

ALTA DATABRICKS



Try Databricks free

Test-drive the full Databricks platform free for 14 days on your choice of AWS, Microsoft Azure or Google Cloud.

- ✓ Simplify data ingestion and automate ETL
Ingest data from hundreds of sources. Use a simple declarative approach to build data pipelines.
- ✓ Collaborate in your preferred language
Code in Python, R, Scala and SQL with coauthoring, automatic versioning, Git integrations and RBAC.
- ✓ 12x better price/performance than cloud data warehouses
See why over 7,000 customers worldwide rely on Databricks for all their workloads from BI to AI.



Create your Databricks account

1/2

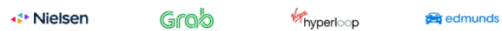
First name	Last Name
<input type="text" value="Juan"/>	<input type="text" value="García"/>
Email	
<input type="text" value="juangmuelas@gmail.com"/>	
Company	Title
<input type="text" value="CIDEAD"/>	<input type="text" value="STUDENT"/>
<small>This field is required</small>	<small>This field is required</small>
Phone (Optional)	
<input type="text"/>	
Country	
<input type="text" value="Spain"/>	
<input type="checkbox"/> Yes, I would like to receive marketing communications regarding Databricks services, events and open source products. I understand I can update my preferences at any time.	
<input type="button" value="Continue"/>	



Try Databricks free

Test-drive the full Databricks platform free for 14 days on your choice of AWS, Microsoft Azure or Google Cloud.

- ✓ Simplify data ingestion and automate ETL
Ingest data from hundreds of sources. Use a simple declarative approach to build data pipelines.
- ✓ Collaborate in your preferred language
Code in Python, R, Scala and SQL with coauthoring, automatic versioning, Git integrations and RBAC.
- ✓ 12x better price/performance than cloud data warehouses
See why over 7,000 customers worldwide rely on Databricks for all their workloads from BI to AI.



Choose a cloud provider

2/2

	Amazon Web Services
	Microsoft Azure
	Google Cloud Platform

By clicking "Get Started," you agree to the [Privacy Policy](#) and [Terms of Service](#).

Don't have a cloud account?

Community Edition is a limited Databricks environment for personal use and training.

[Get started with Community Edition](#)

By clicking "Get started with Community Edition," you agree to the [Privacy Policy](#) and [Terms of Service](#).



Check your email to start your trial.

Thank you for signing up. Please validate your email address to start your trial.

Here are some resources to help you deploy your first workspace.

1. [Review the administration guide](#) on the requirements to set up your Databricks service.
 - Not an admin on your AWS Account? Share [this guide](#) with your admin to deploy a workspace for you!
2. [Follow our Quickstart guide to create your first workspace.](#)

You can also check out our [Docs](#) and [Community](#) sites to get your questions answered.

Note: if you signed up for Community Edition, you'll go to your first workspace as soon as you verify your email address.

Tras recibir el correo, verificamos con una contraseña y nos pasa directamente a la versión Community.

Data Science & Engineering

You're using Databricks Community Edition. Upgrade for unlimited clusters and collaboration features. [Upgrade now](#)

Notebook
Create a new notebook for querying, data processing, and machine learning.
[Create a notebook](#)

Data import
Quickly import data, preview its schema, create a table, and query it in a notebook.
[Browse files](#)

AutoML
Quickly train ML models for discovery and iteration.
[Start AutoML](#)

Transform data
dbt Core

Guide: Quickstart tutorial
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.
[Start tutorial](#)

Recents

Name	Last viewed
There are no recents yet	

[Documentation](#) [Release notes](#) [Blog posts](#)

github.com/databricks-academy

Product Solutions Open Source Pricing

Search Sign in Sign up

Databricks Academy

884 followers United States of America

Overview Repositories 44 Projects Packages People

Popular repositories

[data-engineering-with-databricks-english](#) Public

Python 1k 990

[apache-spark-programming-with-databricks](#) Public

Python 229 281

[advanced-data-engineering-with-databricks](#) Public

Python 219 285

[data-analysis-with-databricks-sql](#) Public

Python 110 130

[ml-in-production-english](#) Public

Machine Learning in Production

Python 95 91

[scalable-machine-learning-with-apache-spark-english](#) Public

Python 71 125

People

This organization has no public members. You must be a member to see who's a part of this organization.

Top languages

Python Scala

databricks-academy / apache-spark-programming-with-databricks Public

Code Pull requests 2 Actions Security Insights

published 7 branches 15 tags

Go to file Code - About

daboncanplay published v2.2.9		ab8dccc on Jan 12 29 commits
ASP 1 - Introductions	published v2.2.9 2 months ago	
ASP 2 - Spark Core	published v2.2.9 2 months ago	
ASP 3 - Functions	published v2.2.9 2 months ago	
ASP 4 - Performance	published v2.2.9 2 months ago	
ASP 5 - Streaming	published v2.2.9 2 months ago	
ASP 6 - Delta Lake	published v2.2.9 2 months ago	
Includes	publishing v2.2.8 3 months ago	
Solutions	published v2.2.9 2 months ago	

No description, website, or topics provided.

Readme

CC0-1.0 license

229 stars

6 watching

281 forks

Releases 15

Apache Spark Programming wit... Latest on Jan 12

+ 14 releases

Copiamos el enlace si no lo queremos descargar

databricks-academy / apache-spark-programming-with-databricks Public

Code Pull requests 2 Actions Security Insights

Releases Tags

Find a release

Jan 12

daboncanplay

v2.2.9

ab8dccc

Compare

Apache Spark Programming with Databricks, v2.2.9 Latest

Minimum Cores: 4

Cluster Mode: Single Node

DBRs: 11.3 LTS, 11.3 Photon or 11.3 ML

Assets 4

apache-spark-programming-with-databricks-v2.2.9-notebooks.dbc	316 KB	Jan 12
apache-spark-programming-with-databricks.pdf	5.7 MB	Jan 12
Source code (zip)		Jan 12
Source code (tar.gz)		Jan 12

Dec 21, 2022

daboncanplay

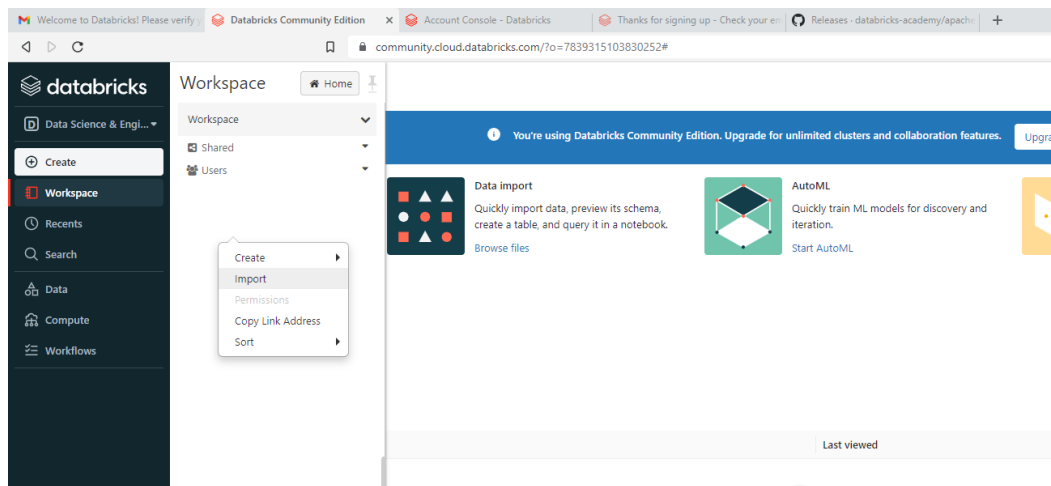
v2.2.8

.....

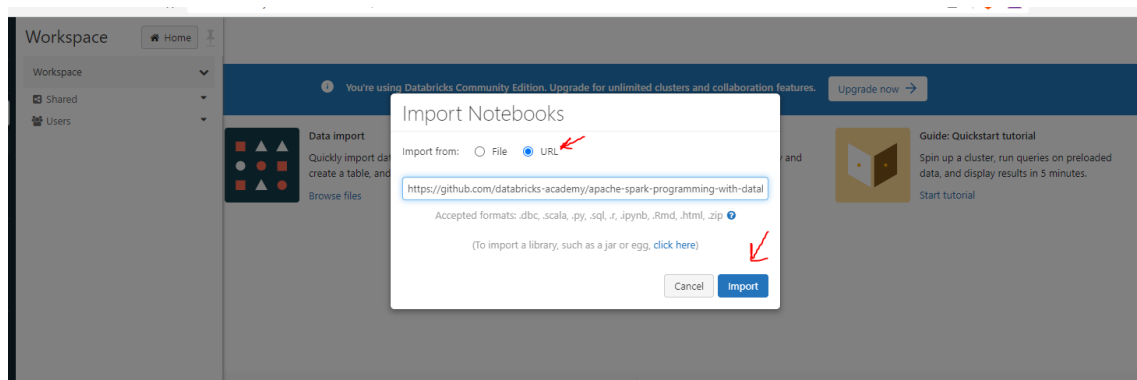
Apache Spark Programming with Databricks, v2.2.8

Minimum Cores: 4

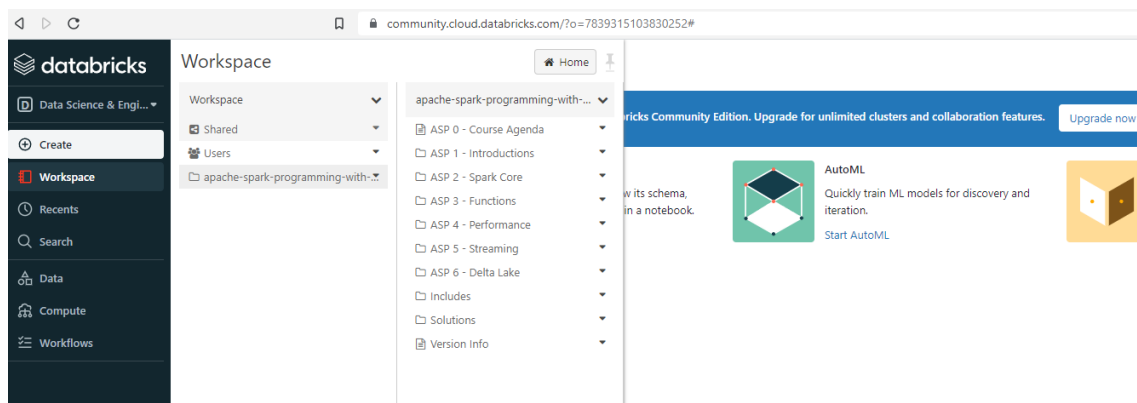
<https://github.com/databricks-academy/apache-spark-programming-with-databricks/releases/download/v2.2.9/apache-spark-programming-with-databricks-v2.2.9-notebooks.dbc>



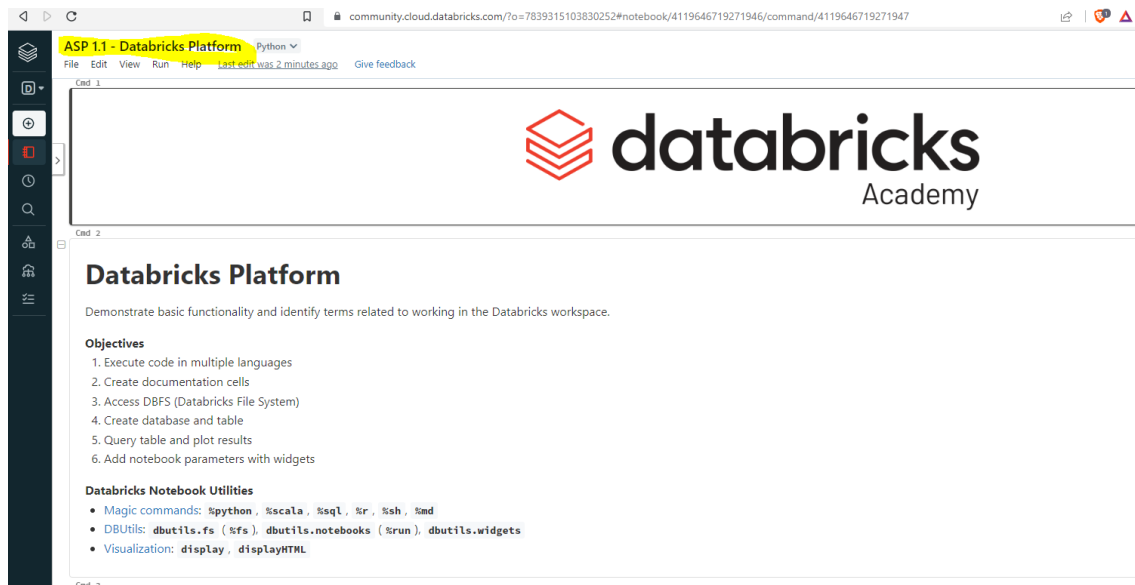
En Workspace, botón derecho: importar. Y pegamos nuestro enlace.



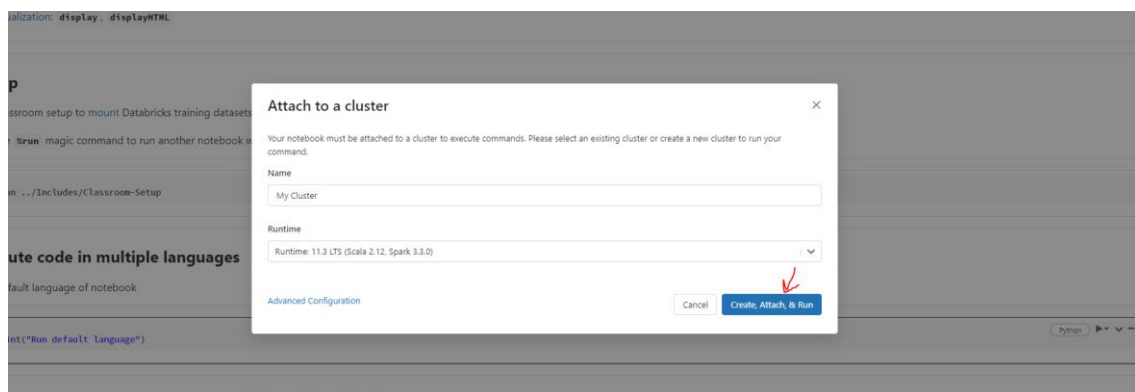
Tras unos momentos, podemos verlo:



Podemos ya abrir el primero de los archivos sobre el que trabajar.

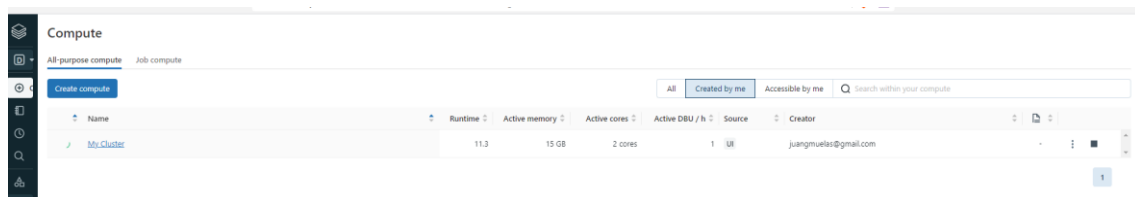


Necesitamos trabajar con un cluster, que se puede crear desde la pestaña “compute”, o el mismo databricks intentará crear uno si se intenta ejecutar una celda (Ctrl+intro). Sino hay ninguno nos “acompañará” en la creación.



En la versión community, los cluster dejan de funcionar a los 20 minutos de inacción, y no es posible volver a arrancarlos. Hay que crear otro nuevo y no hay más problema.

Vemos como carga



Para poder usar sql necesitamos que lo reconozca desde Python

Cmd 33

```

1 files = dbutils.fs.ls(DA.paths.events)
2 display(files)

```

(3) Spark Jobs

Table +

	path	name	size	modificationTime
1	dbfs:/mnt/dbacademy-datasets/apache-spark-programming-with-databricks/v03/ecommerce/events/events.delta/_delta_log/	_delta_log/	0	0
2	dbfs:/mnt/dbacademy-datasets/apache-spark-programming-with-databricks/v03/ecommerce/events/events.delta/part-00000-eb69ecaf-fbe1-4820-9513-24e158ed1e22-c000.snappy.parquet	part-00000-eb69ecaf-fbe1-4820-9513-24e158ed1e22-c000.snappy.parquet	75373205	1677761150000
3	dbfs:/mnt/dbacademy-datasets/apache-spark-programming-with-databricks/v03/ecommerce/events/events.delta/part-00001-e9be20a6-591a-4c06-9284-36d33f8bb378-c000.snappy.parquet	part-00001-e9be20a6-591a-4c06-9284-36d33f8bb378-c000.snappy.parquet	75384788	1677761155000
4	dbfs:/mnt/dbacademy-datasets/apache-spark-programming-with-databricks/v03/ecommerce/events/events.delta/part-00002-5793eed4-8dea-4287-abe1-a8ed30032f86-c000.snappy.parquet	part-00002-5793eed4-8dea-4287-abe1-a8ed30032f86-c000.snappy.parquet	75393846	1677761159000
5	dbfs:/mnt/dbacademy-datasets/apache-spark-programming-with-databricks/v03/ecommerce/events/events.delta/part-00003-3c9024f7-5419-45b5-873d-4756e510a797-c000.snappy.parquet	part-00003-3c9024f7-5419-45b5-873d-4756e510a797-c000.snappy.parquet	75295715	1677761164000

5 rows | 3.51 seconds runtime

Command took 3.51 seconds -- by juangmuelas@gmail.com at 2/3/2023, 13:55:05 on My Cluster

Cmd 34

But, Wait!

cannot use variables in SQL commands.

With the following trick you can!

Declare the python variable as a variable in the spark context which SQL commands can access:

Cmd 35

```

1 spark.conf.set("whatever.events", DA.paths.events)

```

Command took 0.08 seconds -- by juangmuelas@gmail.com at 2/3/2023, 13:55:47 on My Cluster

Cmd 36

Podemos usar la creación de widget para filtrar en nuestras búsquedas

File

Edit

View

Run

Help

Last edit was 31 minutes ago

Give feedback

state

CA

Run the query below and then plot results by selecting the bar chart icon.

Cmd 47

```

1 %sql
2 SELECT traffic_source, SUM(ecommerce.purchase_revenue_in_usd) AS total_revenue
3 FROM events
4 GROUP BY traffic_source

```

Cmd 48

Add notebook parameters with widgets

Use **widgets** to add input parameters to your notebook.

Create a text input widget using SQL.

Cmd 49

```

1 %sql
2 CREATE WIDGET TEXT state DEFAULT "CA"

```

OK

Command took 0.18 seconds -- by juangmuelas@gmail.com at 2/3/2023, 13:59:42 on My Cluster

Cmd 50

Access the current value of the widget using the function **getArgument**

Cmd 51

```

1 %sql
2 SELECT *
3 FROM events
4 WHERE geo.state = getArgument("state")

```

Cmd 52

Favorite Color? Name state

```

3 FROM events
4 WHERE geo.state = getArgument("state")

```

Cnd 52

Remove the text widget

Cnd 53

```

1 $sql
2 REMOVE WIDGET state

```

Cnd 54

To create widgets in Python, Scala, and R, use the DBUtils module: `dbutils.widgets`

Cnd 55

```

1 dbutils.widgets.text("name", "Brickster", "Name")
2 dbutils.widgets.multiselect("colors", "orange", ["red", "orange", "black", "blue"], "Favorite Color?")

```

Command took 0.12 seconds -- by juangmuelas@gmail.com at 2/3/2023, 14:01:35 on My Cluster

Cnd 56

Access the current value of the widget using the `dbutils.widgets` function `get`

Cnd 57

```

1 name = dbutils.widgets.get("name")
2 colors = dbutils.widgets.get("colors").split(",")
3
4 html = "<div>Hi {}! Select your color preference.</div>".format(name)
5 for c in colors:
6     html += "<label for={} style='color:{}'><input type='radio'> {}</label><br>".format(c, c, c)
7
8 displayHTML(html)

```

Accedemos a los datos mediante `get()`.

```

1 dbutils.widgets.removeAll()

```

Cnd 60

Clean up classroom

Clean up any temp files, tables and databases created by this lesson

Cnd 61

```

1 DA.cleanup()

```

Cnd 62

© 2023 Databricks, Inc. All rights reserved.
 Apache, Apache Spark, Spark and the Spark logo are trademarks of the Apache Software Foundation.
[Privacy Policy](#) | [Terms of Use](#) | [Support](#)

Todos acaban con `cleanup()` para borrar todo lo creado con el libro.

[Preview Table](#)

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name

Create in Database

File Type

Column Delimiter
☒ First row is header ☐ Infer schema ☐ Multi-line

[Create Table](#)

Table Preview

airport_id	city	state	name
10165	Adak Island	AK	Adak
10299	Anchorage	AK	Ted Stevens Anchorage International
10304	Aniak	AK	Aniak Airport
10754	Barrow	AK	Wiley Post/Will Rogers Memorial
10551	Bethel	AK	Bethel Airport
10926	Cordova	AK	Merle K Mudhole Smith

Cluster **BDAG4**

[Preview Table](#)

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name **flights_csv**

Create in Database **default**

File Type **CSV**

Column Delimiter **,**

☒ First row is header

☐ Infer schema

☐ Multi-line

[Create Table](#)

Table Preview

DayOfMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	DepDelay	ArrDelay
19	5	DL	11433	13303	-3	1
19	5	DL	14869	12478	0	-8
19	5	DL	14057	14869	-4	-15
19	5	DL	15016	11433	28	24
19	5	DL	11193	12892	-6	-11
19	5	DL	10397	15016	-1	-19

Ejercicio 1

Comprueba que los archivos "airports.csv" y "flights.csv" están en DBFS

Cmd 6

```
1 # TODO: Escribe el código para que se muestre un listado similar a la salida de debajo
```

	path	name	size	modificationTime
1	dbfs:/FileStore/airports.csv	airports.csv	16308	1675077017000
2	dbfs:/FileStore/flights.csv	flights.csv	72088113	1675077351000
3	dbfs:/FileStore/import-stage/	import-stage/	0	0
4	dbfs:/FileStore/notas.txt	notas.txt	61	1675081355000
5	dbfs:/FileStore/tables/	tables/	0	0

5 rows | 11.89 seconds runtime

Command took 11.89 seconds -- by a user at 1/2/2023, 10:31:15 on unknown cluster

Cmd 7

```
mv("/mnt/my-folder/a", "s3n://bucket/b")
```

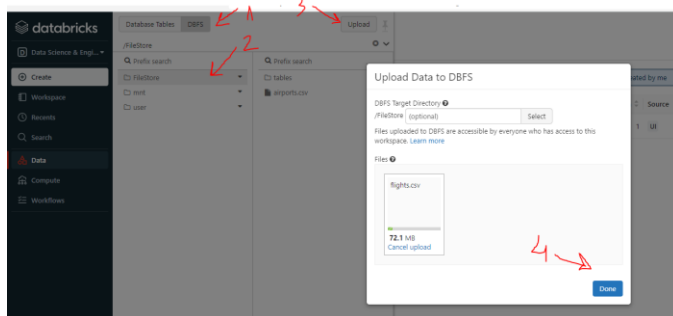
```
mv("/FileStore/tables/airports_csv", " FileStore/tables/airports_csv ")
```

<https://www.youtube.com/watch?v=IOHuO-sbykk>

```
databricks fs mv dbfs:/tmp/my-file.txt dbfs:/parent/child/grandchild/my-file.txt
```

HABILITAR DBFS EN SETTINGS! (hacer un refresh para que se hagan los cambios)

Luego ya Podemos subir los archivos.



1	dbfs/FileStore/airports.csv	airports.csv	16308	1675077017000
2	dbfs/FileStore/flights.csv	flights.csv	72088113	1675077351000
3	dbfs/FileStore/import-stage/	import-stage/	0	0
4	dbfs/FileStore/notes.txt	notes.txt	61	1675081355000
5	dbfs/FileStore/tables/	tables/	0	0

5 rows | 11.89 seconds runtime

Command took 11.89 seconds -- by a user at 1/2/2023, 18:31:15 on unknown cluster

Cell 2

1 %fs ls

Table +

	path	name	size	modificationTime
1	dbfs/FileStore/	FileStore/	0	0
2	dbfs/databricks-datasets/	databricks-datasets/	0	0
3	dbfs/databricks-results/	databricks-results/	0	0
4	dbfs/mnt/	mnt/	0	0
5	dbfs/user/	user/	0	0

5 rows | 1.74 seconds runtime

Command took 1.74 seconds -- by juangmuelasigua11.com at 3/3/2023, 15:47:44 on BDA04

Cell 3

%fs ls FileStore

O también mediante:

```
files = dbutils.fs.ls("dbfs:/FileStore")
display(files)
```

Si hace falta borrar algún fichero subido de más:

```
dbutils.fs.rm("dbfs:/FileStore/flights-1.csv")
```

Pruebo si funcionan las tablas subidas:

%sql

```
select * from airports order by airport_id desc limit 5
```

```
display(spark
    .table("flights")
    .select("*")
    .orderBy("carrier", ascending=False)
    .limit (5)
)
```

Crea los dataframes `airports_df` y `flights_df` a partir de los ficheros anteriores y muestra las primeras filas y su esquema y comprueba que es el esquema es correcto.

Escribe el código para crear `airports_df` y `flights_df`

```
airports_df = spark.table("airports")
```

```
flights_df = spark.table("flights")
```

Escribe el código para que muestre las primeras filas(`airports_df`)

```
display(airports_df)
```

Para mostrar el esquema de `airports_df`

```
airports_df.printSchema()
```

Escribe el código para que muestre las primeras filas(`flights_df`)

```
display(flights_df)
```

Para mostrar el esquema de `flights_df`

```
flights_df.printSchema()
```

Ejercicio 2

Usando los dataframes anteriores muestra las cinco compañías que más vuelos retrasados tienen (las celdas tiene que ser de tipo Python).

- El campo Carrier contiene la compañía aérea.
- Vamos a considerar que un vuelo llega con retraso cuando el vuelo llega más de 15 minutos tarde (campo ArrDelay > 15).
- El nombre del campo que contine la cuenta de vuelos retrasados se llamará delayed_flights.

```
display(spark.sql("SELECT carrier, COUNT(*) as delayed_flights FROM
flights_temp WHERE arrdelay > 15 GROUP BY carrier ORDER BY delayed_flights
DESC LIMIT 5"))
```

MEDIANTE PYSPARCK

```
display(flights_df
        .where("arrdelay >15") #filter
        .groupBy("carrier")
        .count()
        .withColumnRenamed("count", "delayed_flights")
        .orderBy("delayed_flights", ascending= False)
        .limit(5)
    )
```

Ejercicio 3

Almacena los dataframes airports_df y flights_df en sendas tablas temporales y realiza el ejercicio anterior usando una celda de tipo SQL.

```
# Almacena airports_df en la tabla airports y flights_df en la tabla flights
airports_df.createOrReplaceTempView("airports")
flights_df.createOrReplaceTempView("flights")
```

```
%sql
SELECT carrier, COUNT(*) as delayed_flights FROM flights WHERE arrdelay > 15
GROUP BY carrier
ORDER BY delayed_flights DESC LIMIT 5
```

Ejercicio 4

Usando los dataframes airports_df y flights_df muestra los 5 AEROPUERTOS DE DESTINO que mejor recuperación de tiempo en vuelo tienen (las celdas tienen que ser de tipo Python).

- Se considera que se ha recuperado el tiempo de un vuelo cuando habiendo salido con retraso (DepDelay > 15), llega sin retraso (ArrDelay <= 15).
- Se trata de que muestres los nombres de los 5 aeropuertos de llegada que han recuperado el tiempo en un mayor porcentaje de vuelos que salieron retrasados.

```
display(spark.sql(
    """ SELECT airports.name as Airport, COUNT(CASE WHEN arrdelay <= 15 THEN 1
END) / COUNT(*) as percent_recovered
FROM airports,flights WHERE flights.depdelay > 15 AND airports.airport_id =
flights.DestAirportID GROUP BY airports.name ORDER BY percent_recovered DESC
LIMIT 5 """
    )
    )
```

```

from pyspark.sql.functions import count, when, col
#SELECT airports.name as Airport, COUNT(CASE WHEN arrdelay <= 15 THEN 1 END)
/ COUNT(*) as percent_recovered
#FROM airports,flights WHERE flights.depdelay > 15 AND airports.airport_id =
flights.DestAirportID GROUP BY airports.name ORDER BY percent_recovered DESC
LIMIT 5

```

```

subquery = (flights_df
            .join(airports_df.select("airport_id", "name"),
col("DestAirportID") == col("airport_id"))
            .filter(col("depdelay") > 15)
            .groupBy("name")
            .agg(count(when(col("arrdelay") <= 15, 1)).alias("recovered"),
count("*").alias("total"))
            .select("name", (col("recovered") /
col("total")).alias("percent_recovered"))
            )
display(subquery.orderBy("percent_recovered", ascending=False).limit(5))

```

O Tambien:

```

display(flights_df
        .join(airports_df.select("airport_id", "name"), col("DestAirportID")
== col("airport_id"))
        .filter(col("depdelay") > 15)
        .groupBy("name")
        .agg(count(when(col("arrdelay") <= 15, 1)).alias("recovered"),
count("*").alias("total"))
        .select("name", (col("recovered") /
col("total")).alias("percent_recovered"))
        .orderBy("percent_recovered", ascending=False)
        .limit(5)
        )

```

Ejercicio 5

Sube el fichero notas.txt a DBFS y calcula la nota media de cada alumno usando Spark (celdas en Python).

Nota: El fichero notas.txt no tiene estructura de tabla y no vas a poderlo procesar como un Dataframe. Lo recomendable es usar transformaciones y acciones con RDD. En la [guía de programación de RDD](#) puedes aprender los conceptos básicos.

Nota2: La implementación debe tener en cuenta que el fichero notas.txt es un simple ejemplo de un fichero más grande que contendría millones de filas. Sin embargo el número de alumnos será de unos cientos.

```

# Comprobamos que notas.txt está en DBFS
files = dbutils.fs.ls("dbfs:/FileStore")
display (files)

```

```

# Mostramos el contenido
lines = sc.textFile("dbfs:/FileStore/notas.txt")
f = lines.collect()
for x in f:
    print (x)

```

```

# Leemos el archivo
texts = sc.textFile("dbfs:/FileStore/notas.txt")

# Separamos lineas
lines = texts.map(lambda s: s.split())
# muestra este formato: ['pedro', '6', '7']
# Separamos datos en tupla
students= lines.map(lambda x: (x[0], list(map(int, x[1:]))))
#for element in students.collect():
# print(element)
#print(students)
# muestra este formato: ('pedro', [6, 7])

# Convertimos los pares en una lista con el nombre y la nota
list_pairs = students.flatMapValues(lambda x: [(x[i],) for i in
range(len(x))])
#for element in list_pairs.collect():
# print(element)
#print(list_pairs)
# muestra este formato: ('pedro', (6,))('pedro', (7,))

# En el reduce recogemos los datos por nombre y valores
union_data = list_pairs.reduceByKey(lambda x, y: x + y)
#for element in union_data .collect():
# print(element)
#print(union_data )
# muestra este formato:('pedro', (6, 7, 8, 1, 3))

# Sumamos los datos de las notas y los dividimos entre su longitud para la
media
sums = union_data.map(lambda x: (x[0], float(sum(x[1]))/len(x[1])))
media = sums.collect()
print(media)

```

<https://www.youtube.com/watch?v=VonQnrSxQWA>

<https://github.com/databricks-academy/apache-spark-programming-with-databricks/releases>

<https://learn.microsoft.com/es-es/azure/databricks/getting-started/dataframes-python>

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.withColumnRenamed.html#pyspark.sql.DataFrame.withColumnRenamed>

<https://sparkbyexamples.com/spark/using-groupby-on-dataframe/>

<https://sparkbyexamples.com/pyspark/pyspark-flatmap-transformation/>

<https://databricks-prod->

<cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3137082781873852/3704545280501166/1264763342038607/latest.html>

<https://spark.apache.org/docs/latest/sql-ref-syntax-qry-select.html>

<https://docs.databricks.com/getting-started/dataframes-python.html>

<https://docs.databricks.com/dev-tools/databricks-utils.html#file-system-utility-dbutilsfs&language-python>

<https://spark.apache.org/docs/latest/rdd-programming-guide.html#resilient-distributed-datasets-rdds>