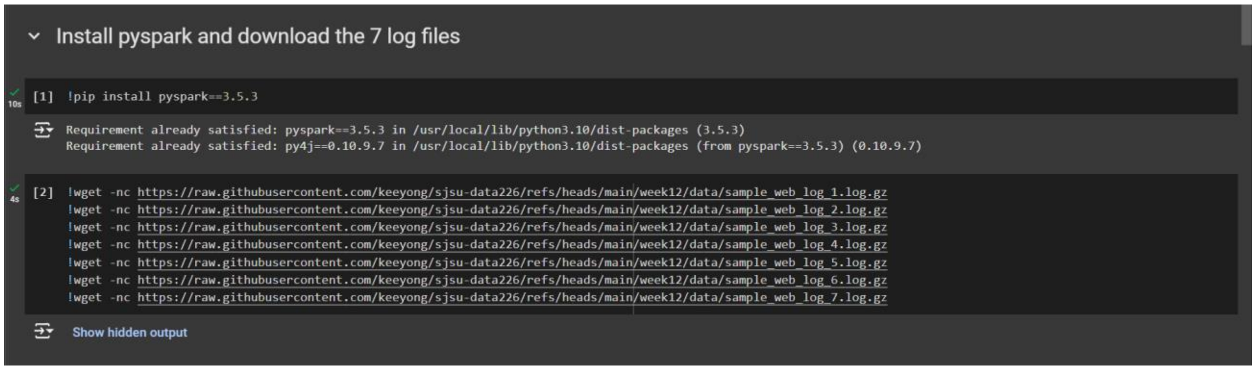
Assignment-10

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



Install pyspark and download the 7 log files

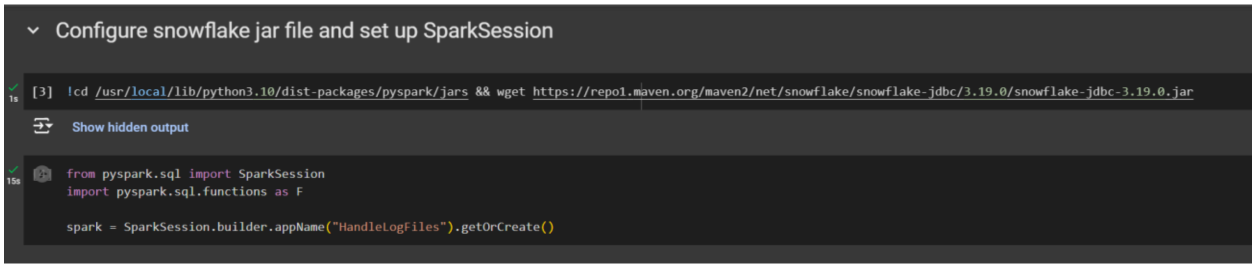
```
[1] !pip install pyspark==3.5.3

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)
```

```
[2] !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
```

Show hidden output

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)



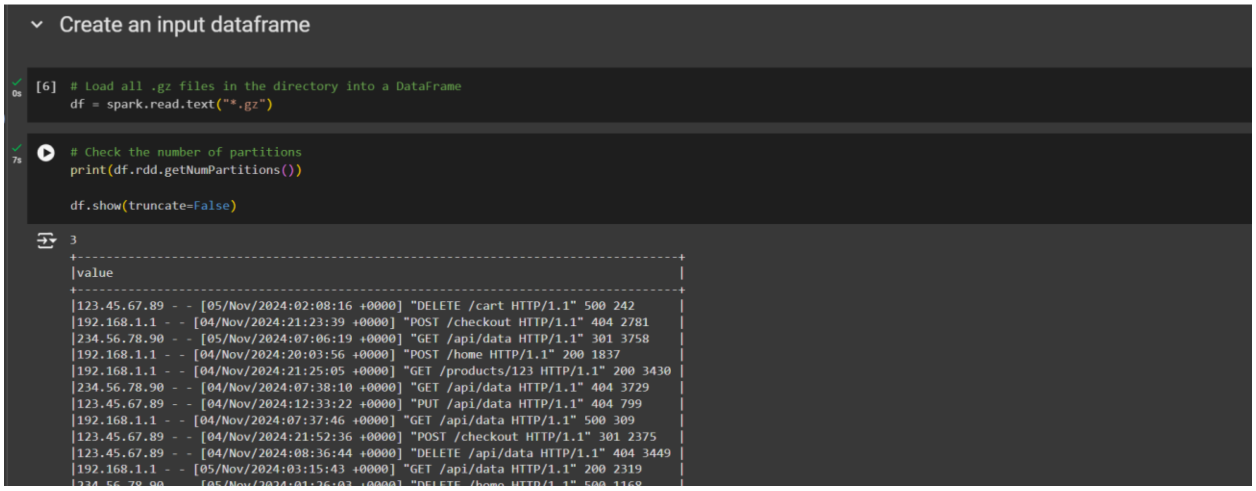
Configure snowflake jar file and set up SparkSession

```
[3] !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
```

Show hidden output

```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F

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



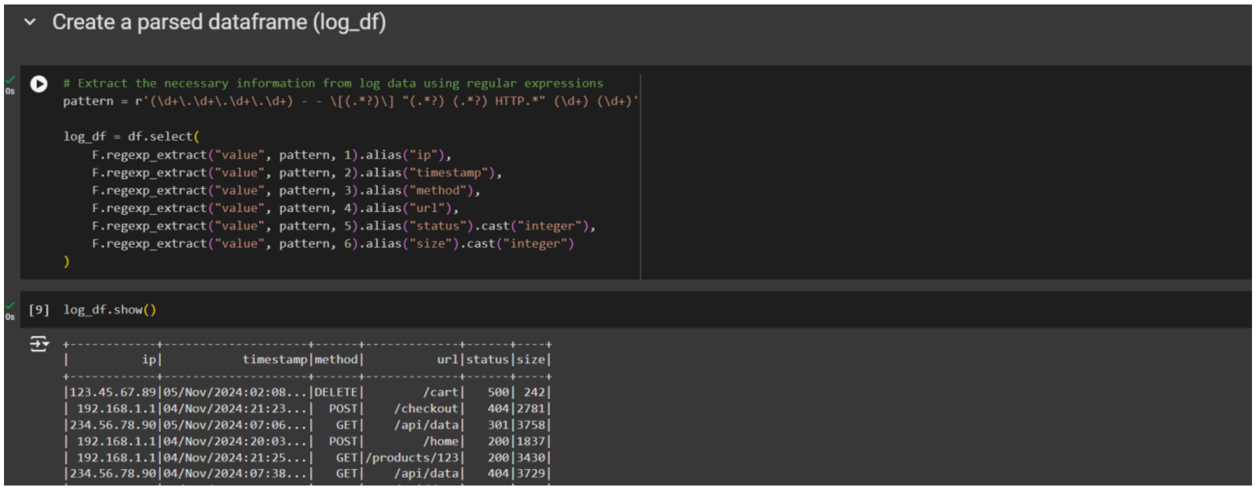
Create an input dataframe

```
[6] # 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  |
```



Create a parsed dataframe (log_df)

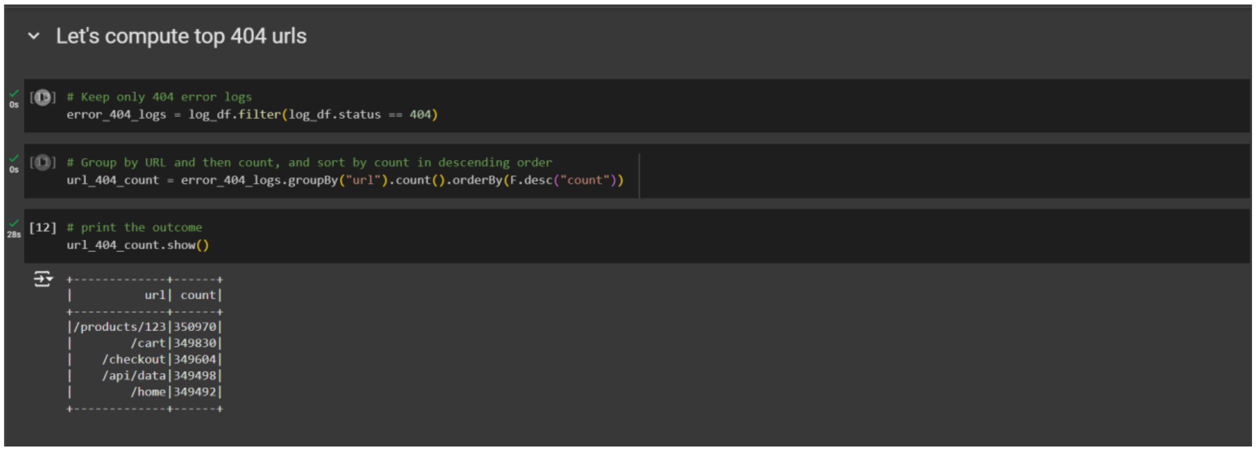
```
# Extract the necessary information from log data using regular expressions
pattern = r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?) (.*?) HTTP.*" (\d+) (\d+)'

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")
)
```

```
[9] 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|
```

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



Let's compute top 404 urls

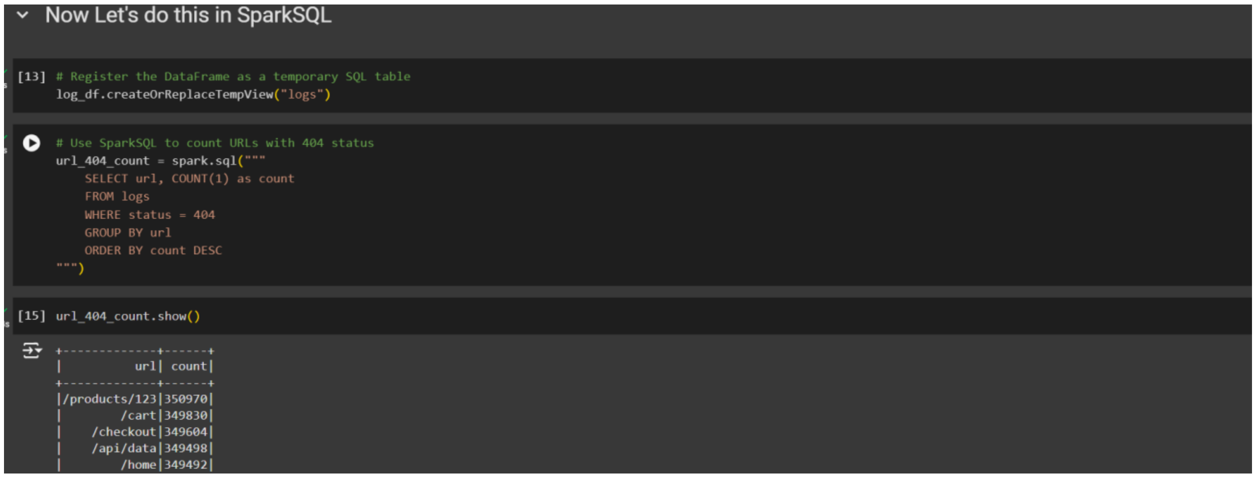
```
# Keep only 404 error logs
error_404_logs = log_df.filter(log_df.status == 404)
```

```
# Group by URL and then count, and sort by count in descending order
url_404_count = error_404_logs.groupBy("url").count().orderBy(F.desc("count"))
```

```
[12] # print the outcome
url_404_count.show()
```

```
+-------------+------+
|          url| count|
+-------------+------+
|/products/123|350970|
|        /cart|349830|
|    /checkout|349604|
|    /api/data|349498|
|        /home|349492|
+-------------+------+
```

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



Now Let's do this in SparkSQL

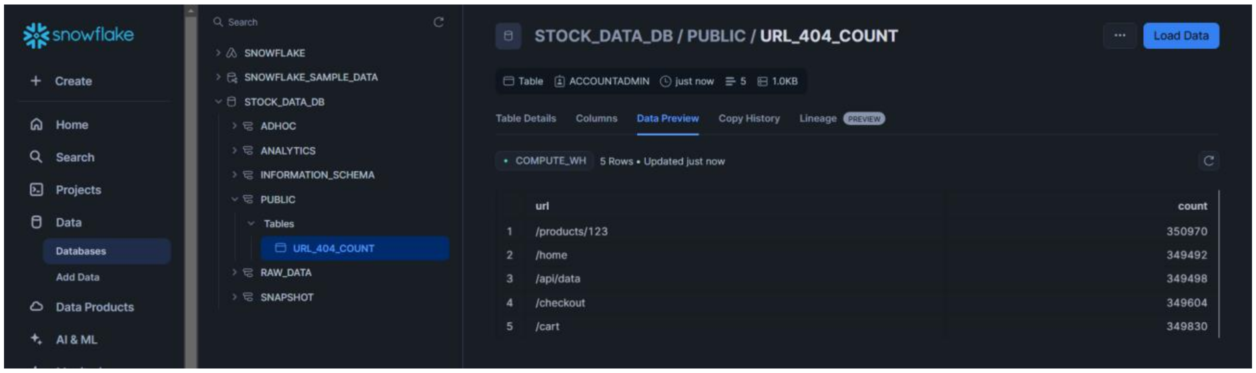
```
[13] # Register the DataFrame as a temporary SQL table
log_df.createOrReplaceTempView("logs")
```

```
# Use SparkSQL to count URLs with 404 status
url_404_count = spark.sql("""
    SELECT url, COUNT(1) as count
    FROM logs
    WHERE status = 404
    GROUP BY url
    ORDER BY count DESC
""")
```

```
[15] url_404_count.show()
```

```
+-------------+------+
|          url| count|
+-------------+------+
|/products/123|350970|
|        /cart|349830|
|    /checkout|349604|
|    /api/data|349498|
|        /home|349492|
```

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

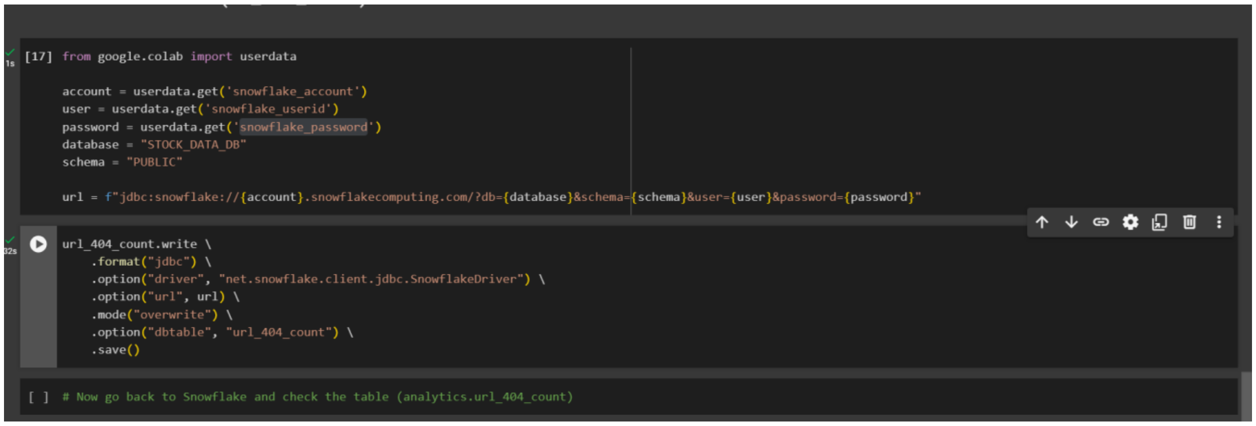


STOCK_DATA_DB / PUBLIC / URL_404_COUNT

Table ACCOUNTADMIN just now 5 1.0KB

COMPUTE_WH 5 Rows • Updated just now

	url	count
1	/products/123	350970
2	/home	349492
3	/api/data	349498
4	/checkout	349604
5	/cart	349830



```
[17] from google.colab import userdata

     account = userdata.get('snowflake_account')
     user = userdata.get('snowflake_userid')
     password = userdata.get('snowflake_password')
     database = "STOCK_DATA_DB"
     schema = "PUBLIC"

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

```
url_404_count.write \
    .format("jdbc") \
    .option("driver", "net.snowflake.client.jdbc.SnowflakeDriver") \
    .option("url", url) \
    .mode("overwrite") \
    .option("dbtable", "url_404_count") \
    .save()
```

```
[ ] # Now go back to Snowflake and check the table (analytics.url_404_count)
```

https://colab.research.google.com/drive/1rYIHzsvERLNVe3SUv9fbv1_bHK5d6aL9?usp=sharing