

PySpark

PySpark

Shweta Ajay Shinde

Masters in Data Analytics, San Jose State University

Data 226: Data warehousing

Instructor: Keeyong Han

15th November 2024

PySpark

Introduction to PySpark

1. Install pyspark and download the 7 log files provided in the demo colab code (+1 pt)

Install pyspark:-

```
!pip install pyspark==3.5.3
```

Requirement already satisfied: pyspark==3.5.3 in /usr/local/lib/python3.10/dist-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark==3.5.3) (0.10.9.7)

Download 7 log file:-

```
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_1.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_2.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_3.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_4.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_5.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_6.log.gz
!wget -nc https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_7.log.gz
```

2024-11-12 05:46:59 (91.5 MB/s) - 'sample_web_log_2.log.gz' saved [10277610/10277610]

```
--2024-11-12 05:46:59-- https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_3.log.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10276732 (9.8M) [application/octet-stream]
Saving to: 'sample_web_log_3.log.gz'

sample_web_log_3.lo 100%[=====>] 9.80M --.-KB/s in 0.1s

2024-11-12 05:46:59 (90.6 MB/s) - 'sample_web_log_3.log.gz' saved [10276732/10276732]
```

```
--2024-11-12 05:47:00-- https://raw.githubusercontent.com/keeyong/sjsu-data226/refs/heads/main/week12/data/sample_web_log_4.log.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10277331 (9.8M) [application/octet-stream]
Saving to: 'sample_web_log_4.log.gz'

sample_web_log_4.lo 100%[=====>] 9.80M --.-KB/s in 0.1s

2024-11-12 05:47:00 (95.1 MB/s) - 'sample_web_log_4.log.gz' saved [10277331/10277331]
```

✓ 2s completed at 9:47 PM

PySpark

2. Configure snowflake jar file, set up SparkSession, create an input dataframe (df) and a parsed dataframe (log_df) provided in the demo colab code (+1 pt)

Configure snowflake jar file and set up SparkSession:-

```

✓ 0s  !cd /usr/local/lib/python3.10/dist-packages/pyspark/jars && wget https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.19.0/snowflake-jdbc-3.19.0.jar

--2024-11-12 05:55:35-- https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.19.0/snowflake-jdbc-3.19.0.jar
Resolving repo1.maven.org (repo1.maven.org)... 199.232.192.209, 199.232.196.209, 2a04:4e42:4c::209, ...
Connecting to repo1.maven.org (repo1.maven.org)[199.232.192.209]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 70986770 (68M) [application/java-archive]
Saving to: 'snowflake-jdbc-3.19.0.jar'

snowflake-jdbc-3.19 100%[=====] 67.70M 197MB/s in 0.3s

2024-11-12 05:55:36 (197 MB/s) - 'snowflake-jdbc-3.19.0.jar' saved [70986770/70986770]

```


```

✓ 13s [5] from pyspark.sql import SparkSession
import pyspark.sql.functions as F

spark = SparkSession.builder.appName("HandleLogFiles").getOrCreate()


```

```

✓ 5s  # Load all .gz files in the directory into a DataFrame
df = spark.read.text("*.gz")

```


```

 # Check the number of partitions
print(df.rdd.getNumPartitions())

df.show(truncate=False)

```

```

 3
+-----+
|value|
+-----+
|123.45.67.89 - - [05/Nov/2024:02:08:16 +0000] "DELETE /cart HTTP/1.1" 500 242|
|192.168.1.1 - - [04/Nov/2024:21:23:39 +0000] "POST /checkout HTTP/1.1" 404 2781|
|234.56.78.90 - - [05/Nov/2024:07:06:19 +0000] "GET /api/data HTTP/1.1" 301 3758|
|192.168.1.1 - - [04/Nov/2024:20:03:56 +0000] "POST /home HTTP/1.1" 200 1837|
|192.168.1.1 - - [04/Nov/2024:21:25:05 +0000] "GET /products/123 HTTP/1.1" 200 3430|
|234.56.78.90 - - [04/Nov/2024:07:38:10 +0000] "GET /api/data HTTP/1.1" 404 3729|
|123.45.67.89 - - [04/Nov/2024:12:33:22 +0000] "PUT /api/data HTTP/1.1" 404 799|
|192.168.1.1 - - [04/Nov/2024:07:37:46 +0000] "GET /api/data HTTP/1.1" 500 309|
|123.45.67.89 - - [04/Nov/2024:21:52:36 +0000] "POST /checkout HTTP/1.1" 301 2375|
|123.45.67.89 - - [04/Nov/2024:08:36:44 +0000] "DELETE /api/data HTTP/1.1" 404 3449|
|192.168.1.1 - - [05/Nov/2024:03:15:43 +0000] "GET /api/data HTTP/1.1" 200 2319|
|234.56.78.90 - - [05/Nov/2024:01:26:03 +0000] "DELETE /home HTTP/1.1" 500 1168|
|234.56.78.90 - - [05/Nov/2024:03:26:33 +0000] "DELETE /cart HTTP/1.1" 500 1262|
|123.45.67.89 - - [04/Nov/2024:20:46:25 +0000] "PUT /home HTTP/1.1" 301 4401|
|123.45.67.89 - - [05/Nov/2024:08:07:51 +0000] "GET /api/data HTTP/1.1" 301 3736|
|123.45.67.89 - - [04/Nov/2024:21:01:30 +0000] "DELETE /cart HTTP/1.1" 404 2418|
|123.45.67.89 - - [04/Nov/2024:09:40:29 +0000] "POST /api/data HTTP/1.1" 301 3260|
|234.56.78.90 - - [04/Nov/2024:09:23:42 +0000] "GET /home HTTP/1.1" 200 1488|
|192.168.1.1 - - [04/Nov/2024:11:53:57 +0000] "POST /products/123 HTTP/1.1" 200 2627|
|234.56.78.90 - - [05/Nov/2024:01:26:01 +0000] "PUT /cart HTTP/1.1" 500 4406|
+-----+
only showing top 20 rows

```

PySpark

Extract the information in table format:-

```
[7] # Extract the necessary information from log data using regular expressions
pattern = r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?) (.*?) HTTP.*" (\d+) (\d+)'

log_df = df.select(
    F.regexp_extract("value", pattern, 1).alias("ip"),
    F.regexp_extract("value", pattern, 2).alias("timestamp"),
    F.regexp_extract("value", pattern, 3).alias("method"),
    F.regexp_extract("value", pattern, 4).alias("url"),
    F.regexp_extract("value", pattern, 5).alias("status").cast("integer"),
    F.regexp_extract("value", pattern, 6).alias("size").cast("integer")
)
```

```
log_df.show()
```

ip	timestamp	method	url	status	size
123.45.67.89	05/Nov/2024:02:08...	DELETE	/cart	500	242
192.168.1.1	04/Nov/2024:21:23...	POST	/checkout	404	2781
234.56.78.90	05/Nov/2024:07:06...	GET	/api/data	301	3758
192.168.1.1	04/Nov/2024:20:03...	POST	/home	200	1837
192.168.1.1	04/Nov/2024:21:25...	GET	/products/123	200	3430
234.56.78.90	04/Nov/2024:07:38...	GET	/api/data	404	3729
123.45.67.89	04/Nov/2024:12:33...	PUT	/api/data	404	799
192.168.1.1	04/Nov/2024:07:37...	GET	/api/data	500	309
123.45.67.89	04/Nov/2024:21:52...	POST	/checkout	301	2375
123.45.67.89	04/Nov/2024:08:36...	DELETE	/api/data	404	3449
192.168.1.1	05/Nov/2024:03:15...	GET	/api/data	200	2319
234.56.78.90	05/Nov/2024:01:26...	DELETE	/home	500	1168
234.56.78.90	05/Nov/2024:03:26...	DELETE	/cart	500	1262
123.45.67.89	04/Nov/2024:20:46...	PUT	/home	301	4401
123.45.67.89	05/Nov/2024:08:07...	GET	/api/data	301	3736
123.45.67.89	04/Nov/2024:21:01...	DELETE	/cart	404	2418
123.45.67.89	04/Nov/2024:09:40...	POST	/api/data	301	3260
234.56.78.90	04/Nov/2024:09:23...	GET	/home	200	1488
192.168.1.1	04/Nov/2024:11:53...	POST	/products/123	200	2627
234.56.78.90	05/Nov/2024:01:26...	PUT	/cart	500	4406

only showing top 20 rows

PySpark

3. Compute the counts of url and status combination. Use DataFrame operations to compute the count of each unique url and status combo (+2 pt).

```

✓ 38s #compute count of each unique URL and Status combination
count_df=log_df.groupby("url","status").count()
count_df.show()

```

url	status	count
/products/123	500	349853
/products/123	200	349536
/checkout	500	349414
/products/123	301	349904
/checkout	200	351036
/home	500	351457
/cart	500	349433
/home	404	349492
/checkout	404	349604
/cart	301	350262
/api/data	200	349703
/api/data	301	348954
/api/data	500	350091
/cart	200	350305
/home	200	350541
/products/123	404	350970
/home	301	350203
/checkout	301	349914
/cart	404	349830
/api/data	404	349498

PySpark

4. Compute the same (step 3) with SparkSQL (+2 pt)

```

36s  # Register the DataFrame as a temporary SQL table
log_df.createOrReplaceTempView("logs_df_status")
# Use SparkSQL to count URLs with 404 status
count_df_spark = spark.sql("""
    SELECT url, status, COUNT(*) as count_status
    FROM logs_df_status
    GROUP BY url, status
    ORDER BY count_status DESC
""")
count_df_spark.show()

```

```

 +-----+-----+-----+
|          url|status|count_status|
+-----+-----+-----+
|      /home|    500|      351457|
|  /checkout|    200|      351036|
|/products/123|   404|      350970|
|      /home|    200|      350541|
|      /cart|    200|      350305|
|      /cart|   301|      350262|
|      /home|   301|      350203|
|  /api/data|   500|      350091|
|  /checkout|   301|      349914|
|/products/123|   301|      349904|
|/products/123|   500|      349853|
|      /cart|   404|      349830|
|  /api/data|   200|      349703|
|  /checkout|   404|      349604|
|/products/123|   200|      349536|
|  /api/data|   404|      349498|
|      /home|   404|      349492|
|      /cart|   500|      349433|
|  /checkout|   500|      349414|
|  /api/data|   301|      348954|
+-----+-----+-----+

```

5. Load this dataframe from step 4 into a table in Snowflake (+2 pt)

```

42s  [28] from google.colab import userdata

account = userdata.get('account')
user = userdata.get('User')
password = userdata.get('snowflake_password')
database = "dev"
schema = "analytics"

url = f"jdbc:snowflake://{account}.snowflakecomputing.com/?db={database}&schema={schema}&user={user}&password={password}"

count_df_spark.write \
    .format("jdbc") \
    .option("driver", "net.snowflake.client.jdbc.SnowflakeDriver") \
    .option("url", f"jdbc:snowflake://{account}.snowflakecomputing.com") \
    .option("dbtable", "count_df_spark") \
    .option("user", user) \
    .option("password", password) \
    .option("database", database) \
    .option("schema", schema) \
    .mode("overwrite") \
    .save()

```

PySpark

6. Capture the screenshot of the table (SELECT results) in Snowflake (+1 pt)

The screenshot shows the Snowflake web interface. On the left, the 'Databases' tab is active, showing a tree view of the database structure. The 'DEV' database is expanded, showing the 'ANALYTICS' schema, which contains a table named 'COUNT_DF_SPARK'. Below this, the 'COUNT_DF_SPARK' table is listed with 20 rows. The table schema is: url (VARCHAR(16777216)), status (NUMBER(38,0)), and count_status (NUMBER(38,0)).

In the center, the SQL query editor shows the following query:

```
select * from DEV.ANALYTICS.COUNT_DF_SPARK;
```

Below the query editor, the 'Results' tab is active, displaying the query results in a table format. The table has three columns: url, status, and count_status. The results are as follows:

	url	status	count_status
1	/home	500	351457
2	/api/data	301	348954
3	/checkout	500	349414
4	/cart	500	349433
5	/home	404	349492
6	/api/data	404	349498
7	/products/123	200	349536
8	/checkout	404	349604
9	/api/data	200	349703
10	/cart	404	349830
11	/products/123	500	349853

On the right side of the interface, the 'Query Details' panel is visible, showing the query duration as 351ms, the number of rows as 20, and the query ID as 01b85fd0-0004-2cf4-0-... There is also an 'Ask Copilot' button at the bottom right of the panel.

Git Link:

https://github.com/ShindeShwetaK/DW_Assignment10/blob/main/Shweta_Shinde_Homework10.ipynb