PySpark: Cleaning Data and Getting insights from data

- There are 3 Parts: Installing Spark, Loading & Data cleaning and Getting insights from data. These are the entry points for any data analytics project
- In my earlier notebooks, I discussed in detail about Spark installationa and uploading data in Colab. This notebook is focused on Data cleansing.
- Data cleansing is the process of analyzing the quality of data in a data source, approving/rejecting the suggestions by the system and making changes to the data.The quality of data is important in getting useful information from it.

PART 1. Configure PySpark environment

Copy & Paste code below.

Read more
https://github.com/kyramichel/Pyspark_Cloud/blob/master/PySpark_GoogleColab.ipynb

```
#update the packages existing on the machine
!apt-get update

#install java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null


#install spark: get the file
!wget -q https://archive.apache.org/dist/spark/spark-2.4.1/spark-2.4.1-bin-hadoop2.7.tgz

#unzip the file
!tar xf spark-2.4.1-bin-hadoop2.7.tgz

#set up the ennvironmental variables
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.3.2-bin-hadoop2.7"

#install finspark
!pip install -q findspark

#importing findspark adds pyspark to the system path, so that next time you can import pyspar
import findspark
findspark.init("/content/spark-2.4.1-bin-hadoop2.7")

import pyspark
```

```
#SparkContext: the entry point of spark functionality is the interface to running a spark clu
from pyspark import SparkContext, SparkConf


#import a spark session
from pyspark.sql import SparkSession
#create a session
spark = SparkSession.builder.getOrCreate()
spark

#test the installation
df0 = spark.sql("select 'PySpark' as Hello")
df0.show()
```

```
Hit:1 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease
Ign:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRele
Hit:3 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Ign:6 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x8
Hit:7 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Releas
Hit:8 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x8
Hit:9 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:10 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:11 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Hit:12 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Hit:13 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Reading package lists... Done
+-------+
|  Hello|
+-------+
|PySpark|
+-------+
```

PART 2. Upload, Load & Clean Data

- To upload data, click upload, select your data file Read more how to get in data in Colab: https://github.com/kyramichel/Pyspark_Cloud/blob/master/DataPysparkCloudColab.ipynb
- Load data, create a data frame df
- To get insights from data we can query a data frame in Spark using both Python and SQL

```
df = spark.read.csv('data.csv', header=True, inferSchema=True)
df.show()
```

```
+--------+---------+------+-------------+--------+----------+
| Product|    Price|Region|      Country|Latitude| Longitude|
+--------+---------+------+-------------+--------+----------+
```

```
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|
+--------+---------+------+-------------+--------+----------+
only showing top 20 rows
```

```
df.printSchema()
```

```
root
 |-- Product: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
```

```
#Import pyspark sql functions library to clean data
from pyspark.sql.functions import *


#clean Region column - we create a new col because df is immutable
df1 = df.withColumn("RegionCleaned", when(df.Region.isNull(), 'unknown').otherwise(df.Region)
df1.show()
```

```
+--------+---------+------+-------------+--------+----------+-------------+
| Product|    Price|Region|      Country|Latitude| Longitude|RegionCleaned|
+--------+---------+------+-------------+--------+----------+-------------+
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|           MO|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|           OR|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|           AL|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|           NJ|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|           IL|
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|           TN|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|           TX|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|           ID|
```

```
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|           NJ|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|           UT|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|           CA|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|           TX|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|           IL|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|           CA|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|           FL|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|           NY|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|           VA|
+--------+---------+------+-------------+--------+----------+-------------+
only showing top 20 rows
```

```
df1.select("Region", "RegionCleaned").show()
```

```
+------+-------------+
|Region|RegionCleaned|
+------+-------------+
|    MO|           MO|
|    OR|           OR|
|    AL|           AL|
|    NJ|           NJ|
|    IL|           IL|
|    TN|           TN|
|    NY|           NY|
|    TX|           TX|
|    ID|           ID|
|    NJ|           NJ|
|    UT|           UT|
|    CA|           CA|
|    TX|           TX|
|    NY|           NY|
|    IL|           IL|
|    CA|           CA|
|    NY|           NY|
|    FL|           FL|
|    NY|           NY|
|    VA|           VA|
+------+-------------+
only showing top 20 rows
```

```
df1.drop("Region")
df1.show()
```

```
+--------+---------+------+-------------+--------+----------+-------------+
| Product|    Price|Region|      Country|Latitude| Longitude|RegionCleaned|
+--------+---------+------+-------------+--------+----------+-------------+
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|           MO|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|           OR|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|           AL|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|           NJ|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|           IL|
```

```
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|           TN|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|           TX|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|           ID|
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|           NJ|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|           UT|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|           CA|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|           TX|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|           IL|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|           CA|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|           NY|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|           FL|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|           NY|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|           VA|
+--------+---------+------+-------------+--------+----------+-------------+
only showing top 20 rows
```

```
df1 = df1.withColumnRenamed("RegionCleaned","Region")
df.show()
```

```
+--------+---------+------+-------------+--------+----------+
| Product|    Price|Region|      Country|Latitude| Longitude|
+--------+---------+------+-------------+--------+----------+
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|
+--------+---------+------+-------------+--------+----------+
only showing top 20 rows
```

```
#Use filter to delete entire row when Country is Null

df1 = df.filter(df.Country.isNotNull())
df1.show()
```

```
+--------+---------+------+-------------+--------+----------+
```

```
| Product|    Price|Region|      Country|Latitude| Longitude|
+--------+---------+------+-------------+--------+----------+
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|
+--------+---------+------+-------------+--------+----------+
only showing top 20 rows
```

```
df2 = df1.withColumn("PriceCleaned",
                     when(col("Product") == "Product1","1200")
                     .when(col("Product") == "Product2","3600")
                     .otherwise("7500"))
df2.show()
```

```
+--------+---------+------+-------------+--------+----------+------------+
| Product|    Price|Region|      Country|Latitude| Longitude|PriceCleaned|
+--------+---------+------+-------------+--------+----------+------------+
|Product1|$1,200.00|    MO|United States|  39.195| -94.68194|        1200|
|Product1|$1,200.00|    OR|United States|46.18806|   -123.83|        1200|
|Product2|$3,600.00|    AL|United States|33.52056|  -86.8025|        3600|
|Product1|$1,200.00|    NJ|United States|   39.79| -75.23806|        1200|
|Product1|$1,200.00|    IL|United States|40.69361| -89.58889|        1200|
|Product1|$1,200.00|    TN|United States|36.34333| -88.85028|        1200|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|        1200|
|Product1|$1,200.00|    TX|United States|29.42389| -98.49333|        1200|
|Product1|$1,200.00|    ID|United States|43.69556|-116.35306|        1200|
|Product1|$1,200.00|    NJ|United States|40.03222| -74.95778|        1200|
|Product1|$1,200.00|    UT|United States|40.76083|-111.89028|        1200|
|Product1|$1,200.00|    CA|United States|   32.64|-117.08333|        1200|
|Product1|$1,200.00|    TX|United States|29.61944| -95.63472|        1200|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|        1200|
|Product1|$1,200.00|    IL|United States|40.61278| -89.45917|        1200|
|Product1|$1,200.00|    CA|United States|37.22667|-121.97361|        1200|
|Product1|$1,200.00|    NY|United States|40.71417| -74.00639|        1200|
|Product1|$1,200.00|    FL|United States|25.77389| -80.19389|        1200|
|Product1|$1,200.00|    NY|United States|   40.65|    -73.95|        1200|
|Product1|$1,200.00|    VA|United States|38.96861| -77.34139|        1200|
+--------+---------+------+-------------+--------+----------+------------+
```

```
only showing top 20 rows
```

```
df2.printSchema()
```

```
root
 |-- Product: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- PriceCleaned: string (nullable = false)
```

```
df2 = df2.withColumn("PriceNum", df2["PriceCleaned"].cast("float"))
df2.printSchema()
```

```
root
 |-- Product: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- PriceCleaned: string (nullable = false)
 |-- PriceNum: float (nullable = true)
```

```
df3 = df2.drop("Price", "PriceCleaned")
df3.show()
```

```
+--------+------+-------------+--------+----------+--------+
| Product|Region|      Country|Latitude| Longitude|PriceNum|
+--------+------+-------------+--------+----------+--------+
|Product1|    MO|United States|  39.195| -94.68194|  1200.0|
|Product1|    OR|United States|46.18806|   -123.83|  1200.0|
|Product2|    AL|United States|33.52056|  -86.8025|  3600.0|
|Product1|    NJ|United States|   39.79| -75.23806|  1200.0|
|Product1|    IL|United States|40.69361| -89.58889|  1200.0|
|Product1|    TN|United States|36.34333| -88.85028|  1200.0|
|Product1|    NY|United States|40.71417| -74.00639|  1200.0|
|Product1|    TX|United States|29.42389| -98.49333|  1200.0|
|Product1|    ID|United States|43.69556|-116.35306|  1200.0|
|Product1|    NJ|United States|40.03222| -74.95778|  1200.0|
|Product1|    UT|United States|40.76083|-111.89028|  1200.0|
|Product1|    CA|United States|   32.64|-117.08333|  1200.0|
|Product1|    TX|United States|29.61944| -95.63472|  1200.0|
|Product1|    NY|United States|40.71417| -74.00639|  1200.0|
|Product1|    IL|United States|40.61278| -89.45917|  1200.0|
|Product1|    CA|United States|37.22667|-121.97361|  1200.0|
|Product1|    NY|United States|40.71417| -74.00639|  1200.0|
|Product1|    FL|United States|25.77389| -80.19389|  1200.0|
```

```
|Product1|    NY|United States|   40.65|    -73.95|  1200.0|
|Product1|    VA|United States|38.96861| -77.34139|  1200.0|
+--------+------+-------------+--------+----------+--------+
only showing top 20 rows
```

```
df3 = df3.withColumnRenamed("PriceNum","Price")
df3.show()
```

```
+--------+------+-------------+--------+----------+------+
| Product|Region|      Country|Latitude| Longitude| Price|
+--------+------+-------------+--------+----------+------+
|Product1|    MO|United States|  39.195| -94.68194|1200.0|
|Product1|    OR|United States|46.18806|   -123.83|1200.0|
|Product2|    AL|United States|33.52056|  -86.8025|3600.0|
|Product1|    NJ|United States|   39.79| -75.23806|1200.0|
|Product1|    IL|United States|40.69361| -89.58889|1200.0|
|Product1|    TN|United States|36.34333| -88.85028|1200.0|
|Product1|    NY|United States|40.71417| -74.00639|1200.0|
|Product1|    TX|United States|29.42389| -98.49333|1200.0|
|Product1|    ID|United States|43.69556|-116.35306|1200.0|
|Product1|    NJ|United States|40.03222| -74.95778|1200.0|
|Product1|    UT|United States|40.76083|-111.89028|1200.0|
|Product1|    CA|United States|   32.64|-117.08333|1200.0|
|Product1|    TX|United States|29.61944| -95.63472|1200.0|
|Product1|    NY|United States|40.71417| -74.00639|1200.0|
|Product1|    IL|United States|40.61278| -89.45917|1200.0|
|Product1|    CA|United States|37.22667|-121.97361|1200.0|
|Product1|    NY|United States|40.71417| -74.00639|1200.0|
|Product1|    FL|United States|25.77389| -80.19389|1200.0|
|Product1|    NY|United States|   40.65|    -73.95|1200.0|
|Product1|    VA|United States|38.96861| -77.34139|1200.0|
+--------+------+-------------+--------+----------+------+
only showing top 20 rows
```

```
df3.dtypes
```

```
[('Product', 'string'),
 ('Region', 'string'),
 ('Country', 'string'),
 ('Latitude', 'double'),
 ('Longitude', 'double'),
 ('Price', 'float')]
```

To fill missing Latitude and Longitude values I using different interpolation techniques: mean and median imputation

```
#clean lat column - replace null with 0
df4 = df3.withColumn("Lat1", when(df3.Latitude.isNull(), 0).otherwise(df.Latitude))
df4.printSchema()
```

```
root
 |-- Product: string (nullable = true)
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Price: float (nullable = true)
 |-- Lat1: double (nullable = true)
```

Calculate mean(Latitude) grouped by Country:

```
from pyspark.sql.functions import avg, col, when
from pyspark.sql.window import Window
w = Window().partitionBy('Country')
```

```
df5 = df4.withColumn('Latitude', when(col('Latitude').isNull(), avg(col('Lat1')).over(w)).oth
df5.show()
```

```
+--------+--------------+-----------+----------+-----------+------+----------+
| Product|        Region|    Country|  Latitude|  Longitude| Price|      Lat1|
+--------+--------------+-----------+----------+-----------+------+----------+
|Product2|   Moscow City|     Russia|55.6283333| 37.6608333|3600.0|55.6283333|
|Product1|     Stockholm|     Sweden|59.2833333|       18.3|1200.0|59.2833333|
|Product2|         Skane|     Sweden|      55.6|       13.0|3600.0|      55.6|
|Product1|  Vasterbotten|     Sweden|63.8333333|      20.25|1200.0|63.8333333|
|Product1|     Stockholm|     Sweden|59.3333333|      18.05|1200.0|59.3333333|
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3|
|Product1|       Uppsala|     Sweden|60.3333333|       17.5|1200.0|60.3333333|
|Product1|     Stockholm|     Sweden|59.4166667| 18.0166667|1200.0|59.4166667|
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3|
|Product1|     Stockholm|     Sweden|59.3833333| 18.0666667|1200.0|59.3833333|
|Product1|     Stockholm|     Sweden|59.3166667| 18.1166667|1200.0|59.3166667|
|Product2|     Stockholm|     Sweden|59.3666667| 18.1333333|3600.0|59.3666667|
|Product2|  Ostergotland|     Sweden|      58.6| 16.1833333|3600.0|      58.6|
|Product1|     Stockholm|     Sweden|      59.5|      18.05|1200.0|      59.5|
|Product1|          null|     Jersey|      49.2| -2.0333333|1200.0|      49.2|
|Product1|         Bohol|Philippines| 9.9136111|124.0927778|1200.0| 9.9136111|
|Product1|General Santos|Philippines| 6.1036111|125.2163889|1200.0| 6.1036111|
|Product1|  Kuala Lumpur|   Malaysia| 3.1666667|      101.7|1200.0| 3.1666667|
|Product1|      Istanbul|     Turkey|40.6166667| 29.0666667|1200.0|40.6166667|
|Product1|      Istanbul|     Turkey|41.1980556| 29.0302778|1200.0|41.1980556|
+--------+--------------+-----------+----------+-----------+------+----------+
only showing top 20 rows
```

For Longitute I apply interpolation using a median=startegy

```
df6 = df5.withColumn("Long1", when(df5.Longitude.isNull(), 0).otherwise(df5.Longitude))
df6.show()
```

```
df6.show()
```

```
+--------+--------------+-----------+----------+-----------+------+----------+----------
| Product|        Region|    Country|  Latitude|  Longitude| Price|      Lat1|      Long
+--------+--------------+-----------+----------+-----------+------+----------+----------
|Product2|   Moscow City|     Russia|55.6283333| 37.6608333|3600.0|55.6283333| 37.660833
|Product1|     Stockholm|     Sweden|59.2833333|       18.3|1200.0|59.2833333|       18
|Product2|         Skane|     Sweden|      55.6|       13.0|3600.0|      55.6|       13
|Product1|  Vasterbotten|     Sweden|63.8333333|      20.25|1200.0|63.8333333|      20.2
|Product1|     Stockholm|     Sweden|59.3333333|      18.05|1200.0|59.3333333|      18.0
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3| 18.166666
|Product1|       Uppsala|     Sweden|60.3333333|       17.5|1200.0|60.3333333|       17
|Product1|     Stockholm|     Sweden|59.4166667| 18.0166667|1200.0|59.4166667| 18.016666
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3| 18.166666
|Product1|     Stockholm|     Sweden|59.3833333| 18.0666667|1200.0|59.3833333| 18.066666
|Product1|     Stockholm|     Sweden|59.3166667| 18.1166667|1200.0|59.3166667| 18.116666
|Product2|     Stockholm|     Sweden|59.3666667| 18.1333333|3600.0|59.3666667| 18.133333
|Product2|  Ostergotland|     Sweden|      58.6| 16.1833333|3600.0|      58.6| 16.183333
|Product1|     Stockholm|     Sweden|      59.5|      18.05|1200.0|      59.5|      18.0
|Product1|          null|     Jersey|      49.2| -2.0333333|1200.0|      49.2| -2.033333
|Product1|         Bohol|Philippines| 9.9136111|124.0927778|1200.0| 9.9136111|124.092777
|Product1|General Santos|Philippines| 6.1036111|125.2163889|1200.0| 6.1036111|125.216388
|Product1|  Kuala Lumpur|   Malaysia| 3.1666667|      101.7|1200.0| 3.1666667|      101
|Product1|      Istanbul|     Turkey|40.6166667| 29.0666667|1200.0|40.6166667| 29.066666
|Product1|      Istanbul|     Turkey|41.1980556| 29.0302778|1200.0|41.1980556| 29.030277
+--------+--------------+-----------+----------+-----------+------+----------+----------
only showing top 20 rows
```

```
longCol = df6.select("Long1")
longCol.show()
```

```
+----------+
|     Long1|
+----------+
| -94.68194|
|   -123.83|
|  -86.8025|
| -75.23806|
| -89.58889|
| -88.85028|
| -74.00639|
| -98.49333|
|-116.35306|
| -74.95778|
|-111.89028|
|-117.08333|
| -95.63472|
| -74.00639|
| -89.45917|
|-121.97361|
| -74.00639|
| -80.19389|
|    -73.95|
| -77.34139|
```

```
+----------+
only showing top 20 rows
```

```
#Using LinAlgebra Python library to compute median
import numpy as np
median = np.median(longCol.collect())
median
```

```
-73.7275
```

```
#replace missing Longitude values with median
from pyspark.sql.functions import lit

df7 = df6.withColumn('Longitude', when(col('Longitude').isNull(), lit(median)).otherwise(col(
df7.show()
```

```
+--------+--------------+-----------+----------+-----------+------+----------+----------
| Product|        Region|    Country|  Latitude|  Longitude| Price|      Lat1|      Long
+--------+--------------+-----------+----------+-----------+------+----------+----------
|Product2|   Moscow City|     Russia|55.6283333| 37.6608333|3600.0|55.6283333| 37.660833
|Product1|     Stockholm|     Sweden|59.2833333|       18.3|1200.0|59.2833333|       18
|Product2|         Skane|     Sweden|      55.6|       13.0|3600.0|      55.6|       13
|Product1|  Vasterbotten|     Sweden|63.8333333|      20.25|1200.0|63.8333333|      20.2
|Product1|     Stockholm|     Sweden|59.3333333|      18.05|1200.0|59.3333333|      18.0
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3| 18.166666
|Product1|       Uppsala|     Sweden|60.3333333|       17.5|1200.0|60.3333333|       17
|Product1|     Stockholm|     Sweden|59.4166667| 18.0166667|1200.0|59.4166667| 18.016666
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|      59.3| 18.166666
|Product1|     Stockholm|     Sweden|59.3833333| 18.0666667|1200.0|59.3833333| 18.066666
|Product1|     Stockholm|     Sweden|59.3166667| 18.1166667|1200.0|59.3166667| 18.116666
|Product2|     Stockholm|     Sweden|59.3666667| 18.1333333|3600.0|59.3666667| 18.133333
|Product2|  Ostergotland|     Sweden|      58.6| 16.1833333|3600.0|      58.6| 16.183333
|Product1|     Stockholm|     Sweden|      59.5|      18.05|1200.0|      59.5|      18.0
|Product1|          null|     Jersey|      49.2| -2.0333333|1200.0|      49.2| -2.033333
|Product1|         Bohol|Philippines| 9.9136111|124.0927778|1200.0| 9.9136111|124.092777
|Product1|General Santos|Philippines| 6.1036111|125.2163889|1200.0| 6.1036111|125.216388
|Product1|  Kuala Lumpur|   Malaysia| 3.1666667|      101.7|1200.0| 3.1666667|      101
|Product1|      Istanbul|     Turkey|40.6166667| 29.0666667|1200.0|40.6166667| 29.066666
|Product1|      Istanbul|     Turkey|41.1980556| 29.0302778|1200.0|41.1980556| 29.030277
+--------+--------------+-----------+----------+-----------+------+----------+----------
only showing top 20 rows
```

```
df7 = df6.drop("Lat1", "Long1")
df7.show()
```

```
+--------+--------------+-----------+----------+-----------+------+
| Product|        Region|    Country|  Latitude|  Longitude| Price|
+--------+--------------+-----------+----------+-----------+------+
|Product2|   Moscow City|     Russia|55.6283333| 37.6608333|3600.0|
|Product1|     Stockholm|     Sweden|59.2833333|       18.3|1200.0|
```

```
|Product2|         Skane|     Sweden|      55.6|       13.0|3600.0|
|Product1|  Vasterbotten|     Sweden|63.8333333|      20.25|1200.0|
|Product1|     Stockholm|     Sweden|59.3333333|      18.05|1200.0|
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|
|Product1|       Uppsala|     Sweden|60.3333333|       17.5|1200.0|
|Product1|     Stockholm|     Sweden|59.4166667| 18.0166667|1200.0|
|Product1|     Stockholm|     Sweden|      59.3| 18.1666667|1200.0|
|Product1|     Stockholm|     Sweden|59.3833333| 18.0666667|1200.0|
|Product1|     Stockholm|     Sweden|59.3166667| 18.1166667|1200.0|
|Product2|     Stockholm|     Sweden|59.3666667| 18.1333333|3600.0|
|Product2|  Ostergotland|     Sweden|      58.6| 16.1833333|3600.0|
|Product1|     Stockholm|     Sweden|      59.5|      18.05|1200.0|
|Product1|          null|     Jersey|      49.2| -2.0333333|1200.0|
|Product1|         Bohol|Philippines| 9.9136111|124.0927778|1200.0|
|Product1|General Santos|Philippines| 6.1036111|125.2163889|1200.0|
|Product1|  Kuala Lumpur|   Malaysia| 3.1666667|      101.7|1200.0|
|Product1|      Istanbul|     Turkey|40.6166667| 29.0666667|1200.0|
|Product1|      Istanbul|     Turkey|41.1980556| 29.0302778|1200.0|
+--------+--------------+-----------+----------+-----------+------+
only showing top 20 rows
```

PART 3. Getting insights from our data

Q: Which Product has the highest sale?

```
group_data = df7.groupBy("Product").agg({'Product':'count'})
group_data.show()
```

```
+--------+--------------+
| Product|count(Product)|
+--------+--------------+
|Product3|            15|
|Product2|           135|
|Product1|           845|
+--------+--------------+
```

```
#Product1 has the highest sales
group_data.agg({'count(Product)':'max'}).show()
```

```
+------------------+
|max(count(Product))|
+------------------+
|               845|
+------------------+
```

Q:Which Country sells better (all products)?

```
group_data2 = df7.groupBy("Country").agg({'Product':'count'})
group_data2.show()
```

```
+-------------+--------------+
|      Country|count(Product)|
+-------------+--------------+
|       Russia|             1|
|       Sweden|            13|
|  Philippines|             2|
|       Jersey|             1|
|     Malaysia|             1|
|       Turkey|             6|
|      Germany|            25|
|       France|            27|
|       Greece|             1|
|    Argentina|             1|
|      Belgium|             8|
|      Finland|             2|
|United States|           461|
|        India|             2|
|        China|             1|
|        Malta|             2|
|       Kuwait|             1|
|        Italy|            15|
|       Norway|            16|
|        Spain|            12|
+-------------+--------------+
only showing top 20 rows
```

```
#US sells better: 461
group_data2 = df5.groupBy("Country").agg({'Product':'count'}).sort(col("count(Product)").desc
group_data2.show()
```

```
+--------------------+--------------+
|             Country|count(Product)|
+--------------------+--------------+
|       United States|           461|
|      United Kingdom|           100|
|              Canada|            76|
|             Ireland|            49|
|           Australia|            38|
|         Switzerland|            36|
|              France|            27|
|             Germany|            25|
|         Netherlands|            22|
|              Norway|            16|
|             Denmark|            15|
|               Italy|            15|
|              Sweden|            13|
```

```
|               Spain|           12|
|             Belgium|            8|
|             Austria|            7|
|              Turkey|            6|
|United Arab Emirates|            6|
|         New Zealand|            6|
|              Brazil|            5|
+--------------------+-------------+
only showing top 20 rows
```

Q:Which Country sells better per product?

```
#Breakdown by products
group_data3= df7.groupBy("Country", "Product").agg({'Product':'count'}).sort(col("count(Produ
group_data3.show()
```

```
+--------------+--------+--------------+
|       Country| Product|count(Product)|
+--------------+--------+--------------+
| United States|Product1|           399|
|United Kingdom|Product1|            90|
|        Canada|Product1|            62|
| United States|Product2|            54|
|       Ireland|Product1|            46|
|     Australia|Product1|            30|
|   Switzerland|Product1|            22|
|        France|Product1|            20|
|       Germany|Product1|            20|
|   Netherlands|Product1|            16|
|        Norway|Product1|            15|
|       Denmark|Product1|            15|
|   Switzerland|Product2|            14|
|        Canada|Product2|            14|
|         Spain|Product1|            11|
|United Kingdom|Product2|            10|
|        Sweden|Product1|            10|
|         Italy|Product1|            10|
|     Australia|Product2|             8|
| United States|Product3|             8|
+--------------+--------+--------------+
only showing top 20 rows
```

Breakdown by Region (state) per Country

```
group_data4= df5.groupBy("Country", "Region", "Product").agg({'Product':'count'}).orderBy("Co
group_data4.show()
```

```
+---------+--------------------+--------+--------------+
```

```
|  Country|              Region| Product|count(Product)|
+---------+--------------------+--------+--------------+
|Argentina|        Buenos Aires|Product1|             1|
|Australia|     New South Wales|Product1|             8|
|Australia|   Western Australia|Product2|             2|
|Australia|     South Australia|Product1|             1|
|Australia|            Victoria|Product2|             1|
|Australia|     New South Wales|Product2|             3|
|Australia|   Western Australia|Product1|             4|
|Australia|            Tasmania|Product2|             1|
|Australia|            Victoria|Product1|             8|
|Australia|          Queensland|Product1|             9|
|Australia|          Queensland|Product2|             1|
|  Austria|       Lower Austria|Product1|             2|
|  Austria|              Vienna|Product1|             4|
|  Austria|               Tyrol|Product2|             1|
|  Bahrain|          Al Manamah|Product1|             1|
|  Belgium|           Antwerpen|Product1|             1|
|  Belgium|             Hainaut|Product1|             1|
|  Belgium|Brussels (Bruxelles)|Product1|             5|
|  Belgium|Brussels (Bruxelles)|Product2|             1|
|  Bermuda|            Hamilton|Product1|             1|
+---------+--------------------+--------+--------------+
only showing top 20 rows
```