

(<https://databricks.com>)

Access key and Secret Access key

```
ls /FileStore/shared_uploads/yashbuty07@gmail.com/30cdbiotaws00124_accessKeys__1_-2.csv
```

```
aws_keys=spark.read.format('csv').option('header','true').option('inferschema','true').load('/FileStore/shared_uploads/yashbuty07@gmail.com/30cdbiotaws00124_accessKeys__1_-1.csv')
aws_keys.columns
```

```
ak=aws_keys.select('Access key ID').take(1)[0]['Access key ID']
sk=aws_keys.select('Secret access key').take(1)[0]['Secret access key']
```

Encryption Code

```
import urllib
encoded=urllib.parse.quote(string=sk,safe="")
```

Mounting S3 Buckets on Databricks environment

```
#bucket1
aws_bucket1='aws-bucky7'
mount_name1='/mnt/bucket1'
```

```
source_url1="s3a://{0}:{1}@{2}".format(ak,encoded,aws_bucket1)
```

```
dbutils.fs.mount(source_url1,mount_name1)
```

```
#bucket2
aws_bucket2='aws-bucky-8'
mount_name2='/mnt/b-8new'
```

```
source_url2="s3a://{0}:{1}@{2}".format(ak,encoded,aws_bucket2)
```

```
dbutils.fs.mount(source_url2,mount_name2)
```

```
aws_s3_df1=spark.read.format('csv').option('header','true').option('inferschema','true').load('/mnt/bucket1/P1 data.csv')
aws_s3_df1.show()
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

Importing the client data

```
df1=spark.read.format('csv').option('header','true').option('inferschema','true').load('/mnt/bucket1/P1 data.csv')
```

```
display(df1)
```

Table								
	Row ID ▲	Order ID ▲	Order Date ▲	Ship Date ▲	Ship Mode ▲	Customer ID ▲	Customer Name ▲	Segment
1	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer
2	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate
3	25330	IN-2013-71249	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer
4	13524	ES-2013-1579342	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office
5	47221	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer
6	22732	IN-2013-42360	2013-06-28	2013-07-01	Second Class	JM-15655	Jim Mitchum	Corporate
7	30570	IN-2011-81826	2011-11-07	2011-11-09	First Class	TS-21340	Toby Swindell	Consumer
7,340 rows Truncated data								

```
from pyspark.sql.functions import col
num_rows = df1.count()
num_cols = len(df1.columns)

print((num_rows, num_cols))

(51290, 24)

df1.describe()

Out[9]: DataFrame[summary: string, Row ID: string, Order ID: string, Ship Mode: string, Customer ID: string, Customer Name: string, Segment: string, City: string, State: string, Country: string, Postal Code: string, Market: string, Region: string, Product ID: string, Category: string, Sub-Category: string, Product Name: string, Sales: string, Quantity: string, Discount: string, Profit: string, Shipping Cost: string, Order Priority: string]

df1.printSchema()
```

root	
-- Row ID: integer (nullable = true)	
-- Order ID: string (nullable = true)	
-- Order Date: date (nullable = true)	
-- Ship Date: date (nullable = true)	
-- Ship Mode: string (nullable = true)	
-- Customer ID: string (nullable = true)	
-- Customer Name: string (nullable = true)	
-- Segment: string (nullable = true)	
-- City: string (nullable = true)	
-- State: string (nullable = true)	
-- Country: string (nullable = true)	
-- Postal Code: integer (nullable = true)	
-- Market: string (nullable = true)	
-- Region: string (nullable = true)	
-- Product ID: string (nullable = true)	
-- Category: string (nullable = true)	
-- Sub-Category: string (nullable = true)	
-- Product Name: string (nullable = true)	
-- Sales: string (nullable = true)	
-- Quantity: string (nullable = true)	

Data Cleaning

```
from pyspark.sql.functions import to_date
df2 = df1.withColumn("Order Date", to_date("Order Date", "yyyy-MM-dd"))
display(df2)
```

Table								
	Row ID ▲	Order ID ▲	Order Date ▲	Ship Date ▲	Ship Mode ▲	Customer ID ▲	Customer Name ▲	Segment
1	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer
2	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate

3	25330	IN-2013-71249	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer
4	13524	ES-2013-1579342	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office
5	47221	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer
6	22732	IN-2013-42360	2013-06-28	2013-07-01	Second Class	JM-15655	Jim Mitchum	Corporate
7	30570	IN-2011-81826	2011-11-07	2011-11-09	First Class	TS-21340	Toby Swindell	Consumer
8	31102	MB-2012-86369	2012-04-14	2012-04-18	Standard Class	MB-18085	Mick Brown	Consumer
9	40155	CA-2014-135909	2014-10-14	2014-10-21	Standard Class	JW-15220	Jane Waco	Corporate
10	40936	CA-2012-116638	2012-01-28	2012-01-31	Second Class	JH-15985	Joseph Holt	Consumer
root	34577	CA-2011-102988	2011-04-05	2011-04-09	Second Class	GM-14695	Greg Maxwell	Corporate
-- Row ID: integer (nullable = true)	28879	ID-2012-28402	2012-04-19	2012-04-22	First Class	AJ-10780	Anthony Jacobs	Corporate
-- Order ID: string (nullable = true)	46794	SA-2011-18330	2011-12-27	2011-12-29	Second Class	MM-7260	Magdelene Morse	Consumer
-- Ship Date: date (nullable = true)	27704	IN-2013-73951	2013-06-06	2013-06-08	Second Class	PF-19120	Peter Fuller	Consumer
-- Customer ID: string (nullable = true)	13779	ES-2014-5099955	2014-07-31	2014-08-03	Second Class	BP-11185	Ben Peterman	Corporate
-- Customer Name: string (nullable = true)	31178	CA-2014-41367	2014-11-03	2014-11-06	Second Class	TB-21175	Thomas Boland	Corporate
-- Salesperson: string (nullable = true)								
-- City: string (nullable = true)								
-- State: string (nullable = true)								
-- Country: string (nullable = true)								
-- Postal Code: integer (nullable = true)								
-- Market: string (nullable = true)								
-- Region: string (nullable = true)								
-- Product ID: string (nullable = true)								
-- Category: string (nullable = true)								
-- Sub-Category: string (nullable = true)								
-- Product Name: string (nullable = true)								
-- Sales: string (nullable = true)								
-- Quantity: string (nullable = true)								

Table		
	Country ▲	Order ID count ▲

1	Chad	2
2	Russia	384
3	Paraguay	12
4	Yemen	30
5	Senegal	112
6	Sweden	203
7	Philippines	681

147 rows

```
display(df4.groupby('Product ID').agg(count('Order ID')))
```

Table			
	Product ID ▲	count(Order ID) ▲	
1	FUR-CH-10003365	7	
2	TEC-MA-10001047	2	
3	OFF-ST-10000624	19	
4	FUR-HON-10001558	11	
5	TEC-CO-10002678	1	
6	TEC-MEM-10000374	3	
7	FUR-BO-10000847	5	

10,000 rows | Truncated data

Data Processing and Analysis

1. TOP 5 COUNTRIES WITH HIGHEST QUANTITIES

```
from pyspark.sql.functions import sum, desc, dense_rank
from pyspark.sql.window import Window
```

```
w = Window.orderBy(desc('Total Quantity'))
```

```
top5quantity = df4.groupBy('Country') \
    .agg(sum('Quantity').alias('Total Quantity')) \
    .withColumn('rank', dense_rank().over(w)) \
    .filter(col('rank') <= 5) \
    .drop('rank') \
    .orderBy(desc('Total Quantity'))
```

```
display(top5quantity)
```

Table			
	Country ▲	Total Quantity ▲	
1	United States	58134.36199999998	
2	France	10804	
3	Australia	10673	
4	Mexico	10011	
5	Germany	7745	

5 rows

2. TOP 5 PRODUCTS WITH HIGHEST ORDER COUNT

```
w = Window.orderBy(desc('Order Count'))

top5products = df4.groupBy('Product ID') \
    .agg(count('Order ID').alias('Order Count')) \
    .withColumn('rank', dense_rank().over(w)) \
    .filter(col('rank') <= 5) \
    .drop('rank') \
    .orderBy(desc('Order Count'))

display(top5products)
```

Table

	Product ID ▲	Order Count ▲	
1	OFF-AR-10003651	35	
2	OFF-AR-10003829	31	
3	OFF-BI-10002799	30	
4	OFF-BI-10003708	30	
5	FUR-CH-10003354	28	
6	OFF-BI-10002570	27	
6 rows			

3. TOP 5 PRODUCT BY TOTAL PROFIT

```
top5profit= df4.groupBy('Product Name') \
    .agg(sum('Profit').alias('Total Profit')) \
    .orderBy(desc('Total Profit')) \
    .limit(5)

display(top5profit)
```

Table			
	Product Name ▲	Total Profit ▲	
1	Canon imageCLASS 2200 Advanced Copier	25199.928	
2	Cisco Smart Phone, Full Size	17238.5206	
3	Motorola Smart Phone, Full Size	17027.112999999998	
4	Hoover Stove, Red	11807.969	
5	Sauder Classic Bookcase, Traditional	10672.072999999999	
5 rows			

Writing the processed data to Output S3 bucket

