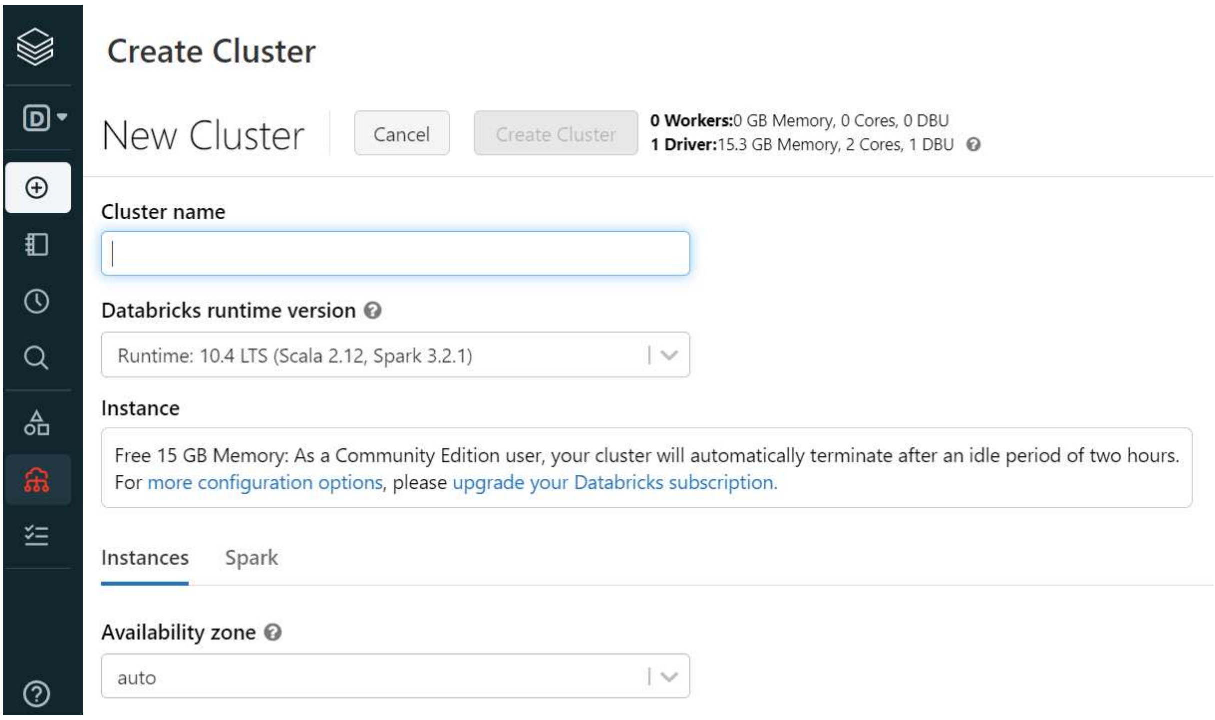


Create Cluster

New Cluster

0 Workers:0 GB Memory, 0 Cores, 0 DBU
1 Driver:15.3 GB Memory, 2 Cores, 1 DBU

Cluster name

Databricks runtime version

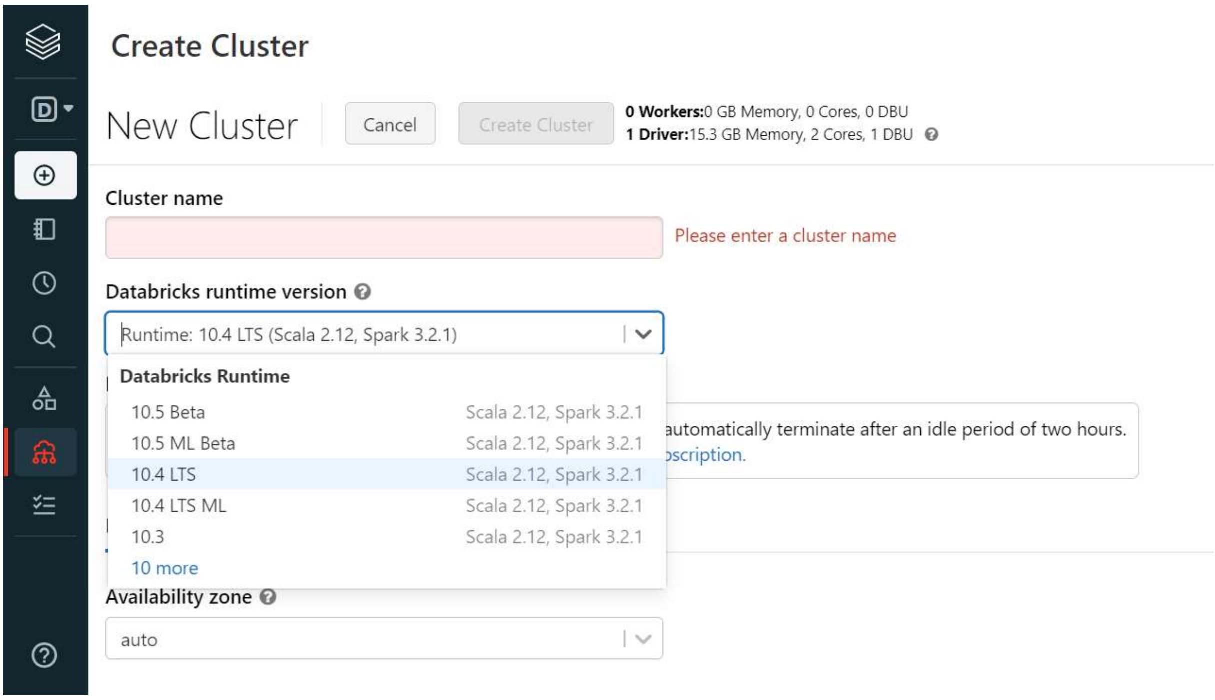
Runtime: 10.4 LTS (Scala 2.12, Spark 3.2.1)

Instance

Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Availability zone

auto



Create Cluster

New Cluster

0 Workers:0 GB Memory, 0 Cores, 0 DBU
1 Driver:15.3 GB Memory, 2 Cores, 1 DBU

Cluster name

Please enter a cluster name

Databricks runtime version

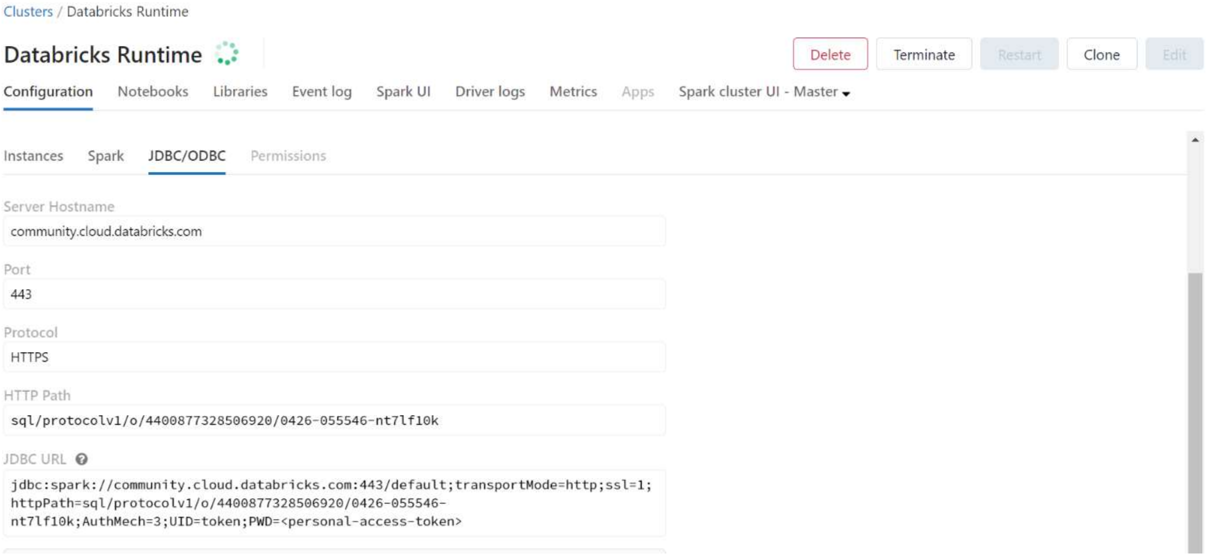
Runtime: 10.4 LTS (Scala 2.12, Spark 3.2.1)

Databricks Runtime
10.5 Beta Scala 2.12, Spark 3.2.1
10.5 ML Beta Scala 2.12, Spark 3.2.1
10.4 LTS Scala 2.12, Spark 3.2.1
10.4 LTS ML Scala 2.12, Spark 3.2.1
10.3 Scala 2.12, Spark 3.2.1
10 more

automatically terminate after an idle period of two hours.
bscription.

Availability zone

auto



Server Hostname
community.cloud.databricks.com

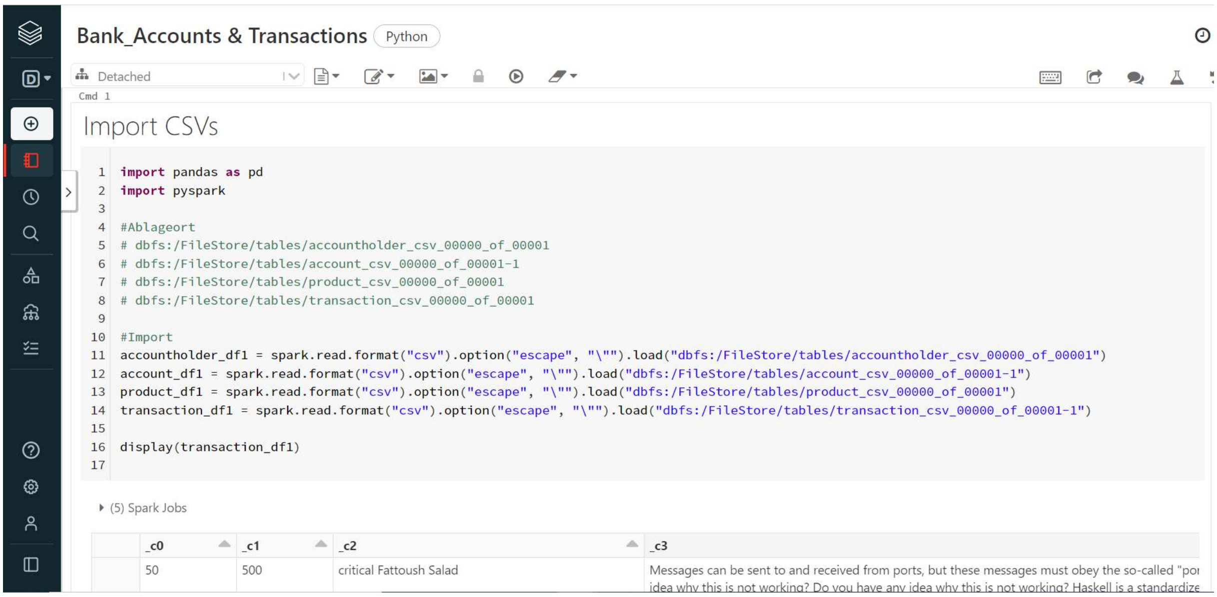
Port
443

Protocol
HTTPS

HTTP Path
sql/protocolv1/o/4408877328506920/0426-055546-nt7lf10k

JDBC URL

```
jdbc:spark://community.cloud.databricks.com:443/default;transportMode=http;ssl=1;
httpPath=sql/protocolv1/o/4408877328506920/0426-055546-
nt7lf10k;AuthMech=3;UID=token;PWD=<personal-access-token>
```

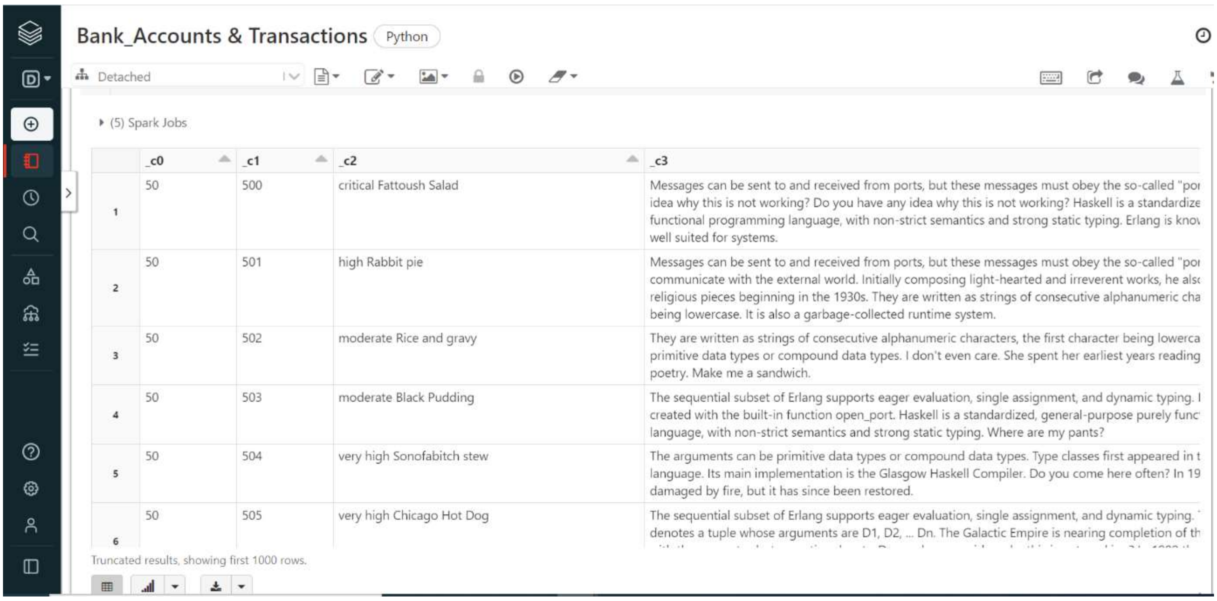


Import CSVs

```python
import pandas as pd
import pyspark

#Ablageort
# dbfs:/FileStore/tables/accountholder_csv_00000_of_00001
# dbfs:/FileStore/tables/account_csv_00000_of_00001-1
# dbfs:/FileStore/tables/product_csv_00000_of_00001
# dbfs:/FileStore/tables/transaction_csv_00000_of_00001

#Import
accountholder_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/accountholder_csv_00000_of_00001")
account_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/account_csv_00000_of_00001-1")
product_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/product_csv_00000_of_00001")
transaction_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/transaction_csv_00000_of_00001-1")

display(transaction_df1)

```

▸ (5) Spark Jobs

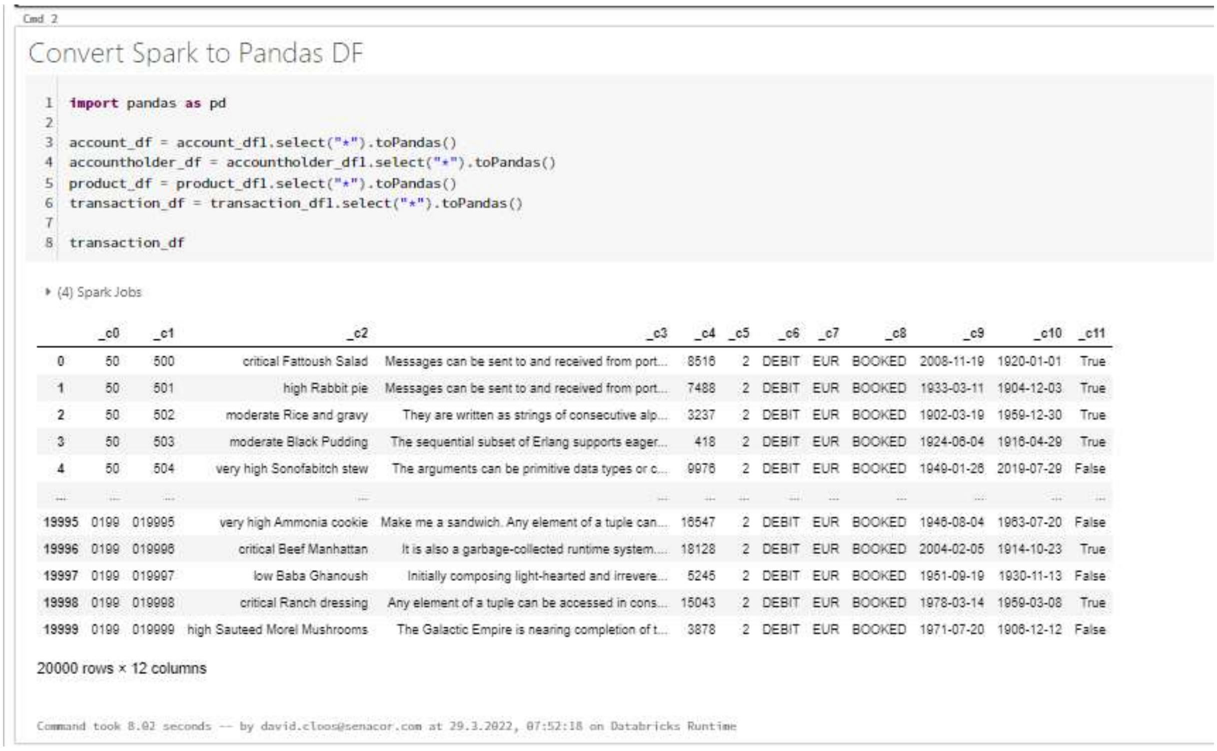
	_c0	_c1	_c2	_c3
	50	500	critical Fattoush Salad	Messages can be sent to and received from ports, but these messages must obey the so-called "poı idea why this is not working? Do you have any idea why this is not working? Haskell is a standardize



▸ (5) Spark Jobs

	_c0	_c1	_c2	_c3
1	50	500	critical Fattoush Salad	Messages can be sent to and received from ports, but these messages must obey the so-called "poı idea why this is not working? Do you have any idea why this is not working? Haskell is a standardize functional programming language, with non-strict semantics and strong static typing. Erlang is knov well suited for systems.
2	50	501	high Rabbit pie	Messages can be sent to and received from ports, but these messages must obey the so-called "poı communicate with the external world. Initially composing light-hearted and irreverent works, he alsc religious pieces beginning in the 1930s. They are written as strings of consecutive alphanumeric cha being lowercase. It is also a garbage-collected runtime system.
3	50	502	moderate Rice and gravy	They are written as strings of consecutive alphanumeric characters, the first character being lowerca primitive data types or compound data types. I don't even care. She spent her earliest years reading poetry. Make me a sandwich.
4	50	503	moderate Black Pudding	The sequential subset of Erlang supports eager evaluation, single assignment, and dynamic typing. I created with the built-in function open_port. Haskell is a standardized, general-purpose purely func language, with non-strict semantics and strong static typing. Where are my pants?
5	50	504	very high Sonofabitch stew	The arguments can be primitive data types or compound data types. Type classes first appeared in t language. Its main implementation is the Glasgow Haskell Compiler. Do you come here often? In 19 damaged by fire, but it has since been restored.
6	50	505	very high Chicago Hot Dog	The sequential subset of Erlang supports eager evaluation, single assignment, and dynamic typing. denotes a tuple whose arguments are D1, D2, ... Dn. The Galactic Empire is nearing completion of th

Truncated results, showing first 1000 rows.



Cmd 2

Convert Spark to Pandas DF

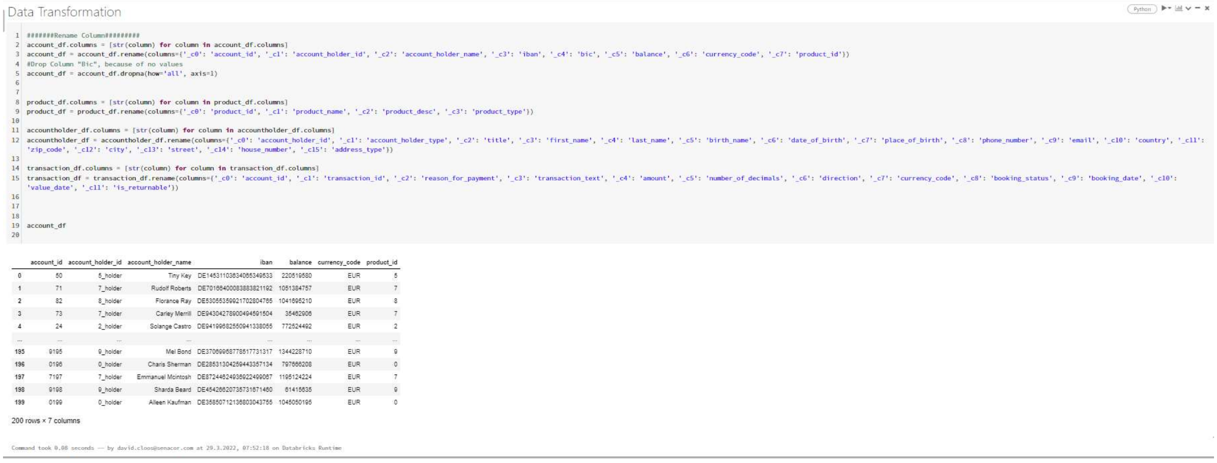
```
import pandas as pd

account_df = account_dfl.select("*").toPandas()
accountholder_df = accountholder_dfl.select("*").toPandas()
product_df = product_dfl.select("*").toPandas()
transaction_df = transaction_dfl.select("*").toPandas()

transaction_df
```

▸ (4) Spark Jobs

	_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	_c10	_c11
0	50	500	critical Fattoush Salad	Messages can be sent to and received from port...	8516	2	DEBIT	EUR	BOOKED	2008-11-19	1920-01-01	True
1	50	501	high Rabbit pie	Messages can be sent to and received from port...	7488	2	DEBIT	EUR	BOOKED	1933-03-11	1904-12-03	True
2	50	502	moderate Rice and gravy	They are written as strings of consecutive alp...	3237	2	DEBIT	EUR	BOOKED	1902-03-19	1959-12-30	True
3	50	503	moderate Black Pudding	The sequential subset of Erlang supports eager...	418	2	DEBIT	EUR	BOOKED	1924-06-04	1916-04-29	True
4	50	504	very high Sonofabitch stew	The arguments can be primitive data types or c...	9976	2	DEBIT	EUR	BOOKED	1949-01-26	2019-07-29	False
...	...	...	...	...	...	...	...	...	...	...	...	...
19995	0199	019995	very high Ammonia cookie	Make me a sandwich. Any element of a tuple can...	16547	2	DEBIT	EUR	BOOKED	1946-08-04	1963-07-20	False
19996	0199	019996	critical Beef Manhattan	It is also a garbage-collected runtime system....	18128	2	DEBIT	EUR	BOOKED	2004-02-05	1914-10-23	True
19997	0199	019997	low Baba Ghanoush	Initially composing light-hearted and irrevere...	5245	2	DEBIT	EUR	BOOKED	1951-09-19	1930-11-13	False
19998	0199	019998	critical Ranch dressing	Any element of a tuple can be accessed in cons...	15043	2	DEBIT	EUR	BOOKED	1978-03-14	1959-03-08	True
19999	0199	019999	high Sauteed Morel Mushrooms	The Galactic Empire is nearing completion of t...	3878	2	DEBIT	EUR	BOOKED	1971-07-20	1908-12-12	False

20000 rows × 12 columns

```
Command took 8.02 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```



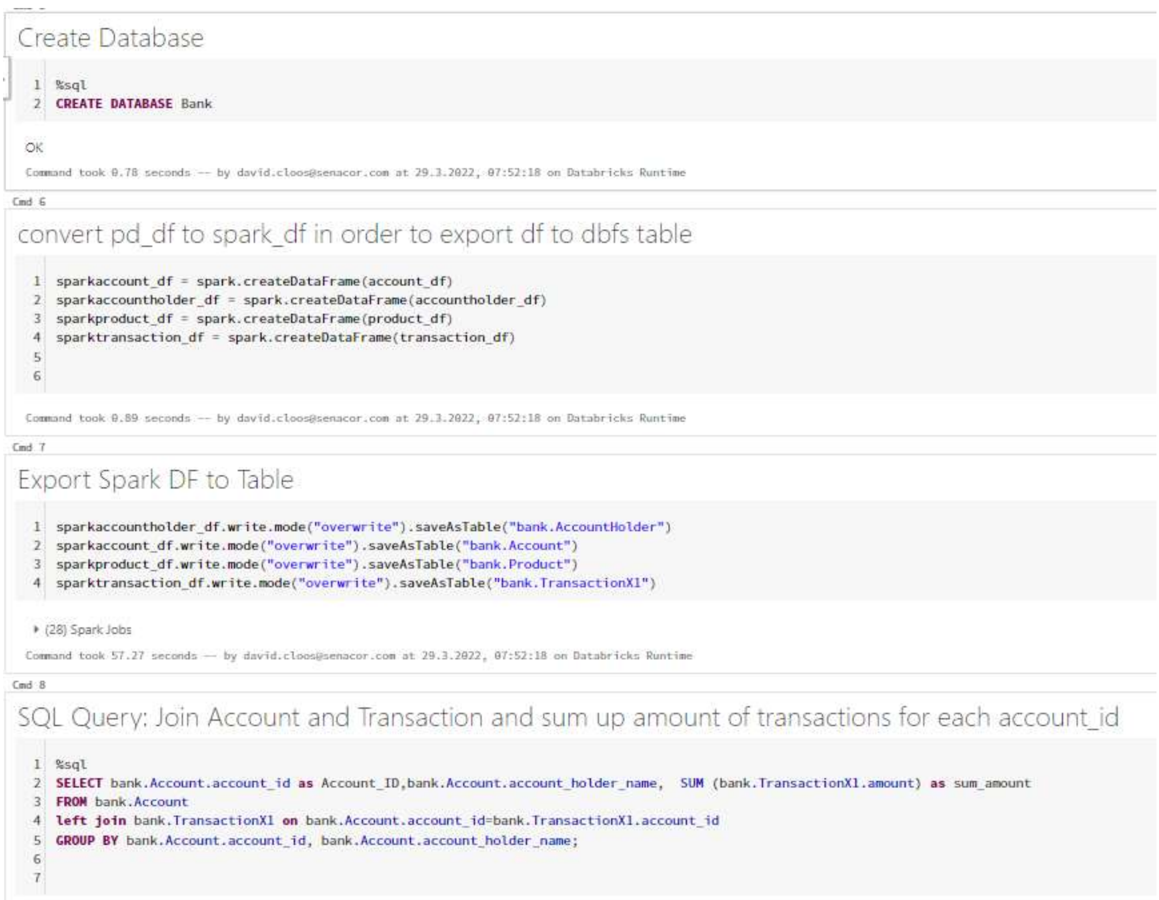
Data Transformation

```
#######Rename Column#########
account_df.columns = [str(column) for column in account_df.columns]
account_df = account_df.rename(columns={'_c0': 'account_id', '_c1': 'account_holder_id', '_c2': 'account_holder_name', '_c3': 'iban', '_c4': 'bic', '_c5': 'balance', '_c6': 'currency_code', '_c7': 'product_id'})
#Drop Column "Bic", because of no values
account_df = account_df.dropna(how='all', axis=1)


product_df.columns = [str(column) for column in product_df.columns]
product_df = product_df.rename(columns={'_c0': 'product_id', '_c1': 'product_name', '_c2': 'product_desc', '_c3': 'product_type'})

accountholder_df.columns = [str(column) for column in accountholder_df.columns]
accountholder_df = accountholder_df.rename(columns={'_c0': 'account_holder_id', '_c1': 'account_holder_type', '_c2': 'title', '_c3': 'first_name', '_c4': 'last_name', '_c5': 'birth_name', '_c6': 'date_of_birth', '_c7': 'place_of_birth', '_c8': 'phone_number', '_c9': 'email', '_c10': 'country', '_c11': 'zip_code', '_c12': 'city', '_c13': 'street', '_c14': 'house_number', '_c15': 'address_type'})

transaction_df.columns = [str(column) for column in transaction_df.columns]
transaction_df = transaction_df.rename(columns={'_c0': 'account_id', '_c1': 'transaction_id', '_c2': 'reason_for_payment', '_c3': 'transaction_text', '_c4': 'amount', '_c5': 'number_of_decimals', '_c6': 'direction', '_c7': 'currency_code', '_c8': 'booking_status', '_c9': 'booking_date', '_c10': 'value_date', '_c11': 'is_returnable'})




account_df
```

	account_id	account_holder_id	account_holder_name	iban	balance	currency_code	product_id
0	50	5_holder	Tiny Key	DE14531103634065349533	220519580	EUR	5
1	71	7_holder	Rudolf Roberts	DE70166400083883821192	1051384757	EUR	7
2	82	8_holder	Florance Ray	DE53055359921702804765	1041695210	EUR	8
3	73	7_holder	Carley Merrill	DE94304278900494591504	35462906	EUR	7
4	24	2_holder	Solange Castro	DE94199682550941338055	772524492	EUR	2
...	...	...	...	...	...	...	...
195	9195	9_holder	Mel Bond	DE37069968778517731317	1344228710	EUR	9
196	0196	0_holder	Charis Sherman	DE28531304259443357134	797666208	EUR	0
197	7197	7_holder	Emmanuel Mcintosh	DE87244624936922499067	1195124224	EUR	7
198	9198	9_holder	Sharda Beard	DE45426620735731671460	61415635	EUR	9
199	0199	0_holder	Alleen Kaufman	DE35850712136803043755	1045050195	EUR	0

200 rows × 7 columns

```
Command took 0.08 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```



Create Database

```
%sql
CREATE DATABASE Bank
```

```
OK
Command took 0.78 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```

Cmd 6

convert pd_df to spark_df in order to export df to dbfs table

```
sparkaccount_df = spark.createDataFrame(account_df)
sparkaccountholder_df = spark.createDataFrame(accountholder_df)
sparkproduct_df = spark.createDataFrame(product_df)
sparktransaction_df = spark.createDataFrame(transaction_df)


```

```
Command took 0.89 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```

Cmd 7

Export Spark DF to Table

```
sparkaccountholder_df.write.mode("overwrite").saveAsTable("bank.AccountHolder")
sparkaccount_df.write.mode("overwrite").saveAsTable("bank.Account")
sparkproduct_df.write.mode("overwrite").saveAsTable("bank.Product")
sparktransaction_df.write.mode("overwrite").saveAsTable("bank.TransactionX1")
```

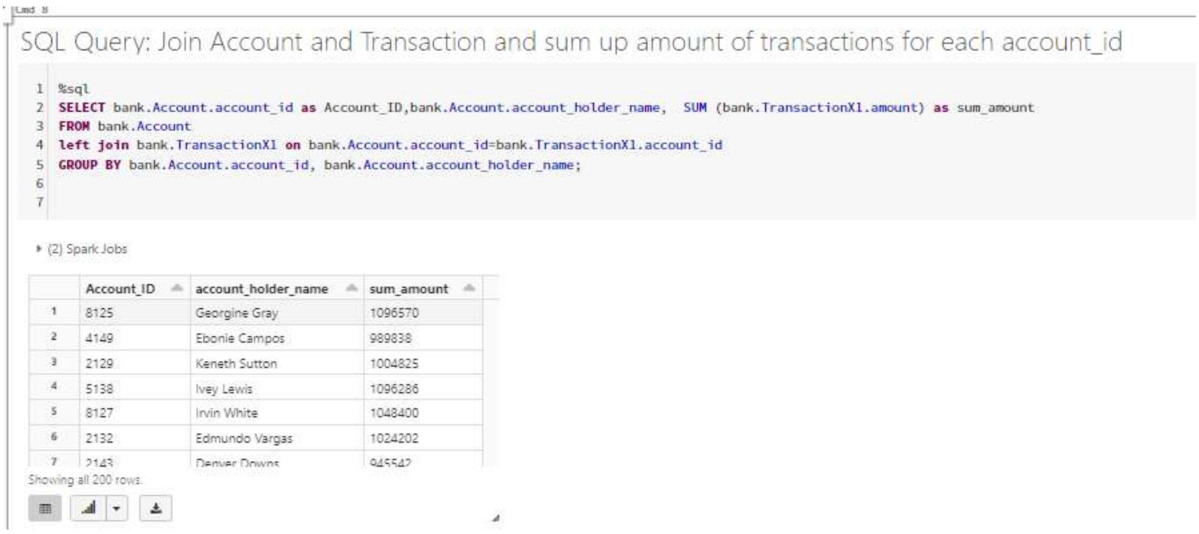
▸ (28) Spark Jobs

```
Command took 57.27 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```

Cmd 8

SQL Query: Join Account and Transaction and sum up amount of transactions for each account_id

```
%sql
SELECT bank.Account.account_id as Account_ID,bank.Account.account_holder_name,  SUM (bank.TransactionX1.amount) as sum_amount
FROM bank.Account
left join bank.TransactionX1 on bank.Account.account_id=bank.TransactionX1.account_id
GROUP BY bank.Account.account_id, bank.Account.account_holder_name;


```



Cmd 8

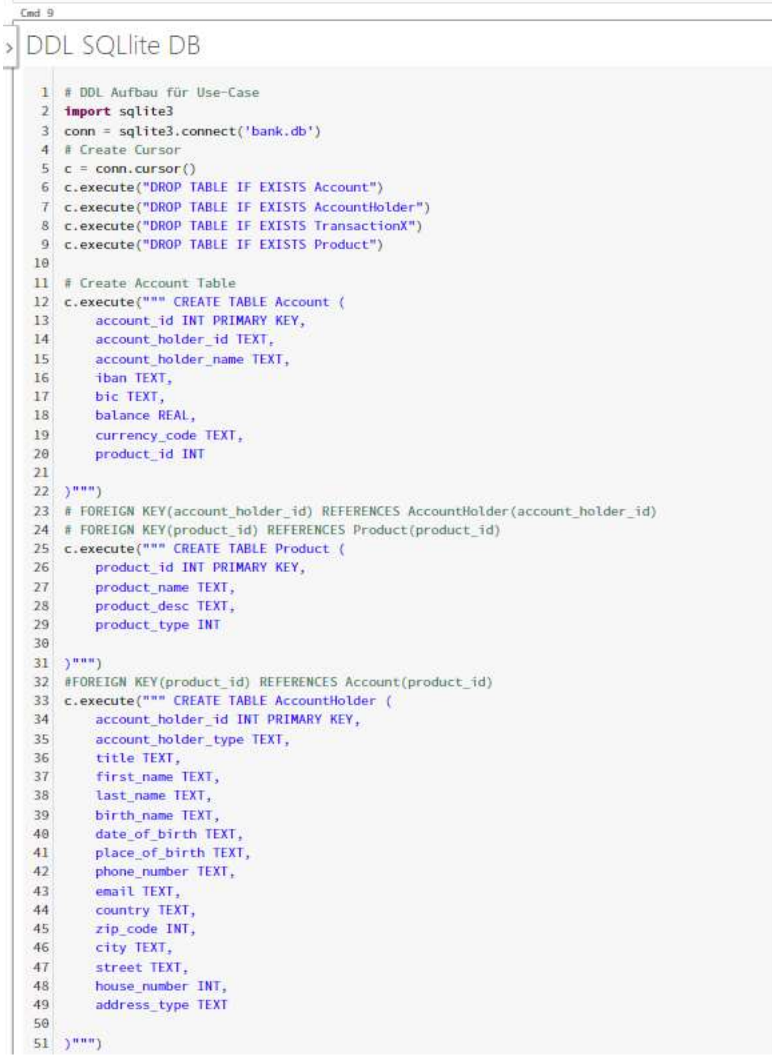
SQL Query: Join Account and Transaction and sum up amount of transactions for each account_id

```
%sql
SELECT bank.Account.account_id as Account_ID,bank.Account.account_holder_name,  SUM (bank.TransactionX1.amount) as sum_amount
FROM bank.Account
left join bank.TransactionX1 on bank.Account.account_id=bank.TransactionX1.account_id
GROUP BY bank.Account.account_id, bank.Account.account_holder_name;


```

▸ (2) Spark Jobs

	Account_ID	account_holder_name	sum_amount
1	8125	Georgine Gray	1096570
2	4149	Ebonie Campos	989838
3	2129	Keneth Sutton	1004825
4	5138	Ivey Lewis	1096286
5	8127	Irvin White	1048400
6	2132	Edmundo Vargas	1024202
7	2143	Denver Downs	945542

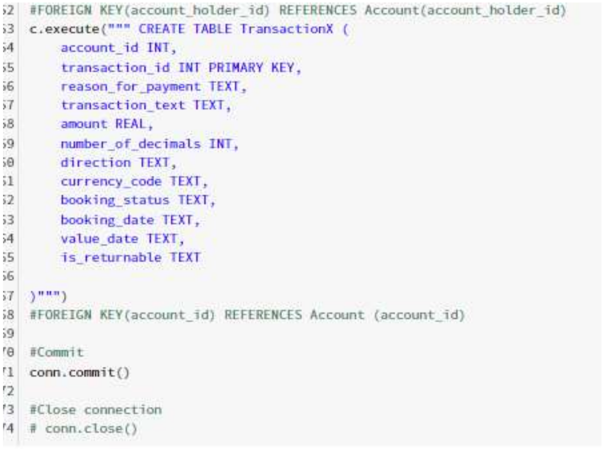
Showing all 200 rows.



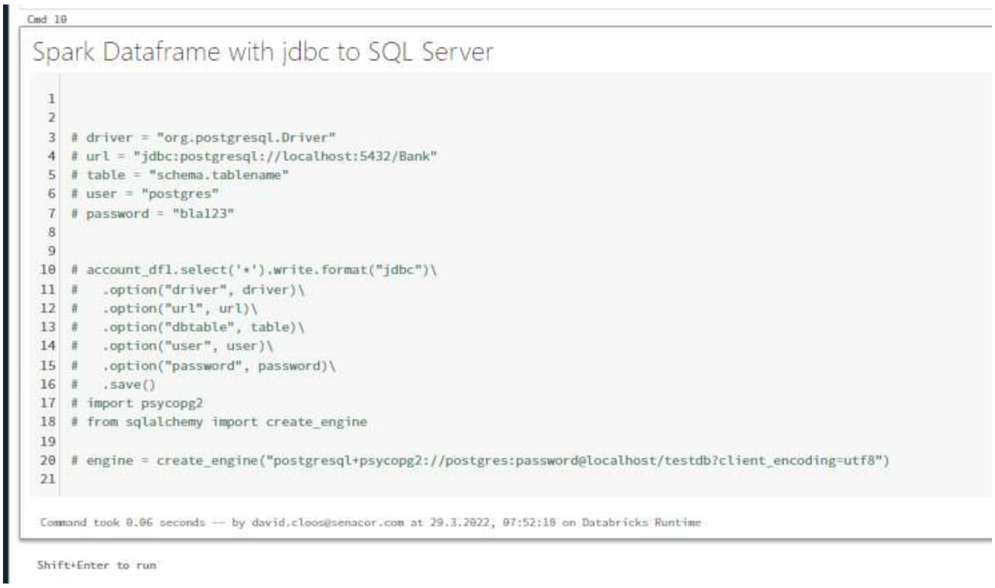
Cmd 9

DDL SQLlite DB

```python
# DDL Aufbau für Use-Case
import sqlite3
conn = sqlite3.connect('bank.db')
# Create Cursor
c = conn.cursor()
c.execute("DROP TABLE IF EXISTS Account")
c.execute("DROP TABLE IF EXISTS AccountHolder")
c.execute("DROP TABLE IF EXISTS TransactionX")
c.execute("DROP TABLE IF EXISTS Product")

# Create Account Table
c.execute(""" CREATE TABLE Account (
    account_id INT PRIMARY KEY,
    account_holder_id TEXT,
    account_holder_name TEXT,
    iban TEXT,
    bic TEXT,
    balance REAL,
    currency_code TEXT,
    product_id INT

)""")
# FOREIGN KEY(account_holder_id) REFERENCES AccountHolder(account_holder_id)
# FOREIGN KEY(product_id) REFERENCES Product(product_id)
c.execute(""" CREATE TABLE Product (
    product_id INT PRIMARY KEY,
    product_name TEXT,
    product_desc TEXT,
    product_type INT

)""")
#FOREIGN KEY(product_id) REFERENCES Account(product_id)
c.execute(""" CREATE TABLE AccountHolder (
    account_holder_id INT PRIMARY KEY,
    account_holder_type TEXT,
    title TEXT,
    first_name TEXT,
    last_name TEXT,
    birth_name TEXT,
    date_of_birth TEXT,
    place_of_birth TEXT,
    phone_number TEXT,
    email TEXT,
    country TEXT,
    zip_code INT,
    city TEXT,
    street TEXT,
    house_number INT,
    address_type TEXT

)""")
```



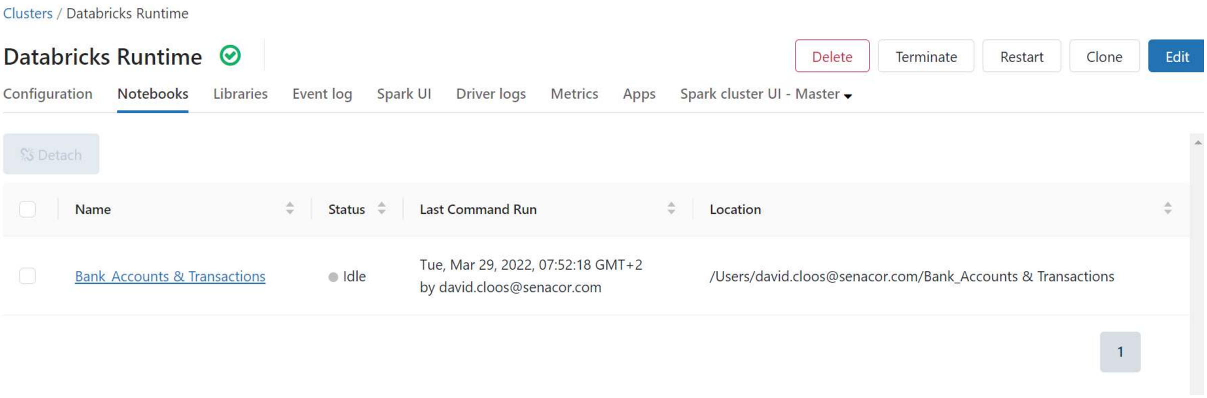
```python
#FOREIGN KEY(account_holder_id) REFERENCES Account(account_holder_id)
c.execute(""" CREATE TABLE TransactionX (
    account_id INT,
    transaction_id INT PRIMARY KEY,
    reason_for_payment TEXT,
    transaction_text TEXT,
    amount REAL,
    number_of_decimals INT,
    direction TEXT,
    currency_code TEXT,
    booking_status TEXT,
    booking_date TEXT,
    value_date TEXT,
    is_returnable TEXT

)""")
#FOREIGN KEY(account_id) REFERENCES Account (account_id)

#Commit
conn.commit()

#Close connection
# conn.close()
```


Cmd 10

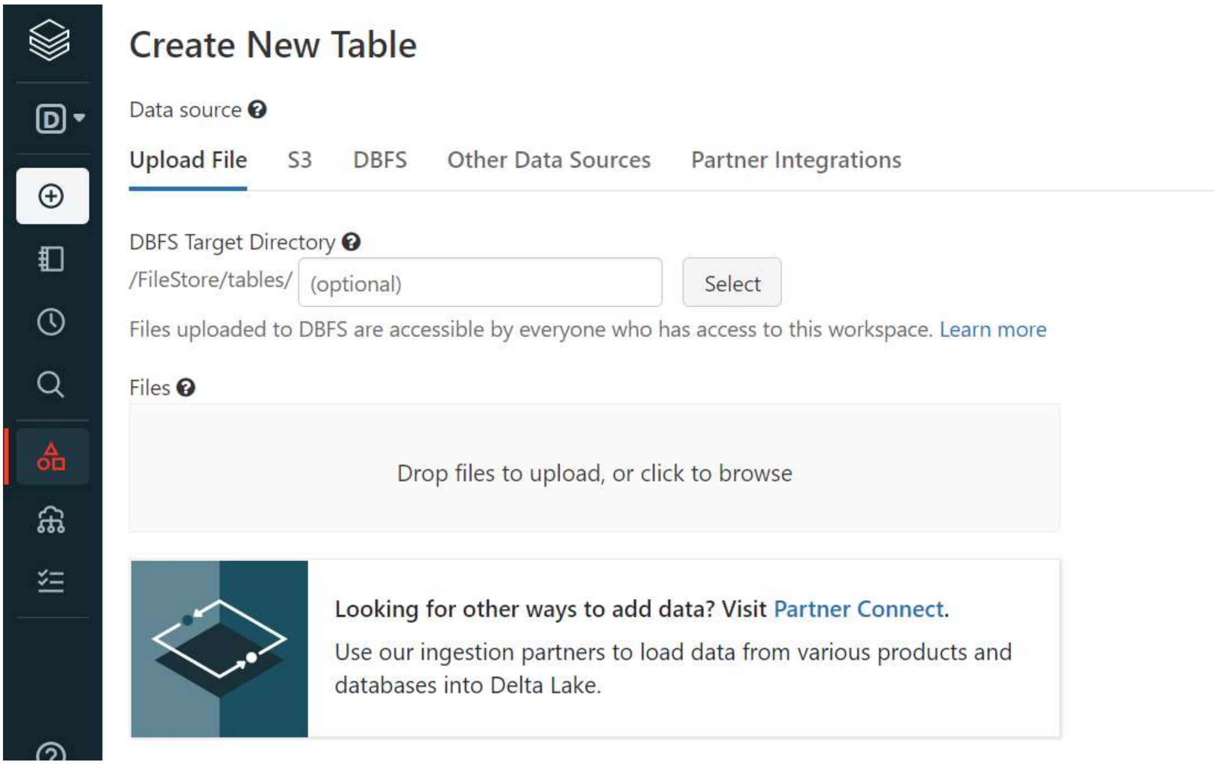
Spark Dataframe with jdbc to SQL Server

```


# driver = "org.postgresql.Driver"
# url = "jdbc:postgresql://localhost:5432/Bank"
# table = "schema.tablename"
# user = "postgres"
# password = "bla123"


# account_df1.select('*').write.format("jdbc")\
#    .option("driver", driver)\
#    .option("url", url)\
#    .option("dbtable", table)\
#    .option("user", user)\
#    .option("password", password)\
#    .save()
# import psycopg2
# from sqlalchemy import create_engine

# engine = create_engine("postgresql+psycopg2://postgres:password@localhost/testdb?client_encoding=utf8")

```

```
Command took 0.06 seconds -- by david.cloos@senacor.com at 29.3.2022, 07:52:18 on Databricks Runtime
```



□	Name	Status	Last Command Run	Location
□	Bank_Accounts & Transactions	● Idle	Tue, Mar 29, 2022, 07:52:18 GMT+2 by david.cloos@senacor.com	/Users/david.cloos@senacor.com/Bank_Accounts & Transactions



Create New Table

Data source

DBFS Target Directory

/FileStore/tables/ (optional)

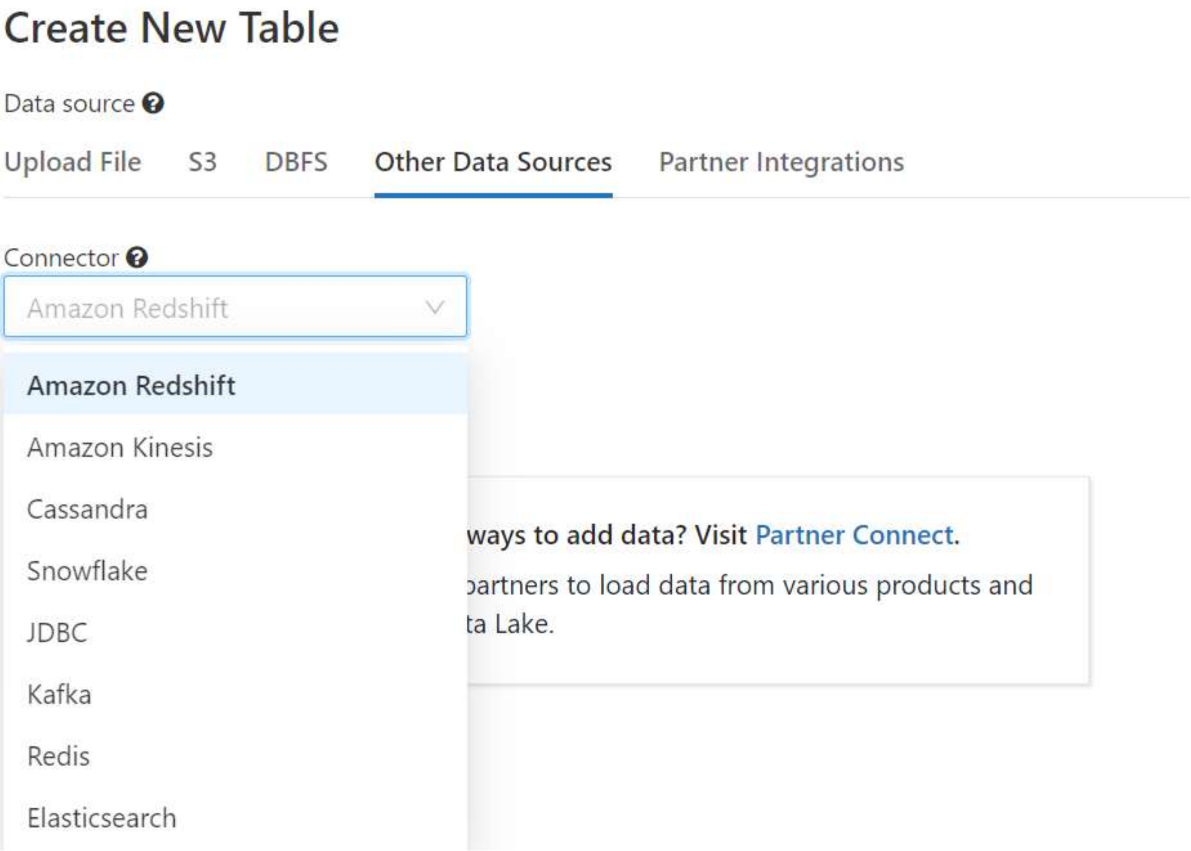
Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

Drop files to upload, or click to browse

Looking for other ways to add data? Visit Partner Connect.

Use our ingestion partners to load data from various products and databases into Delta Lake.



Create New Table

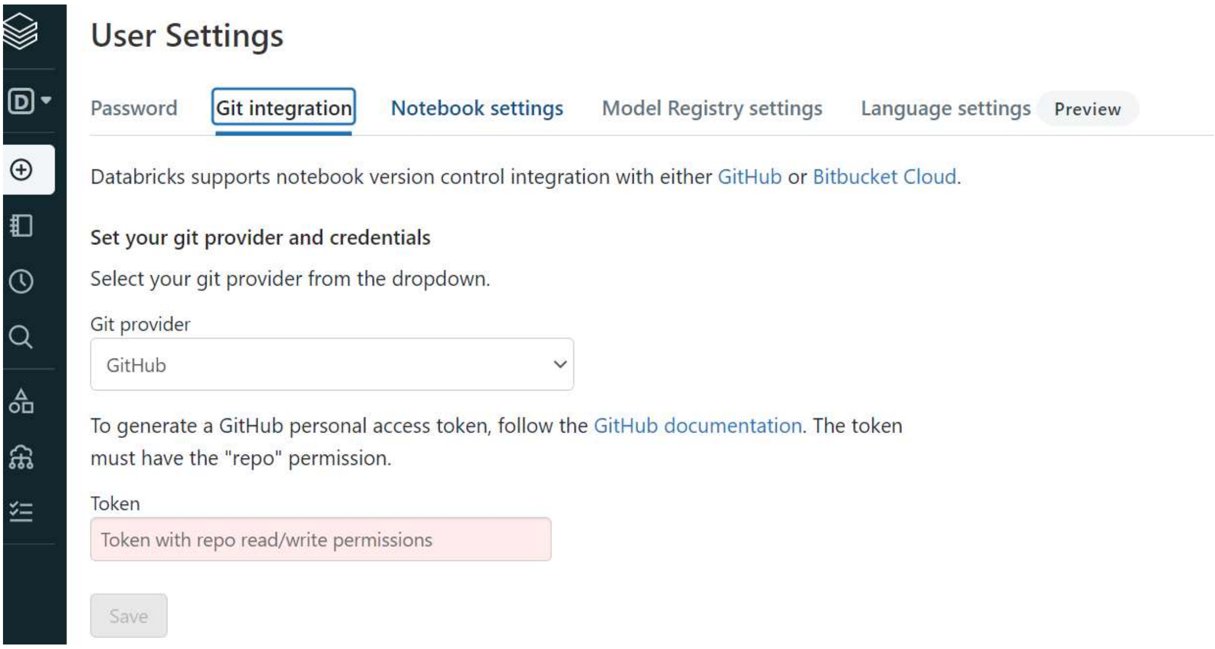
Data source

Connector

Amazon Redshift

ways to add data? Visit Partner Connect.

partners to load data from various products and ta Lake.



Databricks supports notebook version control integration with either GitHub or Bitbucket Cloud.

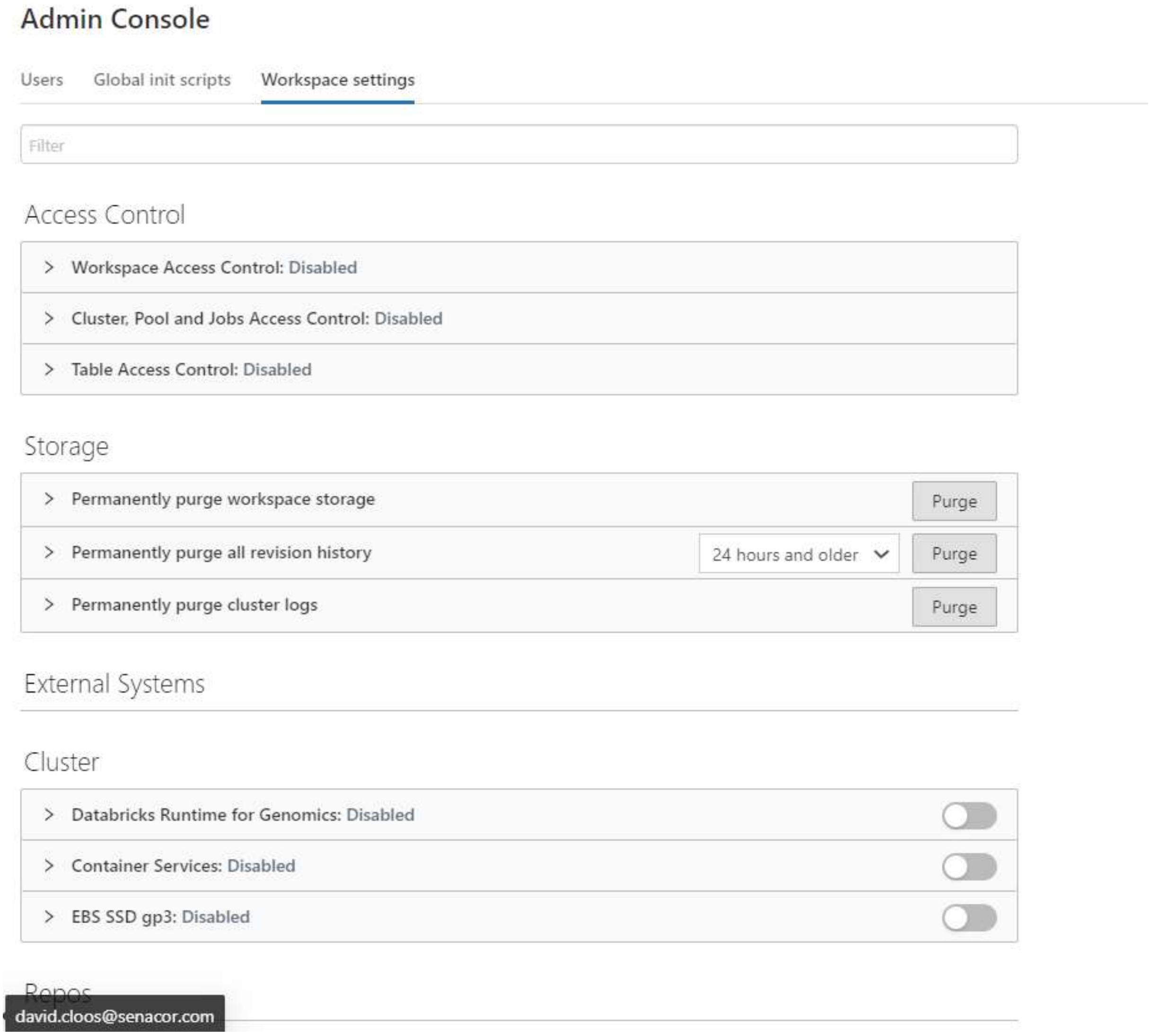
Set your git provider and credentials

Select your git provider from the dropdown.

Git provider: GitHub

To generate a GitHub personal access token, follow the GitHub documentation. The token must have the "repo" permission.

Token: Token with repo read/write permissions



Access Control

- Workspace Access Control: Disabled
- Cluster, Pool and Jobs Access Control: Disabled
- Table Access Control: Disabled

Storage

- Permanently purge workspace storage
- Permanently purge all revision history — 24 hours and older
- Permanently purge cluster logs

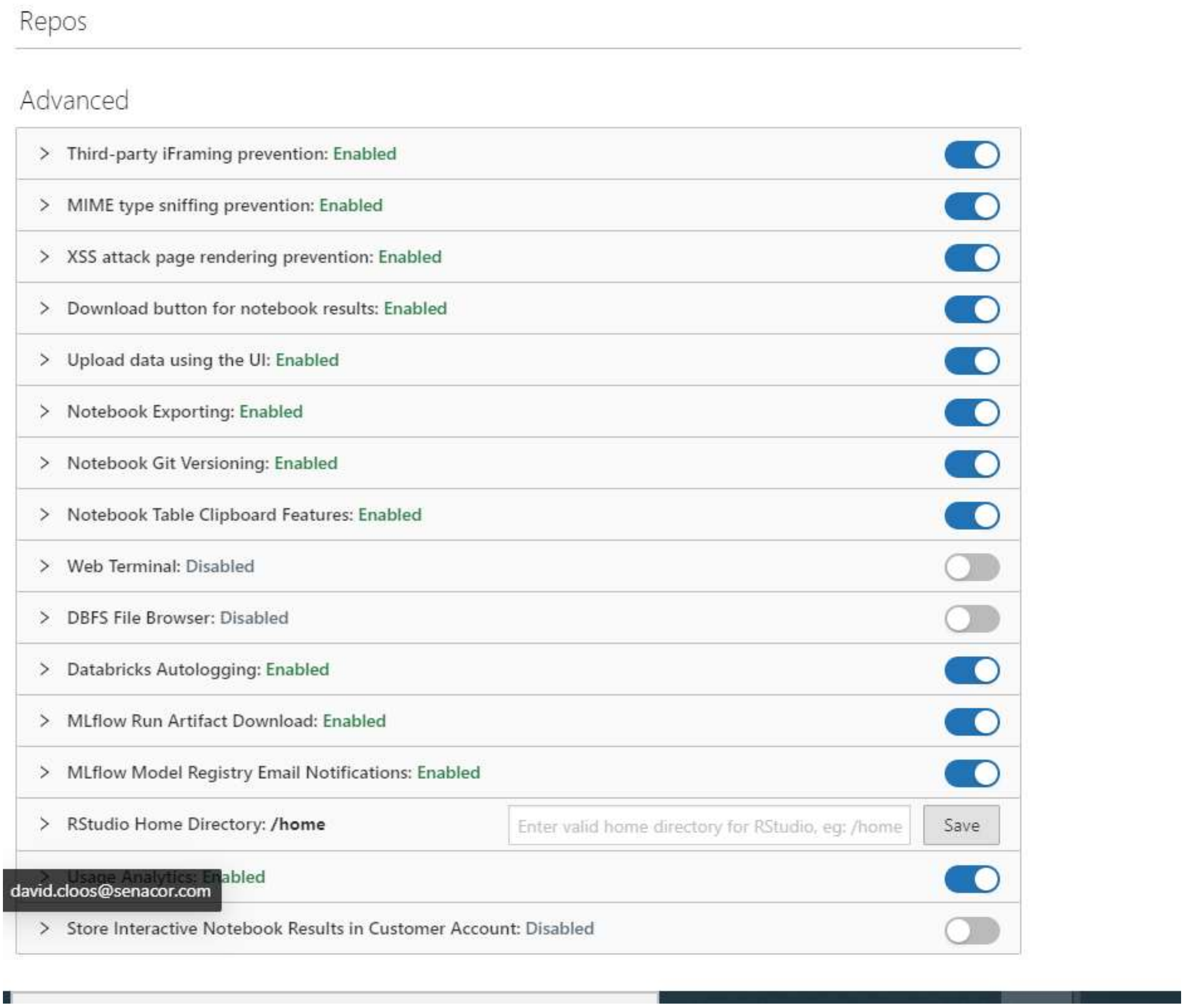
External Systems

Cluster

- Databricks Runtime for Genomics: Disabled
- Container Services: Disabled
- EBS SSD gp3: Disabled

Repos

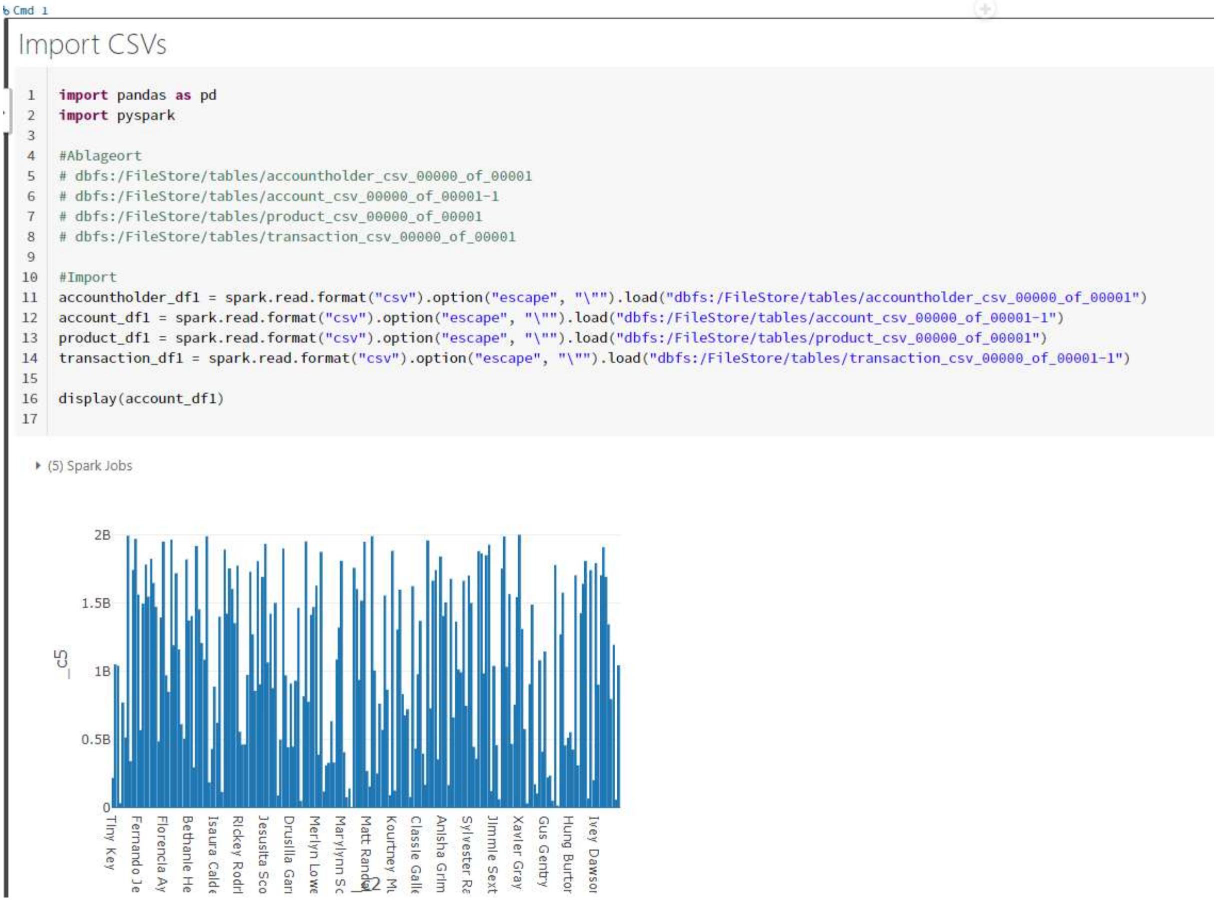
david.cloos@senacor.com



Repos

Advanced

- Third-party iFraming prevention: Enabled
- MIME type sniffing prevention: Enabled
- XSS attack page rendering prevention: Enabled
- Download button for notebook results: Enabled
- Upload data using the UI: Enabled
- Notebook Exporting: Enabled
- Notebook Git Versioning: Enabled
- Notebook Table Clipboard Features: Enabled
- Web Terminal: Disabled
- DBFS File Browser: Disabled
- Databricks Autologging: Enabled
- MLflow Run Artifact Download: Enabled
- MLflow Model Registry Email Notifications: Enabled
- RStudio Home Directory: **/home**
- Usage Analytics: Enabled
- Store Interactive Notebook Results in Customer Account: Disabled



Cmd 1

Import CSVs

```
import pandas as pd
import pyspark

#Ablageort
# dbfs:/FileStore/tables/accountholder_csv_00000_of_00001
# dbfs:/FileStore/tables/account_csv_00000_of_00001-1
# dbfs:/FileStore/tables/product_csv_00000_of_00001
# dbfs:/FileStore/tables/transaction_csv_00000_of_00001

#Import
accountholder_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/accountholder_csv_00000_of_00001")
account_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/account_csv_00000_of_00001-1")
product_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/product_csv_00000_of_00001")
transaction_df1 = spark.read.format("csv").option("escape", "\"").load("dbfs:/FileStore/tables/transaction_csv_00000_of_00001-1")

display(account_df1)

```

(5) Spark Jobs