Transformation

Explore the transformations and write down all the transformations you learned.

Dropping Nulls:

First check for mount point, if we don't have one need to create one.

1 minute ago (3s) 3 Python

```
dbutils.fs.mounts()
```

```
[MountInfo(mountPoint='/databricks-datasets', source='databricks-datasets', encryptionType=''),
 MountInfo(mountPoint='/Volumes', source='UnityCatalogVolumes', encryptionType=''),
 MountInfo(mountPoint='/databricks/mlflow-tracking', source='databricks/mlflow-tracking', encryptionType=''),
 MountInfo(mountPoint='/databricks-results', source='databricks-results', encryptionType=''),
 MountInfo(mountPoint='/databricks/mlflow-registry', source='databricks/mlflow-registry', encryptionType=''),
 MountInfo(mountPoint='/Volume', source='DbfsReserved', encryptionType=''),
 MountInfo(mountPoint='/volumes', source='DbfsReserved', encryptionType=''),
 MountInfo(mountPoint='/mnt/containers1', source='wasbs://container1@adlsdeepthik.blob.core.windows.net', encryptionType
=''),
 MountInfo(mountPoint='/', source='DatabricksRoot', encryptionType=''),
 MountInfo(mountPoint='/mnt/container1', source='wasbs://container1@adlsdeepthik.blob.core.windows.net', encryptionType=''),
 MountInfo(mountPoint='/volume', source='DbfsReserved', encryptionType='')]
```

Just now (1s) 4

```
dbutils.fs.ls('/mnt/containers1')
```

```
[FileInfo(path='dbfs:/mnt/containers1/CSV_Data/', name='CSV_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/containers1/Delta_Data/', name='Delta_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/containers1/Parquet_Data/', name='Parquet_Data/', size=0, modificationTime=0)]
```

Once we check the mount point create a data frame and assign the file data to data frame

Just now (8s) 5 Python

```
df=spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/mnt/containers1/CSV_Data/
Credit.csv")
display(df)
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789
11	5	null	Master	678
12	4	Shriya	Visa	null

We see we have null values in this data frame, to remove null values we have a function called na.drop()

df_null_drop = df.na.drop()

display(df_null_drop)

Just now (1s) 6 Python

```
df_null_drop = df.na.drop()
display(df_null_drop)
```

(1) Spark Jobs

df_null_drop: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789

10 rows | 0.71s runtime Refreshed now

Null values rows are deleted, this function will remove complete row which has Null values.

To check which rows are NULL we need to use below code

from pyspark.sql.functions import col
df_null_rows = df.where(col("Credit_Score").isNull())
display(df_null_rows)

Just now (<1s) 7 Python

```
from pyspark.sql.functions import col
df_null_rows = df.where(col("Credit_Score").isNull())
display(df_null_rows)
```

(1) Spark Jobs

df_null_rows: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_Type	Credit_Score
1	4	Shriya	Visa	null

This code return Rows which have Null values in that particular column which we mentioned.

from pyspark.sql.functions import col
df_null_rows = df.where(col("Credit_Score").isNotNull())
display(df_null_rows)
This code will display the rows which doesn’t have any null values in Credit_Score column

Just now (1s) 8 Python

```python
from pyspark.sql.functions import col
df_null_rows = df.where(col("Credit_Score").isNotNull())
display(df_null_rows)
```

(1) Spark Jobs

df_null_rows: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_...	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789
11	5	null	Master	678

Replace NULL values with some data

df_fill_Null = df.na.fill("Unknown")

display(df_fill_Null)

This code will fill Unknown in all NULL fields of String datatype columns

Just now (1s) 10 Python

```python
df_fill_Null = df.na.fill("Unknown")
display(df_fill_Null)
```

(1) Spark Jobs

df_fill_Null: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789
11	5	Unknown	Master	678
12	4	Shriya	Visa	null

Below is for int type of columns

Just now (<1s) 11 Python

```python
df_fill_Null_Int = df.na.fill(0)
display(df_fill_Null_Int)
```

(1) Spark Jobs

df_fill_Null_Int: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_Name	Credit_...	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789
11	5	null	Master	678
12	4	Shriya	Visa	0

df_fill_Custom = df.na.fill({"Credit_Score": 0, "Credit_Name": "Unknown"})

display(df_fill_Custom)

This code will fill those value sin NULL fields based on the column names

Just now (<1s) 12 Python

```python
df_fill_Custom = df.na.fill({"Credit_Score": 0, "Credit_Name": "Unknown"})
display(df_fill_Custom)
```

(1) Spark Jobs

df_fill_Custom: pyspark.sql.dataframe.DataFrame = [Credit_id: integer, Credit_Name: string ... 2 more fields]

Table

	Credit_id	Credit_...	Credit_Type	Credit_Score
1	1	Deepthi	Master	989
2	2	Rahul	Visa	1124
3	3	Anjana	Visa	1124
4	4	Shriya	Master	765
5	1	Deepthi	Visa	678
6	2	Rahul	Master	1245
7	3	Anjana	Master	989
8	4	Shriya	Visa	765
9	1	Deepthi	Master	989
10	3	Anjana	Master	789
11	5	Unknown	Master	678
12	4	Shriya	Visa	0

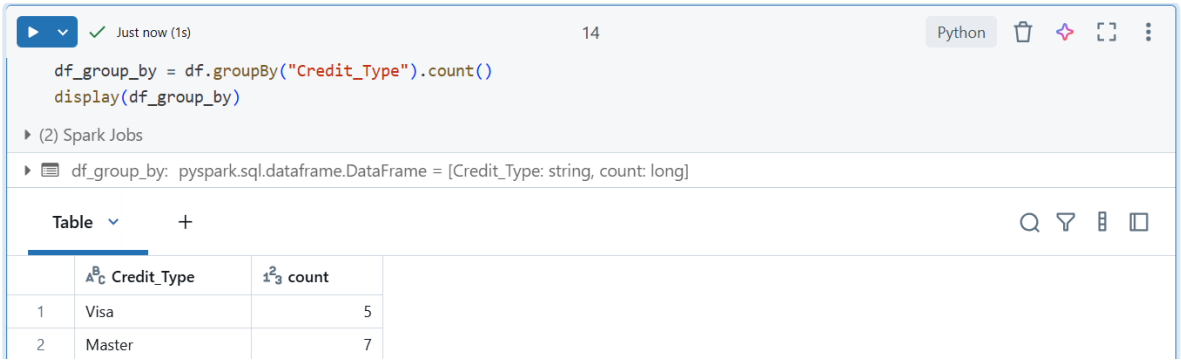
Aggregate Functions in PySpark

Aggregate functions perform calculations on a **group of rows** and return a **single value**. They are commonly used with groupBy() or agg().

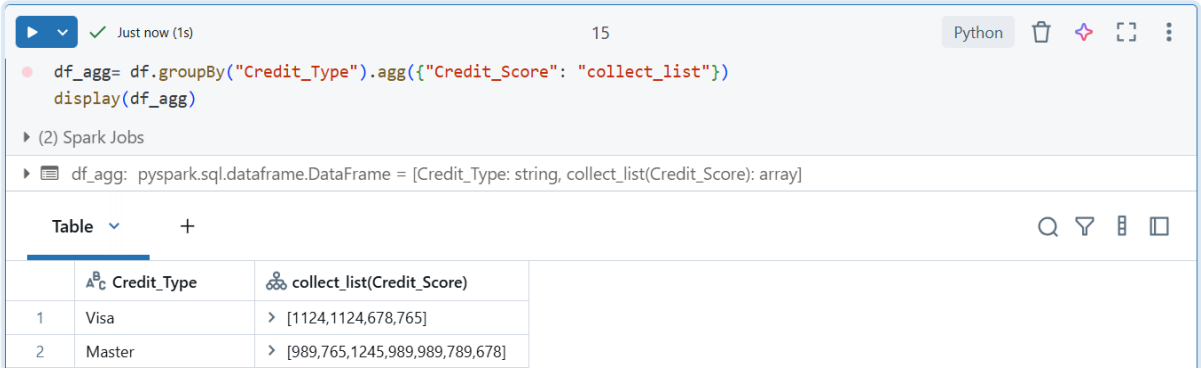
Common Aggregate Functions in PySpark

Function	Description	Example
sum()	Sum of values	df.groupBy("category").agg(sum("sales"))
avg()	Average value	df.groupBy("category").agg(avg("sales"))
count()	Count of rows	df.groupBy("category").agg(count("*"))
min()	Minimum value	df.groupBy("category").agg(min("sales"))
max()	Maximum value	df.groupBy("category").agg(max("sales"))

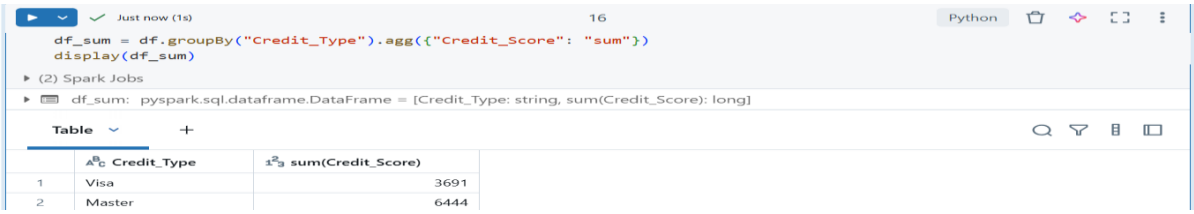
groupBy():
df_group_by = df.groupBy("Credit_Type").count()
display(df_group_by)



df_agg= df.groupBy("Credit_Type").agg({"Credit_Score": "collect_list"})
display(df_agg)



df_sum = df.groupBy("Credit_Type").agg({"Credit_Score": "sum"})
display(df_sum)



from pyspark.sql.functions import col,collect_list,collect_set
df_agg_set = df.groupBy("Credit_Type").agg(collect_list("Credit_Score"))
display(df_agg_set)

Just now (1s) 17 Python

```
from pyspark.sql.functions import col,collect_list,collect_set
df_agg_set = df.groupBy("Credit_Type").agg(collect_list("Credit_Score"))
display(df_agg_set)
```

(2) Spark Jobs

df_agg_set: pyspark.sql.dataframe.DataFrame = [Credit_Type: string, collect_list(Credit_Score): array]

Table

	Credit_Type	collect_list(Credit_Score)
1	Visa	[1124,1124,678,765]
2	Master	[989,765,1245,989,989,789,678]

Collect_set will remove duplicate values and will give the results with unique values

df_agg_set1 = df.groupBy("Credit_Type").agg(collect_set("Credit_Score"))
display(df_agg_set1)

Just now (1s) 18 Python

```
df_agg_set1 = df.groupBy("Credit_Type").agg(collect_set("Credit_Score"))
display(df_agg_set1)
```

(3) Spark Jobs

df_agg_set1: pyspark.sql.dataframe.DataFrame = [Credit_Type: string, collect_set(Credit_Score): array]

Table

	Credit_Type	collect_set(Credit_Score)
1	Visa	[765,678,1124]
2	Master	[765,789,989,678,1245]

df_max = df.groupBy("Credit_Type").agg({"Credit_Score": "max"})
display(df_max)

Just now (<1s) 19 Python

```
df_max = df.groupBy("Credit_Type").agg({"Credit_Score": "max"})
display(df_max)
```

(2) Spark Jobs

df_max: pyspark.sql.dataframe.DataFrame = [Credit_Type: string, max(Credit_Score): integer]

Table

	Credit_Type	max(Credit_Score)
1	Visa	1124
2	Master	1245