Silver Layer Research Notebook

Import Libraries

04:13 PM (<1s) 3

```
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

Access Data Lake using Credentials

04:13 PM (<1s) 5

```
#### This allows Data Bricks to access the storage account(Data Lake)
### replace these values with right credentials

## <storage-account> :  YOUR_STORAGE_ACCOUNT_NAME
## <application-id> : YOUR_APPLICATION_ID
## service_credential : YOUR_SECRET_VALUE
## <directory-id> : YOUR_Directory_ID

spark.conf.set("fs.azure.account.auth.type.awdatastoragedatalake.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.awdatastoragedatalake.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.awdatastoragedatalake.dfs.core.windows.net", "979a0794-0963-4b3c-b078-566c59d89823")
spark.conf.set("fs.azure.account.oauth2.client.secret.awdatastoragedatalake.dfs.core.windows.net", "Hlk8Q~wJ8CjPPLWTPUvbHQ-ub0LTAmAPep1CScKg")
spark.conf.set("fs.azure.account.oauth2.client.endpoint.awdatastoragedatalake.dfs.core.windows.net", "https://login.microsoftonline.com/aa232db2-7a78-4414-a529-33db9124cba7/oauth2/
token")
```

Load Data

04:13 PM (1s) 7

```
# Load AdventureWorks_Calendar dataset
df_cal = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Calendar")
```

(2) Spark Jobs

df_cal: pyspark.sql.dataframe.DataFrame = [Date: date]

04:13 PM (1s) 8

```
# Load AdventureWorks_Customers dataset
df_custom = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/
AdventureWorks_Customers")
```

(2) Spark Jobs

df_custom: pyspark.sql.dataframe.DataFrame = [CustomerKey: integer, Prefix: string ... 11 more fields]

04:13 PM (<1s) 9

```
# Load AdventureWorks_Product_Categories dataset
df_prod_cat = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/
AdventureWorks_Product_Categories")
```

(2) Spark Jobs

df_prod_cat: pyspark.sql.dataframe.DataFrame = [ProductCategoryKey: integer, CategoryName: string]

04:13 PM (1s) 10

```
# Load AdventureWorks_Products dataset
df_prod = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Products")
```

(2) Spark Jobs

df_prod: pyspark.sql.dataframe.DataFrame = [ProductKey: integer, ProductSubcategoryKey: integer ... 9 more fields]

04:13 PM (1s) 11

```
# Load AdventureWorks_Returns dataset
df_returns = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Returns")
```

(2) Spark Jobs

df_returns: pyspark.sql.dataframe.DataFrame = [ReturnDate: date, TerritoryKey: integer ... 2 more fields]

04:13 PM (1s) 12

```
# Concatenate all the Sales_2015, Sales_2016, Sales_2017 datasets and Load
df_sales = spark.read.format('csv')\
            .option("header",True)\
            .option("inferSchema",True)\
            .load('abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Sales*')
```

(2) Spark Jobs

df_sales: pyspark.sql.dataframe.DataFrame = [OrderDate: date, StockDate: date ... 6 more fields]

04:13 PM (<1s) 13

```
# Load AdventureWorks_Territories dataset
df_terr = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/
AdventureWorks_Territories")
```

(2) Spark Jobs

df_terr: pyspark.sql.dataframe.DataFrame = [SalesTerritoryKey: integer, Region: string ... 2 more fields]

04:13 PM (<1s) 14

```
# Load Product_Subcategories dataset
df_sub = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("abfss://bronze@awdatastoragedatalake.dfs.core.windows.net/Product_Subcategories")
```

(2) Spark Jobs

df_sub: pyspark.sql.dataframe.DataFrame = [ProductSubcategoryKey: integer, SubcategoryName: string ... 1 more field]

Transformation

1.Calendar dataset

04:13 PM (<1s) 17

```
# Extract new columns 'Month' and 'Year
df_cal = df_cal.withColumn('Month', month(col('Date'))).withColumn('Year', year(col('Date')))
```

df_cal: pyspark.sql.dataframe.DataFrame = [Date: date, Month: integer ... 1 more field]

04:13 PM (1s) 18

```
# push new calendar dataset to datalake (silver)
df_cal.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Calendar").save()
```

(1) Spark Jobs

2.Customer dataset

04:13 PM (<1s) 20

```
# Concatenate 'Prefix','FirstName','LastName' columns to create a new column named 'Full Name'
df_custom = df_custom.withColumn('Full Name', concat(col('Prefix'), lit(' '), col('FirstName'), lit(' '), col('LastName')))
```

df_custom: pyspark.sql.dataframe.DataFrame = [CustomerKey: integer, Prefix: string ... 12 more fields]

04:13 PM (1s) 21

```
# push new Customer dataset to datalake (silver)
df_custom.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Customers").save()
```

(1) Spark Jobs

3.Products categories dataset

04:13 PM (1s) 23

```
# No Transformation
# push new AdventureWorks_Returns dataset to datalake (silver)
df_prod_cat.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Product_Categories").save()
```

(1) Spark Jobs

4.Products dataset

04:13 PM (<1s) 25

```
# Get first index from the 'productSKU' & 'productName' columns
df_prod = df_prod.withColumn("productSKU",split(col("productSKU"),"-")[0]).withColumn("productName",split(col("productName")," ")[0])
```

df_prod: pyspark.sql.dataframe.DataFrame = [ProductKey: integer, ProductSubcategoryKey: integer ... 9 more fields]

04:13 PM (1s) 26

```
# push new products dataset to datalake (silver)
df_prod.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Products").save()
```

(1) Spark Jobs

5.AdventureWorks_Returns

04:13 PM (1s) 28

```
# No Transformation
# push new AdventureWorks_Returns dataset to datalake (silver)
df_returns.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Returns").save()
```

(1) Spark Jobs

6.Product Sales 2015, 2016, 2017 dataset

04:13 PM (<1s) 30

```
# 1. convert 'StockDate' column into Time-Stamp Date format
df_sales = df_sales.withColumn('StockDate', to_timestamp(col('StockDate')))

# 2. replace S -> T
df_sales = df_sales.withColumn('OrderNumber', regexp_replace('OrderNumber', 'S', 'T'))

# 3. Multiply "OrderLineItem" and "OrderQuantity"
df_sales = df_sales.withColumn('OrderQuantity * OrderLineItem', col('OrderQuantity') * col('OrderLineItem'))
```

df_sales: pyspark.sql.dataframe.DataFrame = [OrderDate: date, StockDate: timestamp ... 7 more fields]

04:13 PM (1s) 31

```
# push new Product Sales dataset to datalake (silver)
df_sales.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/Product_Sales").save()
```

(1) Spark Jobs

7.AdventureWorks_Territories

```
04:13 PM (<1s)                                                    33

# No Transformation
# push new AdventureWorks_Territories dataset to datalake (silver)
df_terr.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/AdventureWorks_Territories").save()

(1) Spark Jobs
```

8.Product Subcategories

```
04:13 PM (1s)                                                     35

# No Transformation
# push new Product Subcategories dataset to datalake (silver)
df_sub.write.format("parquet").mode("append").option("path", "abfss://silver@awdatastoragedatalake.dfs.core.windows.net/Product_Subcategories").save()

(1) Spark Jobs
```

Data Analysis

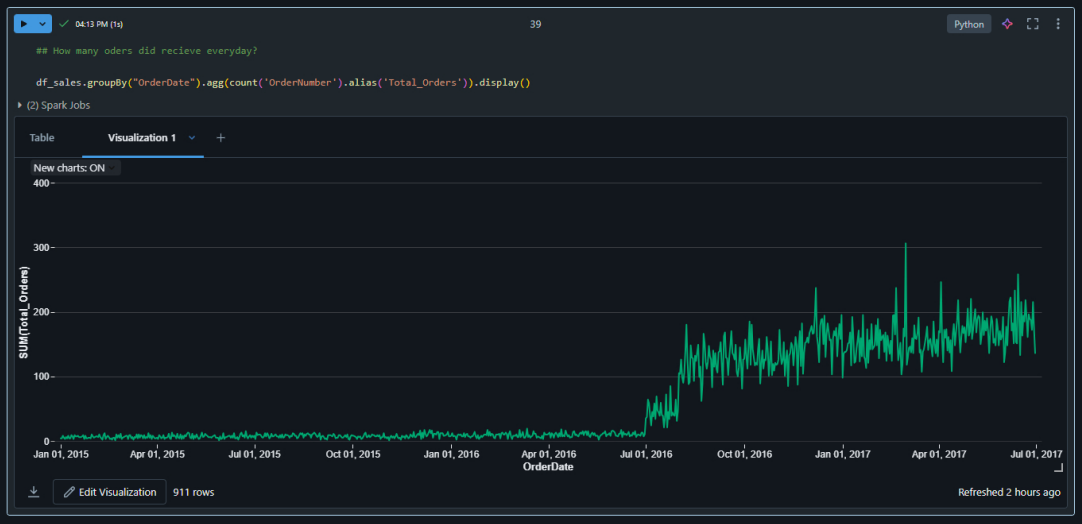
1.Sales Data Analysis

```
04:13 PM (<1s)                                                    38

df_sales.show(n=10, truncate=False)

(1) Spark Jobs
```

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	OrderQuantity * OrderLineItem
2017-01-01	2003-12-13 00:00:00	TO61285	529	23791	1	2	2	4
2017-01-01	2003-09-24 00:00:00	TO61285	214	23791	1	3	1	3
2017-01-01	2003-09-04 00:00:00	TO61285	540	23791	1	1	1	1
2017-01-01	2003-09-28 00:00:00	TO61301	529	16747	1	2	2	4
2017-01-01	2003-10-21 00:00:00	TO61301	377	16747	1	1	1	1
2017-01-01	2003-10-23 00:00:00	TO61301	540	16747	1	3	1	3
2017-01-01	2003-09-04 00:00:00	TO61269	215	11792	4	1	1	1
2017-01-01	2003-10-21 00:00:00	TO61269	229	11792	4	2	1	2
2017-01-01	2003-10-24 00:00:00	TO61286	528	11530	6	2	2	4
2017-01-01	2003-09-27 00:00:00	TO61286	536	11530	6	1	2	2

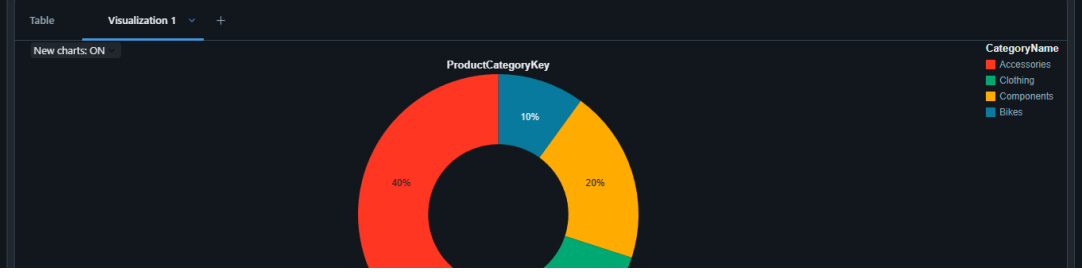
only showing top 10 rows

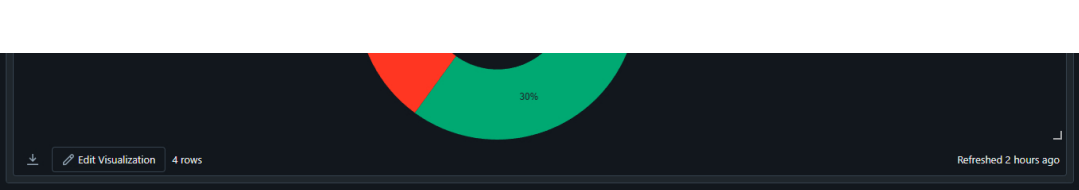
```
04:13 PM (1s)                                                     39                                                Python

## How many oders did recieve everyday?

df_sales.groupBy("OrderDate").agg(count('OrderNumber').alias('Total_Orders')).display()

(2) Spark Jobs
```



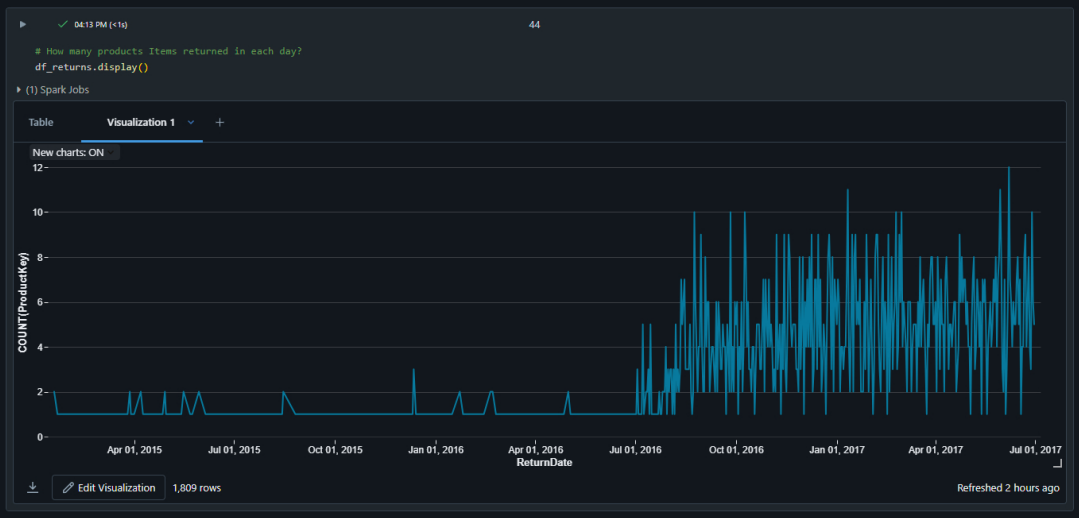
2.Product Categories Data Analysis

```
04:13 PM (<1s)                                                    42

# show all product categories
df_prod_cat.display()

(1) Spark Jobs
```





3.Product Retuen Data Analysis



[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts