TREINAMENTO DATABRICKS

Apostila com os comandos das videoaulas Notebooks

Utilização dos comandos

Apostila criada por Grimaldo Oliveira

Curso DATABRICKS
Prof. Grimaldo Oliveira
grimaldo_lopes@hotmail.com

Conheça o Professor Grimaldo Oliveira

Sou professor das pós-graduações das universidades **UNIFACS, CATÓLICA DO SALVADOR** e **ISL Wyden**. **Mestre** pela **Universidade do Estado da Bahia** (UNEB) no Curso de Mestrado Profissional Gestão e Tecnologias Aplicadas à Educação (GESTEC). Possuo Especialização em Análise de Sistemas pela Faculdade Visconde de Cairu e **Bacharelado** em **Estatística** pela Universidade Federal da Bahia. Atuo profissionalmente como consultor há mais de **15 anos nas áreas de Data Warehouse, Mineração de Dados, Ferramentas de Tomada de Decisão e Estatística**.

BI PRO

Idealizador do treinamento online **BI PRO** com + de 10 módulos contendo todas as disciplinas para formação completa na área de dados. Quem participa do **BI PRO** tem acesso gratuito: todos os meus cursos de dados da Udemy, + ebook BI COMO DEVE SER - O Guia Definitivo, espaço de mentoria para retirada de dúvidas, respostas das atividades. Acesse www.bipro.com.br

Autor do eBook **BI COMO DEVE SER - O Guia Definitivo,** com ele você poderá entender os conceitos e técnicas utilizados para o desenvolvimento de uma solução BI, tudo isso de forma objetiva e prática, com linguagem acessível tanto para técnicos quanto gestores e analista de negócio. Acesse www.bicomodeveser.com.br

Site de **cupons** do prof. Grimaldo, com desconto de todos os seus cursos de dados da Udemy, atualizado diariamente com diversas promoções, incluindo cursos gratuitos. Acesse https://is.gd/CUPOMCURSOSPROFGRIMALDO

Curso DATABRICKS
Prof. Grimaldo Oliveira
grimaldo_lopes@hotmail.com

Idealizador do Blog **BI COM VATAPÁ** reúne informações diversas sobre a área de dados com detalhes sobre o mundo de Business Intelligence, Big Data, Ciência de dados, Mineração de dados e muitos outros. Acesse http://bicomvatapa.blogspot.com/

Aula – comandos

Esta apostila contém os comandos utilizados nos NOTEBOOKS para interação com o DATABRICKS

- Listando o arquivo de vinhos carregado diretamente para o DBFS

```
%sql
select * from vinho
```

- Carregando arquivo CSV em python

```
%python

clientes = spark.read.format('csv').options(header='true',
inferSchema='true',
delimiter=';').load('/FileStore/tables/carga/clientes_cartao.c
sv')

display(clientes)
```

- Carregando arquivo CSV em Scala

```
%scala
  val cliente = spark.read.format("csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .option("delimiter", ";")
  .load("/FileStore/tables/carga/clientes_cartao.csv")

display(cliente)
```

- Trabalhando com o arquivo carregado em Scala no SQL

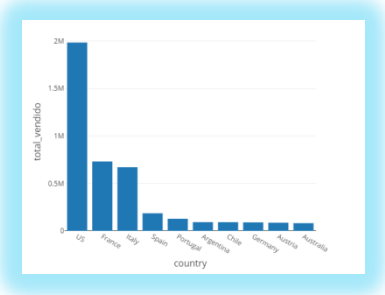
```
%scala
cliente.createOrReplaceTempView("dados_cliente")
```

- Exibindo os dados no SQL

```
%sql
select * from dados_cliente
```

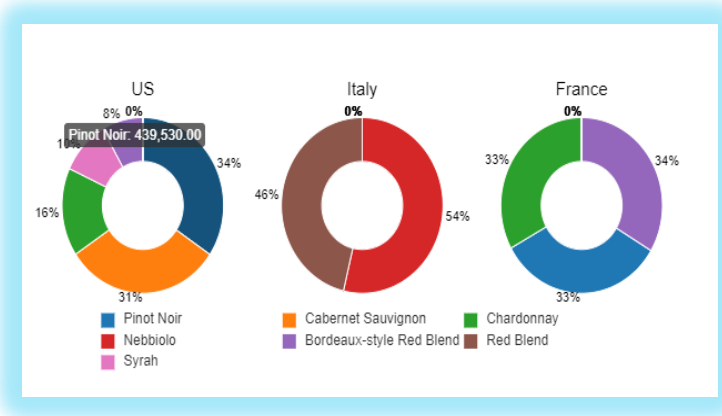
- Preparando uma agregação com os dados, gráfico de barra

```
%sql
select pais, sum(preco) as total_vendido from vinho
where preco > 0
group by pais
order by total_vendido desc
limit 10
```



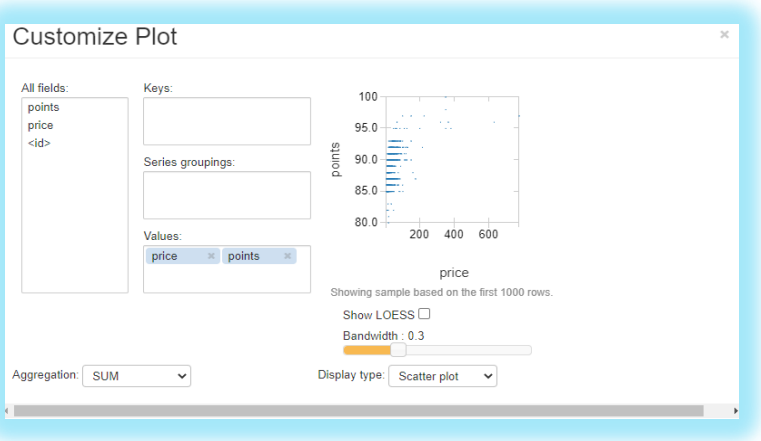
- Preparando uma agregação com os dados, gráfico de setores

```
%sql
select pais, variante, sum(preco) as total_vendido from vinho
where preco > 0
group by pais,variante
order by total_vendido desc
limit 10
```



- Preparando análise de dispersão, gráfico de dispersão

```
%sql
select pontos, preco from vinho
```



Notebook : Leitura e gravação arquivos JSON

- Lista de arquivos Json que estão armazenados no DBFS

```
%fs ls /databricks-datasets/structured-streaming/events/
```

- Exibindo um arquivo Json com as informações

```
%fs head /databricks-datasets/structured-streaming/events/file-1.json
```

- Carregando 1 arquivo Json para o dataframe

```
%python
# Lendo 1 arquivo JSON
dataf = spark.read.json("/databricks-datasets/structured-streaming/events/file-1.json")
dataf.printSchema()
dataf.show()
```

- Carregando 2 arquivos Json para o dataframe

```
%python
#Lendo 2 arquivos JSON
dataf2 = spark.read.json(['/databricks-datasets/structured-streaming/events/file-1.json','/databricks-datasets/structured-streaming/events/file-2.json'])
dataf2.show()
```

- Carregando TODOS os arquivos Json para o dataframe

```
%python
#Lendo todos os arquivos JSON
dataf3 = spark.read.json("/databricks-datasets/structured-streaming/events/*.json")
dataf3.show()
```

- Unificando todos os arquivos que foram guardados no dataframe dataf3 para um novo arquivo JSON

```
%python
#Gravação dos dados que estão no dataframe para JSON em um único
arquivo
dataf3.write.json("/FileStore/tables/JSON/eventos.json")
```

- Criação de uma tabela para executar SQL

```
%python
spark.sql("CREATE OR REPLACE TEMPORARY VIEW view_evento USING json
OPTIONS" +
      " (path '/FileStore/tables/JSON/eventos.json')")
spark.sql("select action from view_evento").show()
```

Notebook : Leitura e gravação arquivos PARQUET

- Criação de um dataframe com dados

```
#criando um dataframe com dados fixos
dados =[("Grimaldo
","Oliveira","Brasileira","Professor","M",3000),
              ("Ana ","Santos","Portuguesa","Atriz","F",4000),

("Roberto","Carlos","Brasileira","Analista","M",4000),
              ("Maria
","Santanna","Italiana","Dentista","F",6000),

("Jeane","Andrade","Portuguesa","Medica","F",7000)]
colunas=["Primeiro_Nome","Ultimo_nome","Nacionalidade","Trabal
ho","Genero","Salario"]
datafparquet=spark.createDataFrame(dados,colunas)
datafparquet.show()
```

- Gravando o arquivo parquet

```
#criando o arquivo parquet
datafparquet.write.parquet("/FileStore/tables/parquet/pessoal.
parquet")
```

- Subscrevendo o arquivo parquet

```
#Permite uma atualização do arquivo parquet
datafparquet.write.mode('overwrite').parquet('/FileStore/table
s/parquet/pessoal.parquet')
```

- Lendo o arquivo parquet e guardando em um dataframe

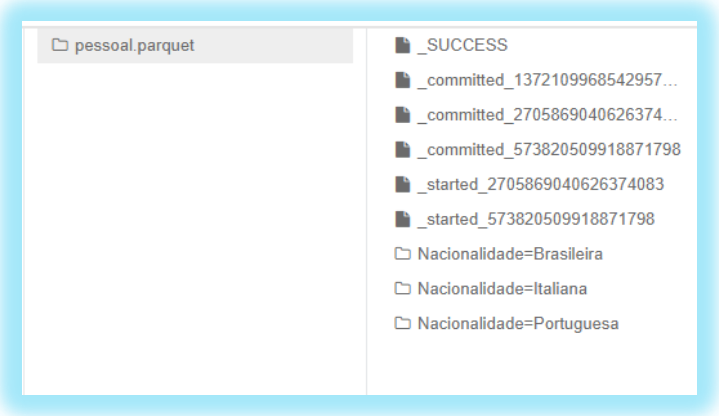
```
#Realizando uma atualização do arquivo parquet
datafleitura=
spark.read.parquet("/FileStore/tables/parquet/pessoal.parquet"
)
datafleitura.show()
```

- Realizando uma consulta SQL

```
#Criando uma consulta em SQL
datafleitura.createOrReplaceTempView("Tabela_Parquet")
ResultSQL = spark.sql("select * from Tabela_Parquet where
salario >= 6000 ")
ResultSQL.show()
```

- Particionando os dados do arquivo parquet em grupos

```
#Particionando os dados em um arquivo parquet
datafparquet.write.partitionBy("Nacionalidade","salario").mode
("overwrite").parquet("/FileStore/tables/parquet/pessoal.parqu
et")
```



- Exibindo os dados do parquet cuja a nacionalidade é brasileira

```
#Lendo o aquivo participonado do parquet
datafnacional=spark.read.parquet("/FileStore/tables/parquet/pe
ssoal.parquet/Nacionalidade=Brasileira")
datafnacional.show(truncate=False)
```

- Realizando uma pesquisa via SQL no arquivo parquet particionado

```
#Consultando diretamente o arquivo parquet particionado via
SQL
spark.sql("CREATE OR REPLACE TEMPORARY VIEW Cidadao USING
parquet OPTIONS (path
\"/FileStore/tables/parquet/pessoal.parquet/Nacionalidade=Bras
ileira\")")
spark.sql("SELECT * FROM Cidadao where
Ultimo_nome='Oliveira'").show()
```

Notebook : Leitura arquivos CSV e gravação arquivos PARQUET

- Realizando a leitura do CSV e guardando no dataframe

```
#Leitura de arquivo CSV
dataframesp= spark.read.format("csv").option("header",
"true").load("/FileStore/tables/arquivo/Datafiniti_Hotel_Revie
ws_Jun19.csv")
dataframesp.show()
```

- Realizando a leitura do CSV e guardando dados no dataframe (via Scala)

```
%scala
val dataframescala =
sqlContext.read.format("com.databricks.spark.csv")
.option("delimiter", ",")
.load("/FileStore/tables/arquivo/Datafiniti_Hotel_Reviews_Jun1
9.csv")
dataframescala.show()
```

- Gravando o dataframe no formato em parquet

```
#criando o arquivo parquet
dataframesp.write.parquet("/FileStore/tables/parquet/csvparque
t.parquet")
```

- Realizando a leitura em parquet

```
#Realizando uma leitura do arquivo parquet
datafleitura=spark.read.parquet("/FileStore/tables/parquet/csv
parquet.parquet")
datafleitura.show()
```

Comandos : Sistema de arquivos DATABRICKS

- Acessando o help de comando
```
%fs help
```

- Listando as pastas
```
%fs ls /
```

- Listando as pastas
```
dbutils.fs.ls("/")
```

- Criando as pastas (exemplo, criando a pasta vendas)
```
%fs mkdirs vendas
```

- Mostrando a pasta criada (exemplo, a pasta vendas)
```
%fs ls /FileStore/
```

- Copiando um arquivo de uma pasta para outra
```
%fs cp /FileStore/tables/carga/vinhos_no_mundo.csv
/FileStore/vendas2/copia_vinhos.csv
```

- Copiando um arquivo de uma pasta para outra
```
dbutils.fs.cp("/FileStore/tables/carga/vinhos_no_mundo.csv",
"/FileStore/vendas3/copia2_vinhos.csv"
```

- Renomeando (troca de nome) de um arquivo
```
%fs mv /FileStore/vendas2/copia_vinhos.csv
/FileStore/vendas2/copia_muda_vinhos.csv
```

- Renomeando (troca de nome) de um arquivo
```
dbutils.fs.mv("/FileStore/vendas2/copia_muda_vinhos.csv",
"/FileStore/vendas2/copia_troca_vinhos.csv")
```

- Elimina um arquivo
```
%fs rm /FileStore/vendas2/copia_troca_vinhos.csv
```

- Elimina um arquivo
```
%fs rm /FileStore/vendas3/copia2_vinhos.csv
ou
dbutils.fs.rm("/FileStore/vendas3/copia2_vinhos.csv")
```

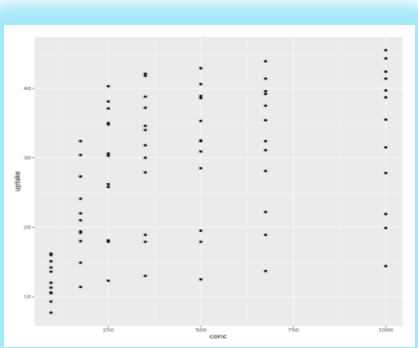
- Elimina a pasta
```
%fs rm -r /FileStore/vendas2/
Ou
dbutils.fs.rm("/FileStore/vendas2/", recurse=True)
```

- Realizando uma pesquisa em bash-linux
```
%%bash
find /databricks -name "*.csv" | grep "fa"
```

Scripts R : Construção de gráficos e dashboards

- Script 1

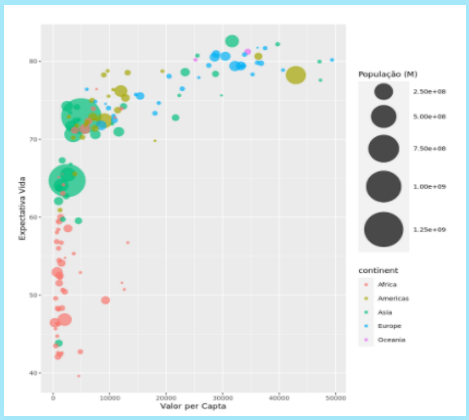
```
## uso base interna CO2
library(ggplot2)
ggplot(data = CO2) +
  geom_point(mapping = aes(x = conc, y = uptake))
```



- Script 2

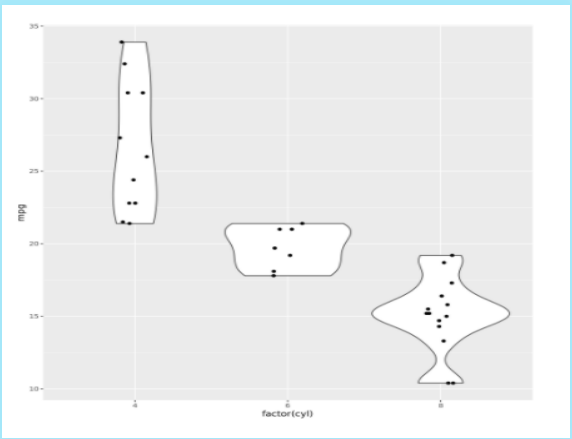
```
#Mapa de Bolhas
# Libraries
install.packages("gridExtra")
library(ggplot2)
library(dplyr)
# carga dos dados
install.packages("gapminder")
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-
year)

# Exibição do gráfico
data %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%
  ggplot(aes(x=gdpPercap, y=lifeExp, size=pop, color=continent))
+
  geom_point(alpha=0.7) +
  scale_size(range = c(.5, 24), name="População (M)") +
  labs(x = "Valor per Capta", y = "Expectativa Vida")
```



- Script 3

```
library(ggplot2)
# base interna mtcars - Violino
dados <- ggplot(mtcars, aes(factor(cyl), mpg))
dados + geom_violin()
```



Scripts pyspark : erro ao carregar arquivo vinhos_no_mundo.csv

- Definindo o schema da tabela

```
from pyspark.sql.types import StructType, IntegerType, DateType,
StringType, DecimalType, FloatType
Record_schema = (StructType().
add("registro", StringType()).
add("pais", StringType()).
add("descricao", StringType()).
add("destino",StringType()).
add("pontos",FloatType()).
add("preco",FloatType()).
add("provincia",StringType()).
add("regiao_1",StringType()).
add("regiao_2",StringType()).
add("somelier",StringType()).
add("twiter_somelier",StringType()).
add("endereco",StringType()).
add("variante",StringType()).
add("vinicola",StringType())
)
```

- Carregado o schema da tabela

```
# File location and type
file_location = "/FileStore/tables/carga/vinhos_no_mundo.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 = spark.read.format(file_type).schema(Record_schema) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

- Criando a visão temporária

```
# Create a view or table

temp_table_name = "vinhos_no_mundo_csv"
df.createOrReplaceTempView(temp_table_name)
```

- Executando a consulta via SQL da view temporária

```
%sql

/* Query the created temp table in a SQL cell */
select * from `vinhos_no_mundo_csv`
```

- Criando a tabela permanente

```
permanent_table_name = "vinho"
df.write.format("parquet").saveAsTable(permanent_table_name)
```