Filter Operations &, |, ==, ~

Group by and Aggregate Functions

In [2]:
```
pip --proxy http://[username]:[password]@noidaproxy.corp.exlservice.com:8000 i
nstall pyspark
```

```
Collecting pyspark
  Downloading https://files.pythonhosted.org/packages/b8/01/b2393cee7f6180d91
50274e92c8bdc1c81220e2ad7554ee5febca1866899/pyspark-3.3.0.tar.gz (281.3MB)
Collecting py4j==0.10.9.5 (from pyspark)
  Using cached https://files.pythonhosted.org/packages/86/ec/60880978512d5569
ca4bf32b3b4d7776a528ecf4bca4523936c98c92a3c8/py4j-0.10.9.5-py2.py3-none-any.w
hl
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py): started
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): finished with status 'done'
  Stored in directory: C:\Users\shrinath195156\AppData\Local\pip\Cache\wheels
\9e\c1\93\d40ec851fc2b278e1056c1353ff95a7a4ef1b219f74ca9c11f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.0
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pyspark
```

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Practice").getOrCreate()
```

In [3]:
```
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://EXLAPLPNyCdxfzp.corp.exlservice.com:4040)

Version

v3.3.0

Master

local[*]

AppName

Practice

In [4]:
```
#reading dataset using spark session
df_pyspark=spark.read.csv("sparktest_1.csv", header=True, inferSchema=True)
df_pyspark.show()
```

```
+---------+--------+----+----------+------+
|     name|    dept| age|experience|salary|
+---------+--------+----+----------+------+
|    Krish|   Sales|  31|        10| 30000|
|Sudhanshu| Finance|  30|         8| 25000|
|    Sunny|      IT|  29|         4| 20000|
|     Paul|Products|  24|         3| 20000|
|   Harsha|   Sales|  21|         1| 15000|
|  Shubham|      IT|  23|         2| 18000|
|   Mahesh|Products|null|      null| 40000|
|     null| Finance|  34|        10| 38000|
|     null| Finance|  36|      null|  null|
+---------+--------+----+----------+------+
```

In [5]:
```
#Filtering salary of the people < or = 23000
df_pyspark.filter("salary<=23000").show()
```

```
+-------+--------+---+----------+------+
|   name|    dept|age|experience|salary|
+-------+--------+---+----------+------+
|  Sunny|      IT| 29|         4| 20000|
|   Paul|Products| 24|         3| 20000|
| Harsha|   Sales| 21|         1| 15000|
|Shubham|      IT| 23|         2| 18000|
+-------+--------+---+----------+------+
```

In [7]:
```
#Filtering salary of the people < or = 23000 and selecting name, age and dept
df_pyspark.filter("salary<=23000").select(["name", "age", "dept","salary"]).sh
ow()
```

```
+-------+---+--------+------+
|   name|age|    dept|salary|
+-------+---+--------+------+
|  Sunny| 29|      IT| 20000|
|   Paul| 24|Products| 20000|
| Harsha| 21|   Sales| 15000|
|Shubham| 23|      IT| 18000|
+-------+---+--------+------+
```

In [9]:
```
#Filtering salary of the people < or = 23000 and selecting name, age and dept
 using diff syntax
df_pyspark.filter(df_pyspark["salary"]<=23000).select(["name", "age", "dept",
"salary"]).show()
```

```
+-------+---+--------+------+
|   name|age|    dept|salary|
+-------+---+--------+------+
|  Sunny| 29|      IT| 20000|
|   Paul| 24|Products| 20000|
| Harsha| 21|   Sales| 15000|
|Shubham| 23|      IT| 18000|
+-------+---+--------+------+
```

In [13]:
```
#Filtering data using multiple conditions for salary < or = 20000 & >=30000 an
d selecting name, age and dept using diff syntax
df_pyspark.filter((df_pyspark["salary"]<=30000) &
                  (df_pyspark["salary"]>=20000)).select(["name", "age", "dept"
,"salary"]).show()
```

```
+---------+---+--------+------+
|     name|age|    dept|salary|
+---------+---+--------+------+
|    Krish| 31|   Sales| 30000|
|Sudhanshu| 30| Finance| 25000|
|    Sunny| 29|      IT| 20000|
|     Paul| 24|Products| 20000|
+---------+---+--------+------+
```

In [14]:
```
#Filtering data using NOT (inverse) ~ in between the range of salary < or = 20
000 & >=30000
#and selecting name, age and dept using diff syntax
df_pyspark.filter(~(df_pyspark["salary"]<=30000) &
                  (df_pyspark["salary"]>=20000)).select(["name", "age", "dept"
,"salary"]).show()
```

```
+------+----+--------+------+
|  name| age|    dept|salary|
+------+----+--------+------+
|Mahesh|null|Products| 40000|
|  null|  34| Finance| 38000|
+------+----+--------+------+
```

Group by and Aggregate Functions

In [41]:
```
#reading dataset 2
df_pyspark1=spark.read.csv("sparktest_2.csv", header=True, inferSchema=True)
df_pyspark1.show()
```

```
+---------+--------+------+
|     name|    dept|salary|
+---------+--------+------+
|    Krish|   Sales| 30000|
|Sudhanshu| Finance| 25000|
|    Sunny|      IT| 20000|
|     Paul|Products| 20000|
|   Harsha|   Sales| 15000|
|  Shubham|      IT| 18000|
|   Mahesh|Products| 40000|
|     Paul| Finance| 18000|
|   Mahesh| Finance|  null|
|    Krish| Finance| 20000|
|Sudhanshu|   Sales| 30000|
|  Shubham|     Ops|  null|
|    Sunny|     Ops| 15000|
|   Harsha|Products| 30000|
+---------+--------+------+
```

In [42]:
```
#Like a data type reviewing schema
df_pyspark1.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- dept: string (nullable = true)
 |-- salary: integer (nullable = true)
```

In [43]:
```
#Using Group by for names to find total salary
df_pyspark1.groupBy("name").sum().show()
```

```
+---------+-----------+
|     name|sum(salary)|
+---------+-----------+
|Sudhanshu|      55000|
|    Sunny|      35000|
|    Krish|      50000|
|   Harsha|      45000|
|     Paul|      38000|
|  Shubham|      18000|
|   Mahesh|      40000|
+---------+-----------+
```

In [44]:
```
#Using Group by dept to find total salary
df_pyspark1.groupBy("dept").sum().show()
```

```
+--------+-----------+
|    dept|sum(salary)|
+--------+-----------+
|   Sales|      75000|
| Finance|      63000|
|     Ops|      15000|
|      IT|      38000|
|Products|      90000|
+--------+-----------+
```

In [45]:
```
#Using Group by dept to find mean salary
df_pyspark1.groupBy("dept").mean().show()
```

```
+--------+-----------+
|    dept|avg(salary)|
+--------+-----------+
|   Sales|    25000.0|
| Finance|    21000.0|
|     Ops|    15000.0|
|      IT|    19000.0|
|Products|    30000.0|
+--------+-----------+
```

In [46]:
```
#Using Group by dept to find count of employee
df_pyspark1.groupBy("dept").count().show()
```

```
+--------+-----+
|    dept|count|
+--------+-----+
|   Sales|    3|
| Finance|    4|
|     Ops|    2|
|      IT|    2|
|Products|    3|
+--------+-----+
```

In [48]:
```
#Using Aggregate sum of Salary to find entire total
df_pyspark1.agg({"salary":"sum"}).show()
```

```
+-----------+
|sum(salary)|
+-----------+
|     281000|
+-----------+
```

In [64]:
```
df_pyspark1.groupBy("name").agg({"salary":"max"}).show()
```

```
+---------+-----------+
|     name|max(salary)|
+---------+-----------+
|Sudhanshu|      30000|
|    Sunny|      20000|
|    Krish|      30000|
|   Harsha|      30000|
|     Paul|      20000|
|  Shubham|      18000|
|   Mahesh|      40000|
+---------+-----------+
```

In [59]:
```
#Using Group by for names to find maximum salary
df_pyspark1.groupBy("name").max().show()
```

```
+---------+-----------+
|     name|max(salary)|
+---------+-----------+
|Sudhanshu|      30000|
|    Sunny|      20000|
|    Krish|      30000|
|   Harsha|      30000|
|     Paul|      20000|
|  Shubham|      18000|
|   Mahesh|      40000|
+---------+-----------+
```