There are multiple ways to create notebook, I Initially launched an EMR cluster.

A screenshot of a computer

Description automatically generated

In the next step we need to create an studio and then create a workspace (Notebooks).

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Launching jupyter from the notebook.

A screenshot of a computer

Description automatically generated

Pyspark practice in jupyter notebook.

A screenshot of a computer

Description automatically generated

Running pyspark commands on putty :

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Join examples:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Reading data from csv s3 ..

A screenshot of a computer

Description automatically generated

Reading a csv file from s3 bucket and displaying data in it and performing few operations.

A screenshot of a computer

Description automatically generated

Showing the male candidates ..

A screenshot of a computer

Description automatically generated

Female candidates...

A screenshot of a computer

Description automatically generated

Performing union operation , combining 2 csv files.

A screenshot of a computer

Description automatically generated

Checking for duplicates if any after combining the 2 files.

A computer screen shot of a black screen

Description automatically generated

Write back to the s3 bucket----

A computer screen shot of a black background

Description automatically generated

A screenshot of a computer

Description automatically generated

Reading a txt file from s3 ---

A black background with text

Description automatically generated

Searching for a particular word from the txt file and checking how many times it has repeated.

A computer screen with white text

Description automatically generated

Reading a Json file from s3 ..

A screenshot of a computer

Description automatically generated

In the below example we are reading a car sales report and performing some analysis ..

A screenshot of a computer screen

Description automatically generated

Using FILTER extracted only the BMW from the Brand column.

A screenshot of a computer program

Description automatically generated

Using group by command we displayed all the 7 brands and their average price of their cars.

A screenshot of a computer

Description automatically generated

Using COUNT checking how many cars were sold by each brand.

A screenshot of a computer program

Description automatically generated

Adding a new column as updated date

A screen shot of a computer

Description automatically generated

Writing back to the S3 bucket.

A computer screen with white text

Description automatically generatedA screenshot of a computer

Description automatically generated

Count distinct is used to count the no of car companies in the brand column.

A screenshot of a computer

Description automatically generated

Broadcast join sample example: It is used when 2 data frames or RDD’s are joining where one of it is a large data set than the other, here instead of distributing the smaller DataFrame across the cluster, Spark broadcasts it to all the worker nodes. This reduces data shuffling and network overhead, as the larger DataFrame can be joined with the smaller one locally on each worker node.

A screen shot of a computer

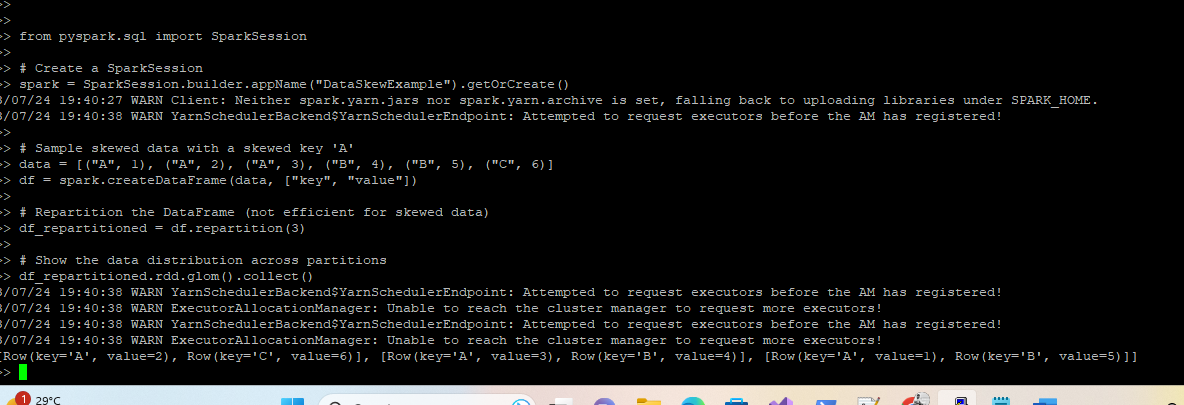
Description automatically generated

Accumulators are variables in Spark used for aggregating information across worker nodes. They allow you to perform calculations and update their values in a distributed manner.

A screenshot of a computer program

Description automatically generated

Data skew : Data skew refers to an imbalanced distribution of data across partitions in a distributed system. In the context of Apache Spark, data skew can negatively impact performance, as some worker nodes may be overloaded while others are underutilized. This leads to longer processing times and inefficient resource utilization.



Reading 2 csv files from s3 and performing Broadcast join:

A screenshot of a computer program

Description automatically generated

Here we got an error message as there are extra column in the second files .

Using print schema and verifying the column names (joined\_df = df1.join(broadcast(df2), df1["Name"] == df2["Name"]) )

A screenshot of a computer program

Description automatically generated

Accumulator example by reading files from S3

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Data Skew using S3:

Using groupby:

print("Data distribution in df1:")

df1.groupBy("column\_name\_to\_check\_for\_skew").count().show()

print("Data distribution in df2:")

df2.groupBy("column\_name\_to\_check\_for\_skew").count().show()

A screenshot of a computer screen

Description automatically generated

df.cache(): the DataFrame **df** is cached in memory using **df.cache()** after reading from S3. Subsequent operations like filtering and grouping are performed on the DataFrame, and when you call actions like **show()**, Spark will use the cached data to reduce recomputation time.

A screenshot of a computer program

Description automatically generated

df.persist() : the DataFrame **df** is persisted to disk using **df.persist()**. By default, it uses the **MEMORY\_AND\_DISK** storage level. Subsequent operations on the DataFrame access the persisted data from disk when calling actions like **show()**, reducing recomputation time.

A screenshot of a computer screen

Description automatically generated

Caching in memory is faster, but it consumes more memory, so it's essential to balance your memory requirements and the size of the DataFrame you are working with. Additionally, make sure to call **unpersist()** when you are done with the cached or persisted DataFrame to release the memory or disk space.

