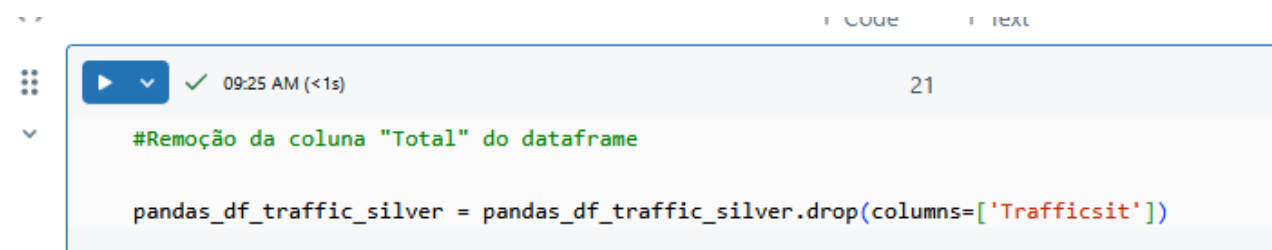
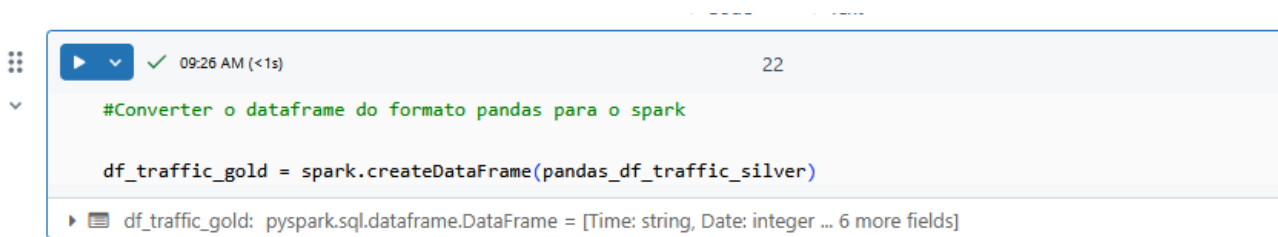


Figura 19

Após a remoção da coluna, foi criado um novo dataframe no formato spark “**df_traffic_gold**” a partir do dataframe “**pandas_df_traffic_silver**” no formato pandas que não possui a coluna “**Trafficsit**” (Figura 20).

```
df_traffic_gold = spark.createDataFrame(pandas_df_traffic_silver)
```



```
#Converter o dataframe do formato pandas para o spark

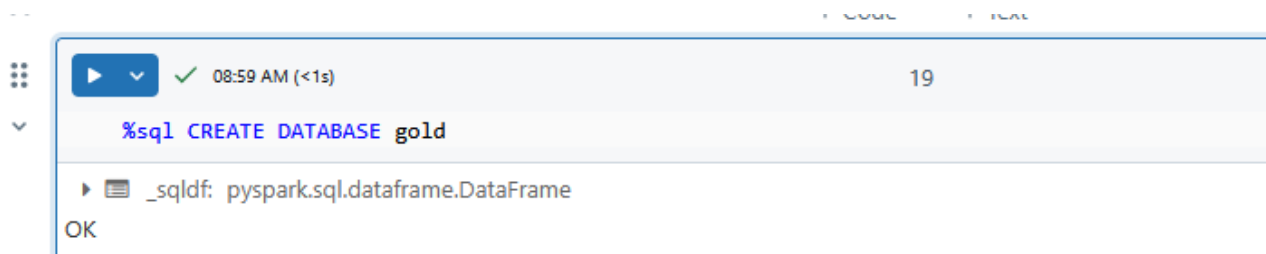
df_traffic_gold = spark.createDataFrame(pandas_df_traffic_silver)
```

df_traffic_gold: pyspark.sql.dataframe.DataFrame = [Time: string, Date: integer ... 6 more fields]

Figura 20

Seguindo as melhores práticas em projetos de Big Data, foi criada uma database “**gold**” para ser utilizada na análise dos dados do dataframe “**df_traffic_gold**”. Para isso, foram utilizados os seguintes comandos (Figura 21):

```
%sql DROP DATABASE gold CASCADE
%sql CREATE DATABASE gold
```



```
%sql CREATE DATABASE gold
```


_sqldf: pyspark.sql.dataframe.DataFrame

OK

Figura 21

Para realizar a análise dos dados, foi criada a tabela “**gold.Traffic**” fazendo uso do seguinte comando (Figura 22):

```
df_traffic_gold.write.format("delta").mode("overwrite").saveAsTable('gold.Traffic')
```



```
#Criação da tabela "gold.Traffic"

df_traffic_gold.write.format("delta").mode("overwrite").saveAsTable('gold.Traffic')
```

(6) Spark Jobs

Figura 22

Para visualizar o conteúdo da tabela “**gold.Traffic**” foi utilizada a seguinte consulta SQL (Figura 23), onde é possível visualizar que a coluna “**Trafficsit**” não está mais na tabela:

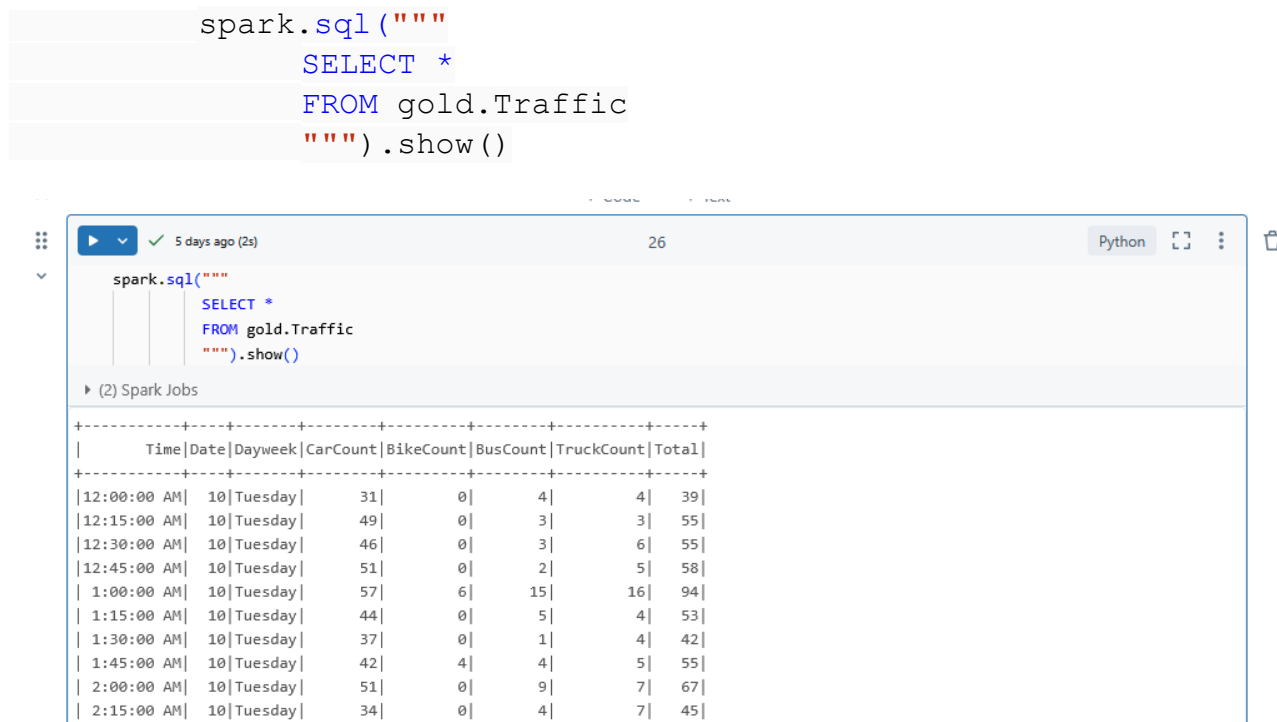


Figura 23

7- Análise dos dados

Este projeto tem como objetivo realizar uma análise das características de tráfego de veículos em um via pública, para possibilitar um melhor planejamento e gerenciamento do trânsito.

Com o intuito de auxiliar essa análise, foram realizados alguns questionamentos. As respostas destes possibilitarão realizar um melhor planejamento do trânsito. São eles:

A) Quais são os horários de maior e menor tráfego de veículos no período de avaliação?

Para obtenção dessa resposta, foi criada a view “**Media_veiculos_por_horario**” que tem como objetivo obter a média do total de veículos contabilizados em cada horário de coleta durante o período de avaliação (1 [um] mês).

Como explicado anteriormente, por dia ocorreram 96 coletas de dados em horários específicos e esses horários serão os mesmos nos dias avaliados.

Os dados utilizados foram o da tabela “**gold.traffic**” e os valores da média do total de veículos por horário foi calculada a partir do campo “**Total**”. Para uma melhor visualização da resposta, a consulta SQL foi configurada para fornecer o resultado arredondado em duas casas decimais “**round(avg(Total),2)**”.

Segue a abaixo os comandos SQL utilizados para criação da view “**Media_veiculos_por_horario**” e o resultado de sua execução (Figura 24).

```
spark.sql("""
CREATE VIEW Media_veiculos_por_horario AS
select Time as Horario, round(avg(Total),2) as Media_veiculos
from gold.Traffic
group by Time
order by Horario desc
""").show()
```

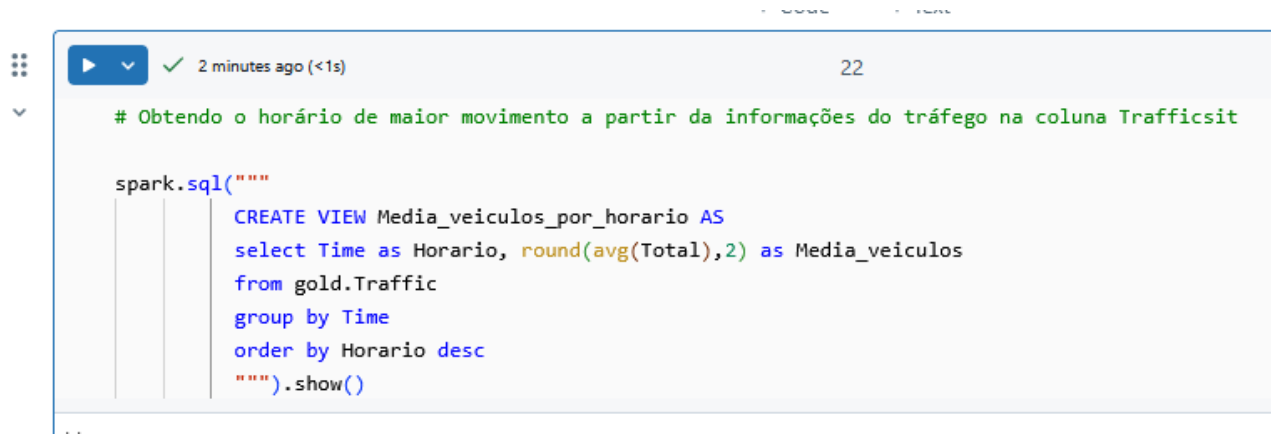


Figura 24

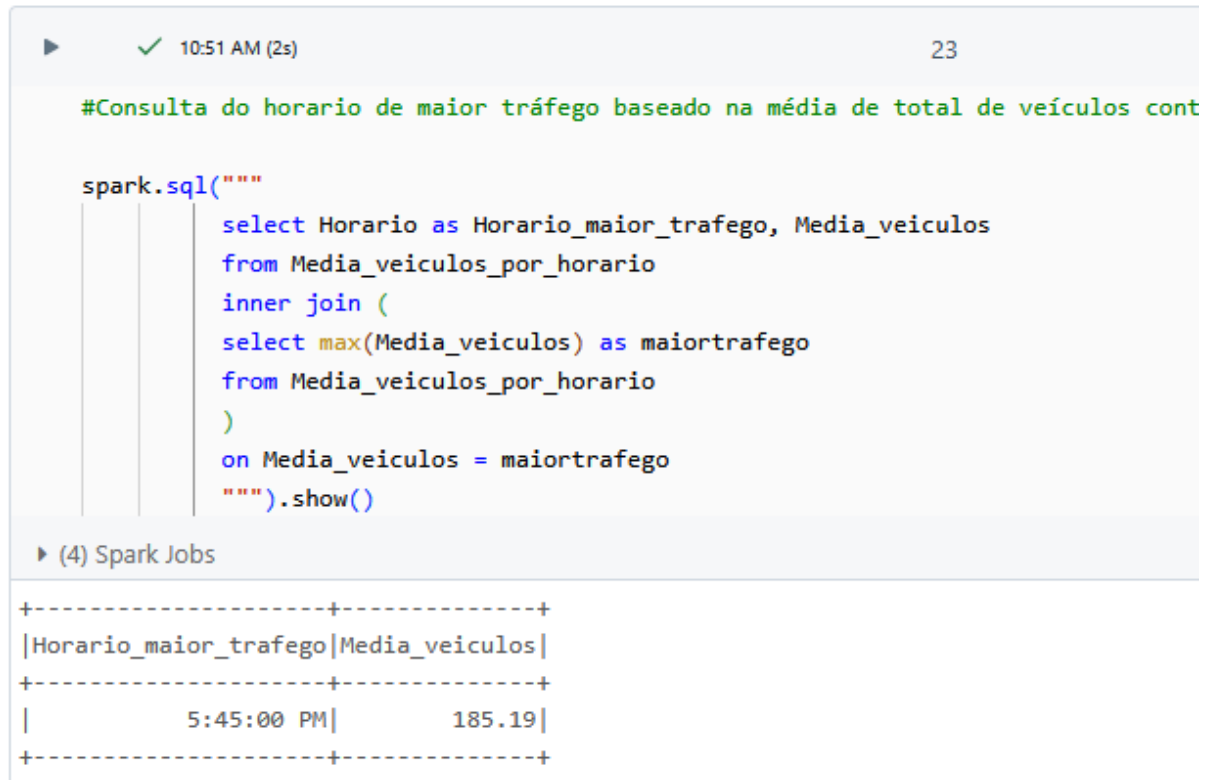
A partir da view “**Media_veiculos_por_horario**” foi possível realizar uma consulta SQL para obter o horário que apresentou o maior movimento de veículos. Para isso, foi inicialmente obtido o maior valor na coluna “**Media_veiculos**” com o parâmetro “**max(Media_veiculos)**” e em seguida identificado o horário que apresentava esse valor avaliando a condição “**Media_veiculos = maiortrafego**”.

Segue abaixo a consulta SQL que foi utilizada para obter essa informação.

```
spark.sql("""
select Horario as Horario_maior_trafego, Media_veiculos
from Media_veiculos_por_horario
inner join (
select max(Media_veiculos) as maiortrafego
from Media_veiculos_por_horario
)
on Media_veiculos = maiortrafego
""").show()
```

Após a execução da consulta SQL, foi possível identificar que a partir dos dados contidos no dataset analisado o horário de maior tráfego identificado foi de “**5:45:00 PM**” e apresentou uma média total de “**185,19**” veículos no mês avaliado (Figura 25).

Com essa informação, é possível o órgão de trânsito da localidade identificar o horário que necessita adotar mais ações para minimizar o impacto do volume de veículos no trânsito nessa área.



The screenshot shows a Databricks notebook interface. At the top, there is a status bar with a green checkmark, the time '10:51 AM (2s)', and the page number '23'. Below this, a comment in green text reads: '#Consulta do horario de maior tráfego baseado na média de total de veículos cont'. The main code block is a Spark SQL query enclosed in triple quotes, which finds the hour with the maximum average number of vehicles. The query is as follows:

```
spark.sql("""
select Horario as Horario_maior_trafego, Media_veiculos
from Media_veiculos_por_horario
inner join (
select max(Media_veiculos) as maiortrafego
from Media_veiculos_por_horario
)
on Media_veiculos = maiortrafego
""").show()
```

Below the code, a section titled '(4) Spark Jobs' is visible. The results of the query are displayed in a table format with dashed lines as separators. The table has two columns: 'Horario_maior_trafego' and 'Media_veiculos'. The single row of data shows '5:45:00 PM' and '185.19'.

Horario_maior_trafego	Media_veiculos
5:45:00 PM	185.19

Figura 25

Para obter o horário de menor movimento utilizou-se uma consulta SQL semelhante a utilizada para obter o horário de maior movimento.

A diferença está que para obter o menor valor na coluna “**Media_veiculos**” foi utilizado o parâmetro “**min(Media_veiculos)**” e em seguida identificado o horário que apresentava esse valor avaliando a condição “**Media_veiculos = menortrafego**”.

Segue abaixo a consulta SQL que foi utilizada para obter essa informação.

```
spark.sql("""
select Horario as Horario_menor_trafego, Media_veiculos
from Media_veiculos_por_horario
inner join (
select min(Media_veiculos) as menortrafego
from Media_veiculos_por_horario
)
on Media_veiculos = menortrafego
""").show()
```

Após a sua execução, foi possível identificar que a partir dos dados contidos no dataset analisado o horário de menor tráfego identificado foi de “11:15:00 PM” e apresentou uma média total de “39,55” veículos no mês avaliado (Figura 26).

Com essa informação, é possível o órgão de trânsito da localidade identificar o horário que não será necessário ter ações adicionais devido ao baixo volume de veículos. Possibilitando direcionar os seus recursos para outra área da localidade que tenha mais necessidade.

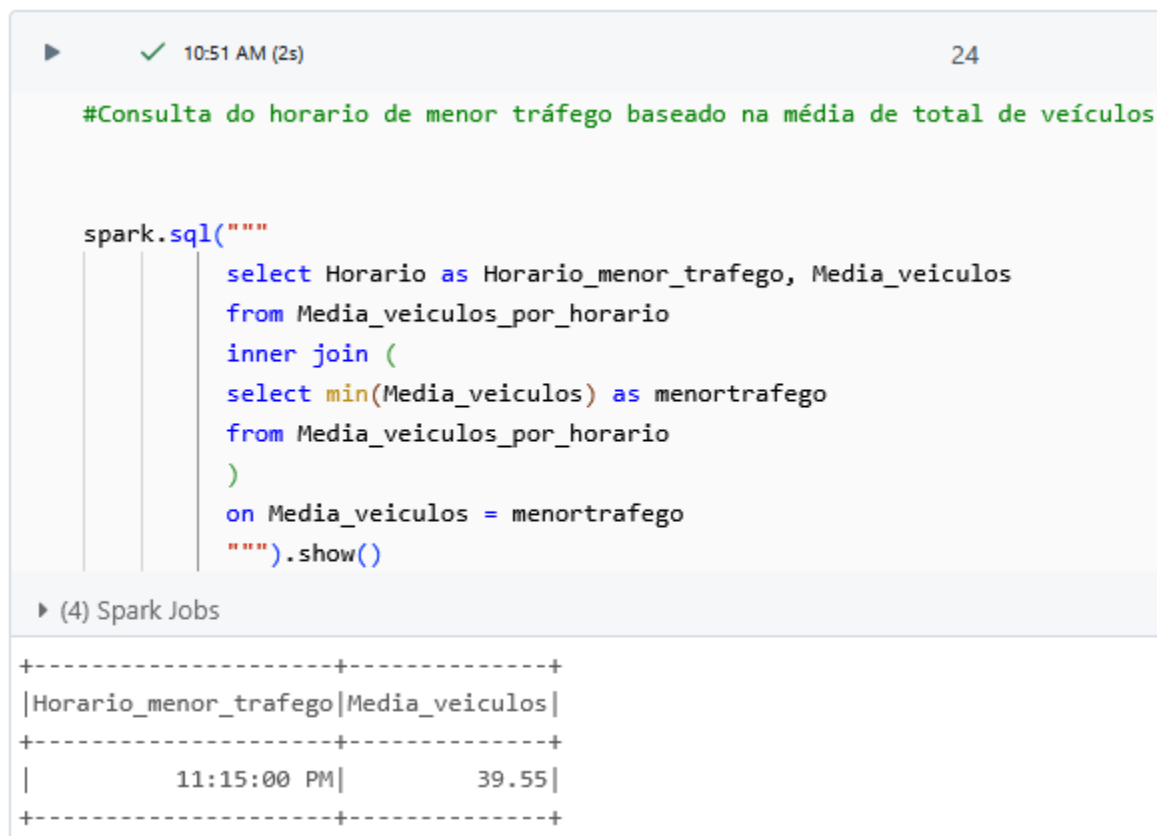


Figura 26

B) Quais são os dias da semana de maior e menor tráfego de veículos no período de avaliação?

Para obtenção dessa resposta, foi criada a view “Media_veiculos_por_dia_semana” que tem como objetivo obter a média de totais de veículos contabilizados em cada dia da semana durante o período de avaliação (1 [um] mês).

Os dados utilizados foram o da tabela “gold.traffic” e os valores da média de totais de veículos por dia da semana foi calculada a partir do campo “Total”. Para uma melhor visualização da resposta, a consulta SQL foi configurada para fornecer o resultado arredondado em duas casas decimais “round(avg(Total),2)”.

Seguem a abaixo os comandos SQL utilizados para criação da view “**Media_veiculos_por_dia_semana**” e o resultado de sua execução (Figura 27).

```
spark.sql("""
    CREATE VIEW Media_veiculos_por_dia_semana AS
    select Dayweek as Dia_semana, round(avg(Total),2) as
Media_veiculos
    from gold.Traffic
    group by Dayweek
    order by Dayweek desc
""").show()
```

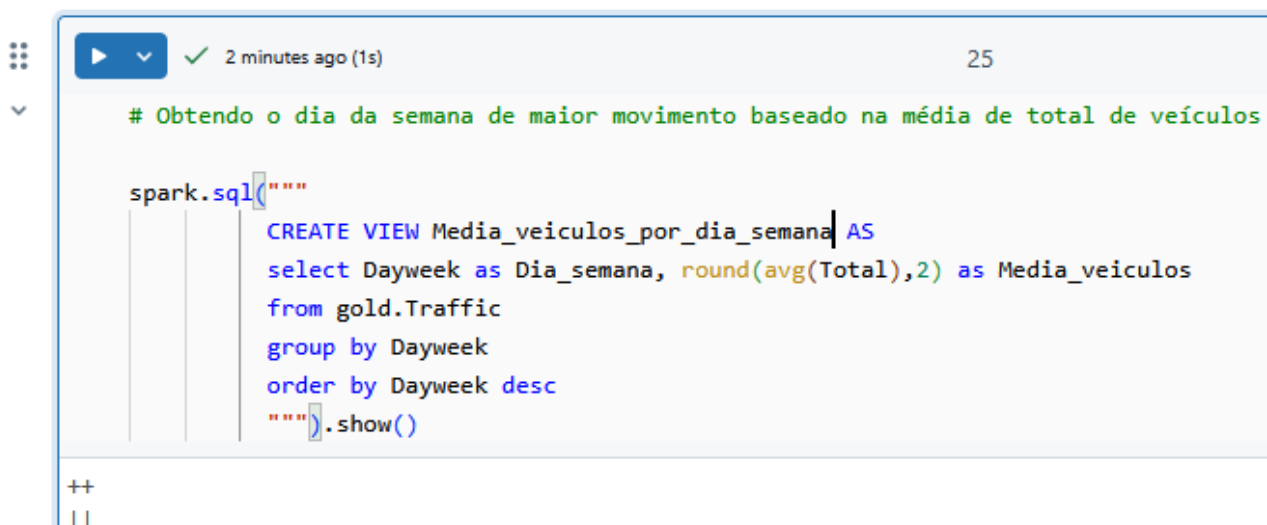


Figura 27

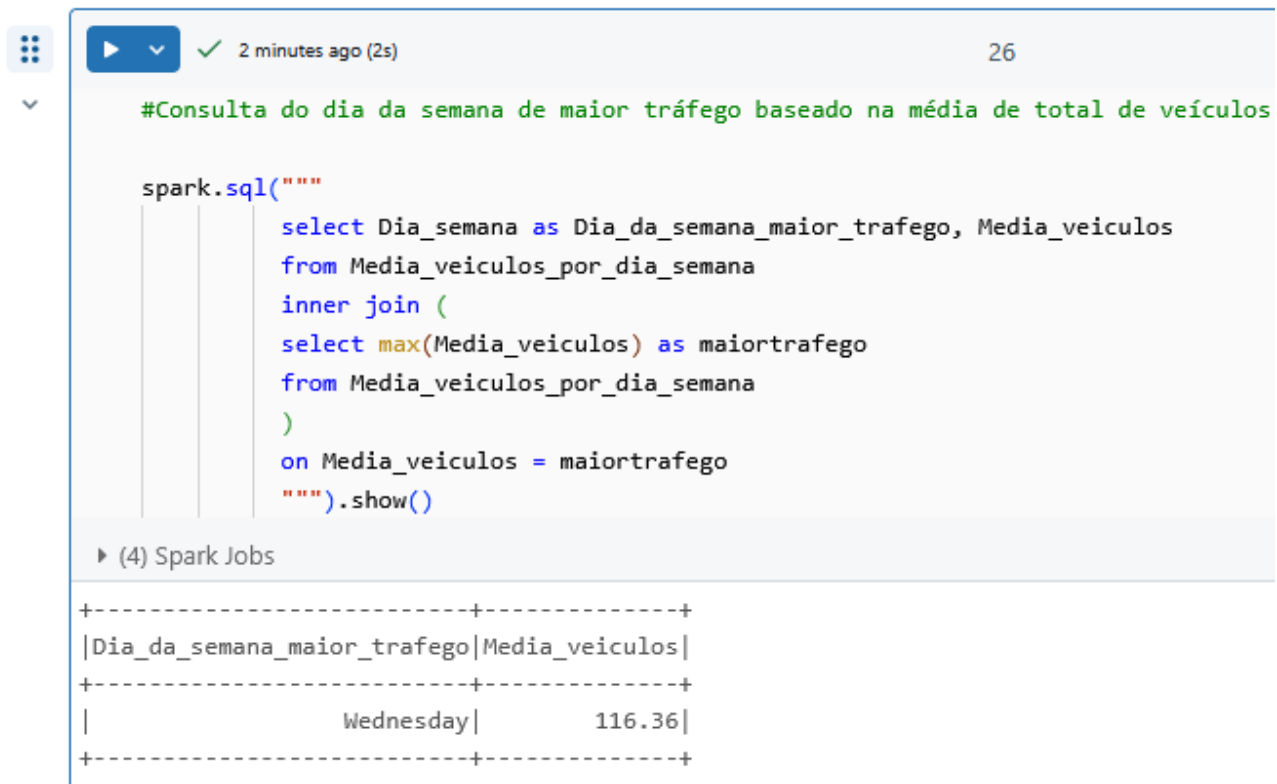
A partir da view “**Media_veiculos_por_dia_semana**” foi possível realizar uma consulta SQL para obter o dia da semana que apresentou o maior movimento de veículos. Para isso, foi inicialmente obtido o maior valor na coluna “**Media_veiculos**” com o parâmetro “**max(Media_veiculos)**” e em seguida identificado o horário que apresentava esse valor avaliando a condição “**Media_veiculos = maiortrafego**”.

Segue abaixo a consulta SQL que foi utilizada para obter essa informação.

```
spark.sql("""
    select Dia_semana as Dia_da_semana_maior_trafego, Media_veiculos
    from Media_veiculos_por_dia_semana
    inner join (
        select max(Media_veiculos) as maiortrafego
        from Media_veiculos_por_dia_semana
    )
    on Media_veiculos = maiortrafego
""").show()
```

Após a sua execução, foi possível identificar que a partir dos dados contidos no dataset analisado o horário de maior tráfego identificado foi “**Wedsneday**” (Quarta-feira) e apresentou uma média total de “**116,36**” veículos no mês avaliado (Figura 28).

Com essa informação, é possível o órgão de trânsito da localidade identificar o dia da semana que necessita adotar mais ações para minimizar o impacto do volume de veículos no trânsito nessa área.



The screenshot shows a Databricks notebook interface. At the top, there's a status bar with a play button, a checkmark, the text "2 minutes ago (2s)", and the number "26". Below this, a comment in green text reads: "#Consulta do dia da semana de maior tráfego baseado na média de total de veículos". The main code cell contains a Spark SQL query:

```
spark.sql("""
    select Dia_semana as Dia_da_semana_maior_trafego, Media_veiculos
    from Media_veiculos_por_dia_semana
    inner join (
        select max(Media_veiculos) as maiortrafego
        from Media_veiculos_por_dia_semana
    )
    on Media_veiculos = maiortrafego
""").show()
```

 Below the code, it says "► (4) Spark Jobs". The output is a table with two columns: "Dia_da_semana_maior_trafego" and "Media_veiculos". The table has one row with the values "Wednesday" and "116.36".

```
#Consulta do dia da semana de maior tráfego baseado na média de total de veículos

spark.sql("""
    select Dia_semana as Dia_da_semana_maior_trafego, Media_veiculos
    from Media_veiculos_por_dia_semana
    inner join (
        select max(Media_veiculos) as maiortrafego
        from Media_veiculos_por_dia_semana
    )
    on Media_veiculos = maiortrafego
""").show()
```

► (4) Spark Jobs

Dia_da_semana_maior_trafego	Media_veiculos
Wednesday	116.36

Figura 28

Para obter o dia da semana de menor movimento utilizou-se uma consulta SQL semelhante a utilizada para obter o dia da semana de maior movimento.

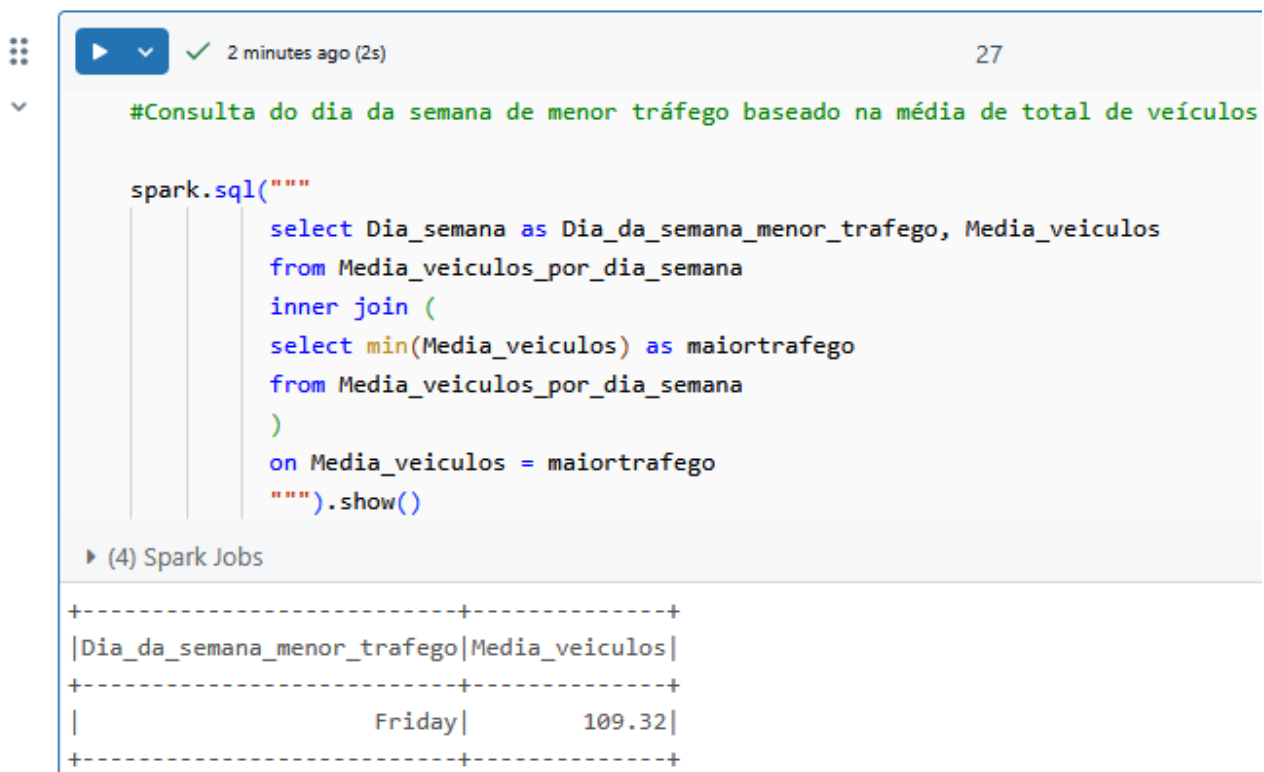
A diferença está que para obter o menor valor na coluna “**Media_veiculos**” foi utilizado o parâmetro “**min(Media_veiculos)**” e em seguida identificado o horário que apresentava esse valor avaliando a condição “**Media_veiculos = menortrafego**”.

Segue abaixo a consulta SQL que foi utilizada para obter essa informação.

```
spark.sql("""
    select Dia_semana as Dia_da_semana_menor_trafego, Media_veiculos
    from Media_veiculos_por_dia_semana
    inner join (
        select min(Media_veiculos) as maiortrafego
        from Media_veiculos_por_dia_semana
    )
    on Media_veiculos = maiortrafego
""").show()
```

Após a sua execução, foi possível identificar que a partir dos dados contidos no dataset analisado o dia da semana de menor tráfego identificado foi “**Friday**” (Sexta-feira) e apresentou uma média total de “**109,32**” veículos no mês avaliado (Figura 29).

Com essa informação, é possível o órgão de trânsito da localidade identificar o dia da semana que não será necessário ter ações adicionais devido ao baixo volume de veículos. Possibilitando direcionar os seus recursos para outra área da localidade que tenha mais necessidade.



The screenshot shows a Databricks SQL interface. At the top, there's a status bar with a play button, a checkmark, the text "2 minutes ago (2s)", and the number "27". Below this is a comment: "#Consulta do dia da semana de menor tráfego baseado na média de total de veículos". The main area contains a Spark SQL query:

```
spark.sql("""
    select Dia_semana as Dia_da_semana_menor_trafego, Media_veiculos
    from Media_veiculos_por_dia_semana
    inner join (
        select min(Media_veiculos) as maiortrafego
        from Media_veiculos_por_dia_semana
    )
    on Media_veiculos = maiortrafego
""").show()
```

Below the query, it says "► (4) Spark Jobs". The results are displayed in a table format:

Dia_da_semana_menor_trafego	Media_veiculos
Friday	109.32

Figura 29

C) No horário de maior movimento, qual é a quantidade média de cada tipo de veículo?

Para obtenção dessa resposta, foi inicialmente criada a view “**Movimento_tipo_veiculos_por_horario**” que tem como objetivo obter a média de tipos de veículos contabilizados em cada horário de coleta durante o período de avaliação (1 [um] mês).

Como explicado anteriormente, por dia ocorreram 96 coletas de dados em horários específicos e esses horários serão os mesmos nos dias avaliados.

Os dados utilizados foram da tabela “**gold.traffic**” e para cada coluna correspondente a um tipo de veículo foi realizado o cálculo da média da quantidade de veículos contabilizados para cada horário. Por exemplo: **avg(Carcount)** , média de valores contabilizados para carros.

Segue a abaixo os comandos SQL utilizados para criação da view “**Movimento_tipo_veiculos_por_horario**” e o resultado de sua execução (Figura 30).

```
spark.sql("""
CREATE VIEW Movimento_tipo_veiculos_por_horario AS
select Time as Horario, avg(CarCount) as Media_carro, avg(TruckCount) as
Media_caminhao, avg(BikeCount) as Media_bicicleta, avg(BusCount) as Media_onibus
from gold.Traffic
group by Horario
order by Horario
""").show()
```

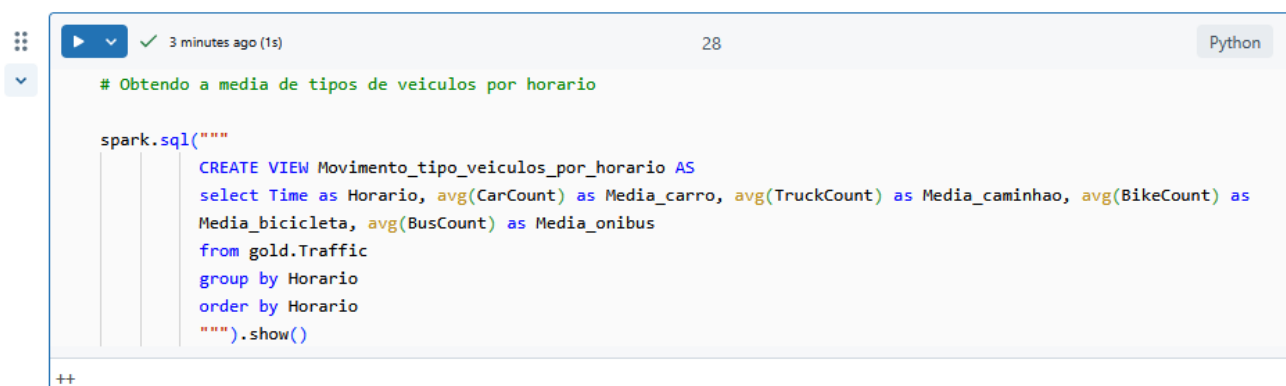


Figura 30

Em seguida foram realizados os seguintes passos para obter a resposta para o questionamento:

- Foi obtido a partir da view “**Media_veiculos_por_horario**”, que disponibiliza a média de veículos totais por horário, o maior valor de média de veículos “**max(media_veiculos)**”;

```
select max(media_veiculos) as maiortrafego
from Media_veiculos_por_horario
)
on media_veiculos = maiortrafego
```

- De posse do valor da maior média de tráfego, foi possível identificar o horário que essa situação ocorreu a partir de uma consulta SQL na view “**Media_veiculos_por_horario**” para “**media_veiculos = maiortrafego**”;

```
select Horario as Horario_maior_trafego, media_veiculos
from Media_veiculos_por_horario
inner join (
select max(media_veiculos) as maiortrafego
from Media_veiculos_por_horario
)
on media_veiculos = maiortrafego
```


- De posse do horário que ocorreu a maior média de veículos foi realizada uma consulta na view “**Movimento_tipo_veiculos_por_horario**” para obter a média por tipo de veículos para esse horário específico de maior movimento.

Segue abaixo a consulta SQL que foi utilizada para obter essa informação.

```
spark.sql("""
    select Horario as Horario_maior_movimento, round(Media_carro,2) as
Media_carro, round(Media_caminhao,2) as Media_caminhao,
round(Media_bicicleta,2) as Media_bicicleta, round(Media_onibus,2) as
Media_onibus
    from Movimento_tipo_veiculos_por_horario
    inner join(
        select Horario as Horario_maior_trafego, media_veiculos
        from Media_veiculos_por_horario
        inner join (
            select max(media_veiculos) as maiortrafego
            from Media_veiculos_por_horario
        )
        on media_veiculos = maiortrafego
    )
    on Horario = Horario_maior_trafego
""").show()
```

Após a sua execução, foi possível identificar as seguintes informações (Figura 31):

- **Horário_maior_movimento:** “5:45:00 PM”;
- **Media_carro:** 117,45
- **Media_caminhao:** 6,75
- **Media_bicicleta:** 26,00
- **Media_onibus:** 55,1

A partir dessas informações, possibilitou identificar que no horário de maior movimento havia uma maior quantidade de carros e uma quantidade considerável de ônibus e bicicletas.

Com essa informação, possibilita o órgão de trânsito da localidade identificar a quantidade de veículos por tipo no momento que ocorre o horário de maior movimento na área analisada. Proporcionando a criação de políticas de trânsito que atendam as necessidades de trânsito da via.

Como foi identificado nesse horário uma quantidade considerável de ônibus e bicicletas. Essa situação pode ser um indicativo de uma necessidade de criação de corredores de ônibus para ter uma maior prioridade ao transporte público e a criação de clicofaixas (um espaço delimitado na própria pista para ciclistas, junto aos demais veículos) ou ciclovias (pista de uso exclusivo para ciclistas).

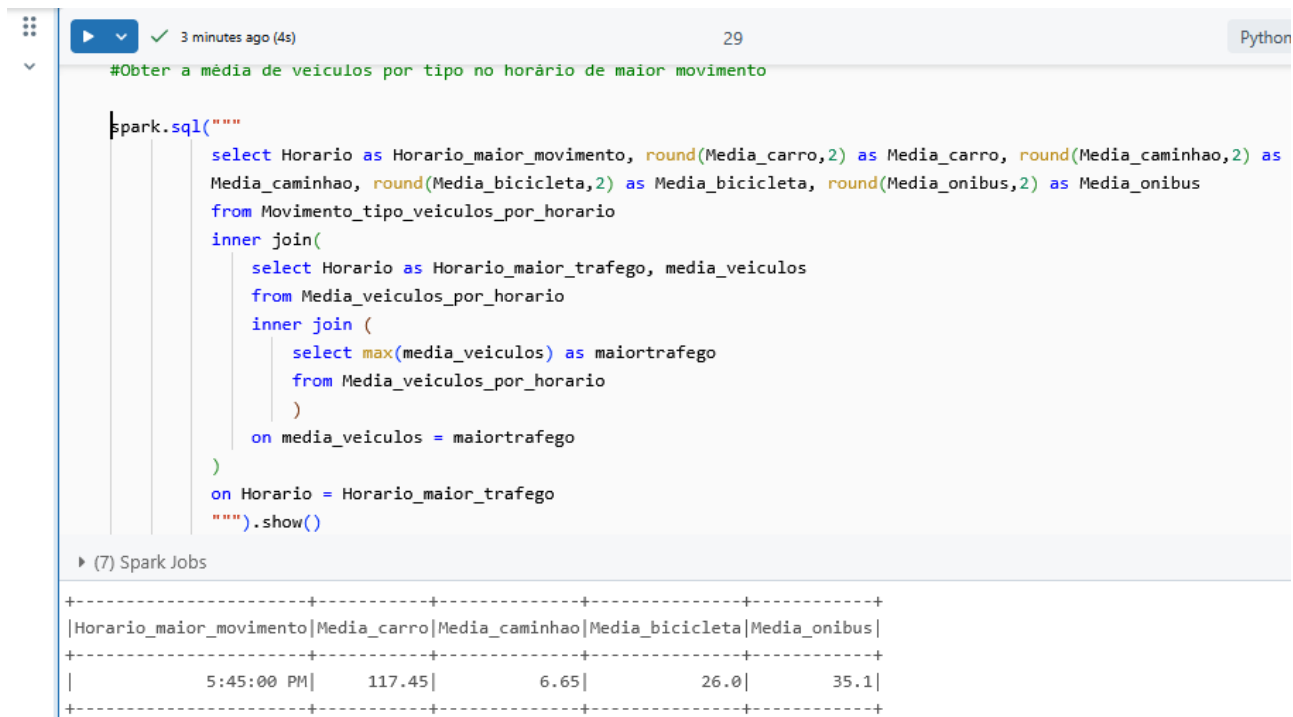


Figura 31

D) Qual é a quantidade média do tráfego de bicicletas nos horários coletados?

Como explicado anteriormente, por dia ocorreram 96 coletas de dados em horários específicos e esses horários serão os mesmos nos dias avaliados.

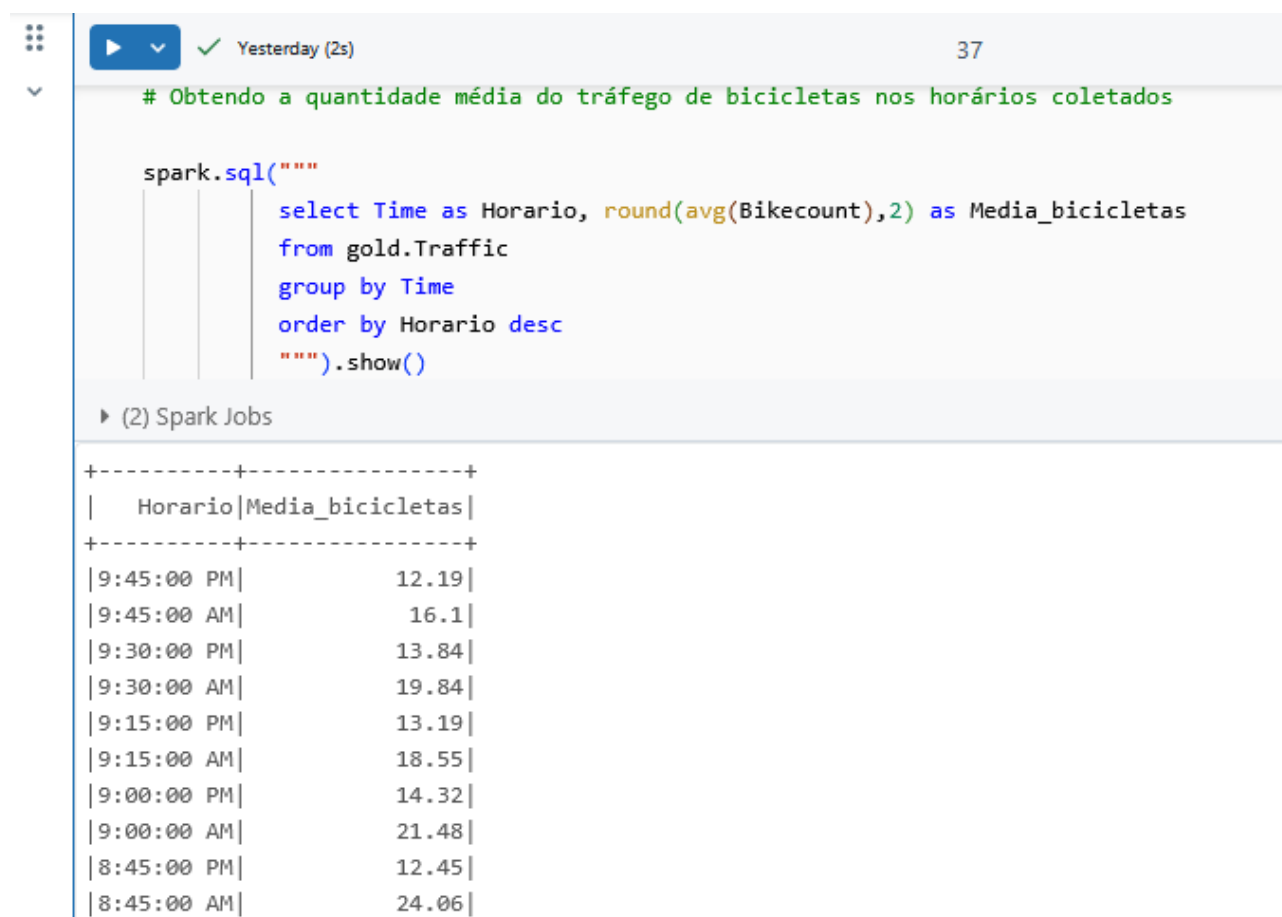
Para obter essa resposta, utilizou-se uma consulta SQL que apresenta as seguintes características:

- Utiliza os dados da tabela “**gold.traffic**”;
- Obtém o horário a partir da coluna “**Time**”;
- Realiza o cálculo da média da contagem de bicicletas (coluna “**Bikecount**”) por horário fazendo o arredondamento de duas casas decimais “**round(avg(Bikecount),2)**”;

```
spark.sql("""
select Time as Horário, round(avg(Bikecount),2) as Media_bicicletas
from gold.Traffic
group by Time
order by Horário desc
""").show()
```

Dessa forma, foi possível obter para cada horário de coleta a média de bicicletas que foram contabilizadas durante o período de monitoração de 1 (um) mês (Figura 32).

Com essa informação, possibilita o órgão de trânsito da localidade identificar qual a melhor estratégia para tratar o trânsito de bicicletas na via que foi monitorada. Sendo possível constatar que pelo volume identificado a via teria a necessidade da criação de uma estrutura para o tráfego de bicicletas (ciclofaixa ou ciclovía). Dependendo da infraestrutura física disponível na via monitorada.



The screenshot shows a Databricks notebook interface. At the top, there's a toolbar with a play button, a checkmark, and the text 'Yesterday (2s)' and '37'. Below the toolbar, the notebook content is displayed. It starts with a comment in green: '# Obtendo a quantidade média do tráfego de bicicletas nos horários coletados'. This is followed by a Spark SQL query in red and blue text. The query selects 'Time' as 'Horario' and the average of 'Bikecount' rounded to 2 decimal places as 'Media_bicicletas' from the 'gold.Traffic' table, grouped by 'Time' and ordered by 'Horario desc'. The query is enclosed in a multi-line string and ends with '.show()'. Below the code, there's a section titled '(2) Spark Jobs'. Underneath, a table of results is shown, with columns 'Horario' and 'Media_bicicletas'. The table contains 10 rows of data, showing the average number of bicycles for different times of day.

```
# Obtendo a quantidade média do tráfego de bicicletas nos horários coletados

spark.sql("""
    select Time as Horario, round(avg(Bikecount),2) as Media_bicicletas
    from gold.Traffic
    group by Time
    order by Horario desc
""").show()
```

► (2) Spark Jobs

Horario	Media_bicicletas
9:45:00 PM	12.19
9:45:00 AM	16.1
9:30:00 PM	13.84
9:30:00 AM	19.84
9:15:00 PM	13.19
9:15:00 AM	18.55
9:00:00 PM	14.32
9:00:00 AM	21.48
8:45:00 PM	12.45
8:45:00 AM	24.06

Figura 32

8- Autoavaliação

Nesse projeto, foram definidos objetivos para obter informações no dataset de coleta de quantidade de veículos por tipo identificados em um período de tempo e em horários específicos. Os questionamentos que foram elaborados puderam ser respondidos e efetuadas análises a partir dos tratamentos e consultas realizadas no ambiente de nuvem do Databricks.

No início, tive uma certa dificuldade no manuseio da console do Databricks devido ao fato de ter sido o primeiro contato com a ferramenta. Devido a isso, foi necessário demandar um certo tempo para aprendizagem do uso da console e comandos que poderiam ser utilizados no notebook da solução.

Como foi utilizada a versão community do Databricks no MVP, as limitações com relação ao uso do cluster acarretaram uma dificuldade adicional na operação da ferramenta. Já que o equipamento (computador) era desligado após um período de tempo de inatividade, sendo necessário realizar com frequência o “clone” do equipamento que fazia parte do cluster.

Segundo informações disponíveis no site da Kaggle, em atualizações futuras do dataset está prevista a inclusão de dados referentes a velocidade dos veículos e novas rotas, possibilitando uma melhor análise para o gerenciamento de tráfego.

Sendo possível, após a atualização, realizar análises mais amplas sobre o tráfego de veículos da localidade. Possibilitando o aprimoramento e criação de novas políticas de trânsito.

9- Conclusão

Este projeto tem como objetivo realizar uma análise das informações sobre o quantitativo e tipos de veículos identificados em uma via pública. Com essas informações será possível realizar um melhor gerenciamento do trânsito em uma cidade e o planejamento de ações.

Para essa análise foi utilizada a solução do Databricks Community, onde foi utilizado a infraestrutura do DBFS para realizar o armazenamento persistente em nuvem do arquivo “csv” do dataset.

Como melhor prática para análise de dados em Big Data, foram realizados análises e tratamento de dados nas camadas bronze, silver e gold. Possibilitando um melhor gerenciamento dos tratamentos que foram realizados nos dados, entre eles, a modificação do nome de duas colunas do dataframe para ser possível transformá-lo em uma tabela de banco de dados.

Após o tratamento dos dados, foi possível realizar as consultas (na tabela gold) para obter as respostas para todos os questionamentos elencados no objetivo do projeto.