Spark on Colab with UI

February 18, 2025

[1]:
```
# prompt: create a spark session and load a loal data via parallelize into an rdd.

from pyspark.sql import SparkSession
from pyspark.sql.types import *

# Create a SparkSession
spark = SparkSession.builder \
    .appName("LoadLocalData") \
    .getOrCreate()
```

[2]:
```
spark
```

[2]:
```
<pyspark.sql.session.SparkSession at 0x7bd1eb12c550>
```

[ ]:
```

```

[ ]:
```
# Sample data
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35)]

# Create an RDD from the local data
rdd = spark.sparkContext.parallelize(data)

# Print the RDD contents
print("RDD Contents:")
for item in rdd.collect():
    print(item)

# Infer schema (optional)
schema = StructType([StructField("Name", StringType(), True),StructField("Age", IntegerType(), True)])
df = spark.createDataFrame(rdd, schema)

# Show DataFrame (optional)
print("\nDataFrame Contents:")
df.show()
```

[ ]:
```
df.collect()
```

[ ]:
```
[Row(Name='Alice', Age=25),
 Row(Name='Bob', Age=30),
 Row(Name='Charlie', Age=35)]
```

[ ]:
```
spark
```

[ ]:
```
<pyspark.sql.session.SparkSession at 0x7c1486910dd0>
```

[ ]:
```
!pip install -q pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.config('spark.ui.port', '4050').getOrCreate()

!wget -qnc https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
!unzip -n -q ngrok-stable-linux-amd64.zip
get_ipython().system_raw('./ngrok http 4050 &')
!sleep 5
!curl -s http://localhost:4040/api/tunnels | grep -Po 'public_url":"(?
 ↪=https)\K[^"]*'
```

[ ]:
```
# prompt: give me a way to access Spark UI on my laptop directly

from pyspark.sql import SparkSession
from pyspark.sql.types import *

# Create a SparkSession with UI settings
spark = SparkSession.builder \
    .appName("LoadLocalData") \
    .config("spark.ui.port", "4040") \
    .config("spark.driver.host", "localhost") \
    .getOrCreate()

# Sample data
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35)]

# Create an RDD from the local data
rdd = spark.sparkContext.parallelize(data)

# Print the RDD contents
print("RDD Contents:")
for item in rdd.collect():
    print(item)

# Infer schema (optional)
schema = StructType([StructField("Name", StringType(), True),␣
 ↪StructField("Age", IntegerType(), True)])
df = spark.createDataFrame(rdd, schema)
```

```
# Show DataFrame (optional)
print("\nDataFrame Contents:")
df.show()

# Print the Spark UI URL
print(f"\nSpark UI URL: http://localhost:4040")

spark
```

```
RDD Contents:
('Alice', 25)
('Bob', 30)
('Charlie', 35)

DataFrame Contents:
+-------+---+
|   Name|Age|
+-------+---+
|  Alice| 25|
|    Bob| 30|
|Charlie| 35|
+-------+---+


Spark UI URL: http://localhost:4040
```

[ ]:
```
<pyspark.sql.session.SparkSession at 0x7ae01e31a690>
```

[ ]:
```
spark.stop()
```

[ ]:
```
from google.colab import output
output.serve_kernel_port_as_window(4040, path='/jobs/index.html')
```

```
Warning: This function may stop working due to changes in browser security.
Try `serve_kernel_port_as_iframe` instead.

<IPython.core.display.Javascript object>
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-
packages (3.5.4)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-
packages (from pyspark) (0.10.9.7)
```

[ ]:
```
from pyspark.sql import SparkSession
from pyspark.sql.types import *

# Create a SparkSession
spark = SparkSession.builder \
    .appName("LoadLocalData") \
    .getOrCreate()
```

[ ]:
```
spark
```

[ ]:
```
<pyspark.sql.session.SparkSession at 0x7a1fc9f52fd0>
```

[ ]:
```
spark.sql('create database customers_db')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
spark.sql('show databases').show()
```

```
+------------+
|   namespace|
+------------+
|customers_db|
|     default|
+------------+
```

[ ]:
```
spark.sql('show databases').filter("namespace like 'customers%'").show()
```

```
+------------+
|   namespace|
+------------+
|customers_db|
+------------+
```

[ ]:
```
spark.sql('use customers_db')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
spark.sql('show tables').show()
```

```
+---------+---------+-----------+
|namespace|tableName|isTemporary|
+---------+---------+-----------+
+---------+---------+-----------+
```

[ ]:
```
data = [(1, "Alice", "Mumbai", "2023-01-15", True),
(2, "Bob", "Delhi", "2023-03-25", False),
```

```
(3, "Charlie", "Chennai", "2023-05-10", True)]

columns = ["customer_id", "name", "city", "registration_date", "is_active"]
```

[ ]:
```
df = spark.createDataFrame(data, columns)
```

[ ]:
```
df.write.saveAsTable("customers_db.customers")
```

[ ]:
```
df.repartition(10).write.saveAsTable("customers_db.customers_2")
```

[ ]:
```
spark.sql('show tables').show()
```

```
+-----------+---------+-----------+
|   namespace|tableName|isTemporary|
+-----------+---------+-----------+
|customers_db|customers|      false|
+-----------+---------+-----------+
```

[ ]:
```
spark.sql('describe exhibtended customers').show(truncate=False)
```

```
+---------------------------+--------------------------------------------------
-----+-------+
|col_name                   |data_type
|comment|
+---------------------------+--------------------------------------------------
-----+-------+
|customer_id                |bigint
|NULL   |
|name                       |string
|NULL   |
|city                       |string
|NULL   |
|registration_date          |string
|NULL   |
|is_active                  |boolean
|NULL   |
|                           |
|       |
|# Detailed Table Information|
|       |
|Catalog                    |spark_catalog
|       |
|Database                   |customers_db
|       |
|Table                      |customers
|       |
|Created Time               |Sat Feb 01 04:05:16 UTC 2025
```

```
|       |
|Last Access                    |UNKNOWN
|       |
|Created By                     |Spark 3.5.4
|       |
|Type                           |MANAGED
|       |
|Provider                       |parquet
|       |
|Location                       |file:/content/spark-
warehouse/customers_db.db/customers|       |
+-----------------------------+-------------------------------------------------
-----+-------+
```

[ ]:
```
spark.sql('drop table customers')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
spark.sql('describe extended customers').show(truncate=False)
```

```
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
<ipython-input-19-df9068ea2f97> in <cell line: 0>()
----> 1 spark.sql('describe extended customers').show(truncate=False)

/usr/local/lib/python3.11/dist-packages/pyspark/sql/session.py in sql(self,␣
 ↪sqlQuery, args, **kwargs)
   1629                     [_to_java_column(lit(v)) for v in (args or [])]
   1630                 )
-> 1631             return DataFrame(self._jsparkSession.sql(sqlQuery, litArgs) ␣
 ↪self)
   1632         finally:
   1633             if len(kwargs) > 0:

/usr/local/lib/python3.11/dist-packages/py4j/java_gateway.py in __call__(self,␣
 ↪*args)
   1320
   1321         answer = self.gateway_client.send_command(command)
-> 1322         return_value = get_return_value(
   1323             answer, self.gateway_client, self.target_id, self.name)
   1324

/usr/local/lib/python3.11/dist-packages/pyspark/errors/exceptions/captured.py i ␣
 ↪deco(*a, **kw)
    183                 # Hide where the exception came from that shows a␣
 ↪non-Pythonic
```

```
    184                 # JVM exception message.
--> 185                 raise converted from None
    186             else:
    187                 raise

AnalysisException: [TABLE_OR_VIEW_NOT_FOUND] The table or view `customers`␣
 ↪cannot be found. Verify the spelling and correctness of the schema and catalo;.
If you did not qualify the name with a schema, verify the current_schema()␣
 ↪output, or qualify the name with the correct schema and catalog.
To tolerate the error on drop use DROP VIEW IF EXISTS or DROP TABLE IF EXISTS.;
 ↪line 1 pos 18;
'DescribeRelation true, [col_name#418, data_type#419, comment#420]
+- 'UnresolvedTableOrView [customers], DESCRIBE TABLE, true
```

[ ]:

[ ]:
```
spark.sql('''
create table if not exists managed_customers (
customer_id int,
name string,
city string,
registration_date date,
is_active boolean
) using csv
''')
```

[ ]: DataFrame[]

[ ]:
```
spark.sql('show tables').show()
```

```
+------------+-----------------+-----------+
|   namespace|        tableName|isTemporary|
+------------+-----------------+-----------+
|customers_db|      customers_2|      false|
|customers_db|managed_customers|      false|
+------------+-----------------+-----------+
```

[ ]:
```
spark.sql('select * from managed_customers').show()
```

```
+-----------+----+----+-----------------+---------+
|customer_id|name|city|registration_date|is_active|
+-----------+----+----+-----------------+---------+
+-----------+----+----+-----------------+---------+
```

[ ]:
```
df - 10gb  2gb
10gb ---> 1000 partitions
200 paritions ->
```

[ ]:
```
df.write.mode('overwrite').saveAsTable("managed_customers")
```

[ ]:
```
spark.sql('select * from managed_customers').show()
```

```
+-----------+-------+-------+-----------------+---------+
|customer_id|   name|   city|registration_date|is_active|
+-----------+-------+-------+-----------------+---------+
|          1|  Alice| Mumbai|       2023-01-15|     true|
|          2|    Bob|  Delhi|       2023-03-25|    false|
|          3|Charlie|Chennai|       2023-05-10|     true|
+-----------+-------+-------+-----------------+---------+
```

[ ]:
```
spark.sql('describe extended managed_customers').show(truncate=False)
```

```
+----------------------------+------------------------------------------------------
------------+-------+
|col_name                    |data_type
|comment|
+----------------------------+------------------------------------------------------
------------+-------+
|customer_id                 |bigint
|NULL   |
|name                        |string
|NULL   |
|city                        |string
|NULL   |
|registration_date           |string
|NULL   |
|is_active                   |boolean
|NULL   |
|                            |
|       |
|# Detailed Table Information|
|       |
|Catalog                     |spark_catalog
|       |
|Database                    |customers_db
|       |
|Table                       |managed_customers
|       |
|Created Time                |Sat Feb 01 04:25:55 UTC 2025
|       |
|Last Access                 |UNKNOWN
```

```
|       |
|Created By                   |Spark 3.5.4
|       |
|Type                         |MANAGED
|       |
|Provider                     |parquet
|       |
|Location                     |file:/content/spark-
warehouse/customers_db.db/managed_customers|       |
+-----------------------------+--------------------------------------------------
------------+-------+
```

[ ]:
```
# prompt: save df as a csv in external table folder

# Assuming 'df' is your DataFrame and you have a folder named␣
↪'external_table_folder' in your desired location.
# Replace 'external_table_folder' with the actual path.

df.repartition(1).write.format("csv").mode("overwrite").save("external_table")
```

[ ]:
```
spark.sql('drop table external_customers')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
spark.sql('show tables').show()
```

```
+------------+------------------+-----------+
|   namespace|         tableName|isTemporary|
+------------+------------------+-----------+
|customers_db|       customers_2|      false|
|customers_db|external_customers|      false|
|customers_db| managed_customers|      false|
+------------+------------------+-----------+
```

[ ]:
```
!ls /content/external_table
```

```
data.csv
```

[ ]:
```
spark.sql('''
create table if not exists external_customers(
  customer_id int,
  name string,
  city string,
  registration_date date,
  is_active boolean
) using csv location '/content/external_table'
''')
```

[ ]: DataFrame[]

[ ]:
```
spark.sql('select * from external_customers').show()
```

```
+-----------+-------+-------+-----------------+---------+
|customer_id|   name|   city|registration_date|is_active|
+-----------+-------+-------+-----------------+---------+
|          1|  Alice| Mumbai|       2023-01-15|     true|
|          2|    Bob|  Delhi|       2023-03-25|    false|
|          3|Charlie|Chennai|       2023-05-10|     true|
+-----------+-------+-------+-----------------+---------+
```

[ ]:
```
# prompt: insert a row in external customer using spark sql insert into

# Insert a new row into the external_customers table
spark.sql("""
INSERT INTO external_customers (customer_id, name, city, is_active)
VALUES (4, 'David', 'Bangalore' , True)
""")

# # Verify the insertion
# spark.sql("SELECT * FROM external_customers").show()
```

[ ]: DataFrame[]

[ ]:
```

```

[ ]:
```
spark.sql('describe extended external_customers').show(truncate = False)
```

```
+----------------------------+------------------------------+-------+
|col_name                    |data_type                     |comment|
+----------------------------+------------------------------+-------+
|customer_id                 |int                           |NULL   |
|name                        |string                        |NULL   |
|city                        |string                        |NULL   |
|registration_date           |date                          |NULL   |
|is_active                   |boolean                       |NULL   |
|                            |                              |       |
|# Detailed Table Information|                              |       |
|Catalog                     |spark_catalog                 |       |
|Database                    |customers_db                  |       |
|Table                       |external_customers            |       |
|Created Time                |Sat Feb 01 04:40:40 UTC 2025  |       |
|Last Access                 |UNKNOWN                       |       |
|Created By                  |Spark 3.5.4                   |       |
|Type                        |EXTERNAL                      |       |
|Provider                    |csv                           |       |
|Location                    |file:///content/external_table|       |
```

```
+--------------------------+----------------------------+-------+
```

[ ]:
```
spark.sql('drop table managed_customers')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
spark.sql('show tables').show()
```

```
+------------+-----------+-----------+
|   namespace|  tableName|isTemporary|
+------------+-----------+-----------+
|customers_db|customers_2|      false|
+------------+-----------+-----------+
```

[ ]:
```
spark.sql('drop table external_customers')
```

[ ]:
```
DataFrame[]
```

[ ]:
```
where is this data stored for Managed table ? in HDFS ?
```