Data Glacier

Your Deep Learning Partner

Data Glacier Data Scientist Internship

Batch: LISUM23: 30

Week11: EDA Presentation and proposed modeling technique

Project: Retail Forecasting

Team member's details:

Group Name: Retail_forecasting

Name	Richa Mishra	Shalu Kumar	Madoka Fujii	Shushun Ren
Email	mricha828@gmail.com	ss4676@njit.edu	mdkfji@gmail.com	shushunr@umich.edu
Country	United States	United States	United States	United States
College/Company	NJIT	NJIT	Sumitomo Mitsui Trust Bank	Umich
Specialization	Data Science	Data Science	Data Science	Data Science

Project life cycle along with deadline:

Project weeks	Deadline	Lifecycle
Week7	Aug 19, 2023	Problem statement, Pre-process
Week8	Aug 26, 2023	Data process, understanding
Week9	Sep 02, 2023	Data Cleaning, Merge, Review
Week10	Sep 09, 2023	EDA, Final recommendation
Week11	**Sep 16, 2023**	**EDA presentation for business users**
Week12	Sep 23, 2023	Model Selection and Model Building/Dashboard
Week13	Sep 30, 2023	Final Project Report and Code

Tabular data details: forecasting_case_study.xlsx:

Total number of observations	1218
Total number of files	1
Total number of features	12
Base format of the file	.xlsx
Size of the data	80KB

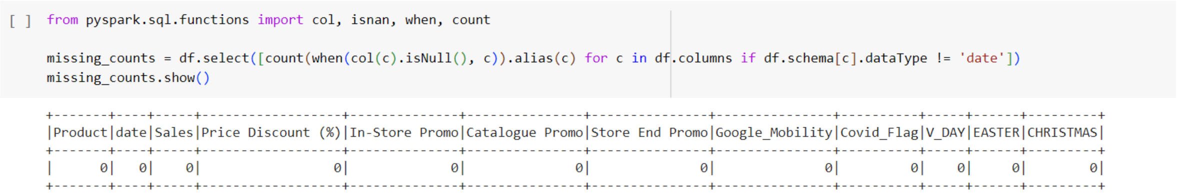
Problem Description:

This major Australian beverage corporation operates within the beverage industry. Their product distribution spans across multiple supermarket chains, and they actively conduct robust promotional campaigns year-round. The demand for their products is subject to fluctuations driven by factors such as holidays and seasonal trends. They require a weekly item-level forecast for each of their products, categorized into weekly intervals.

Data Preparation:

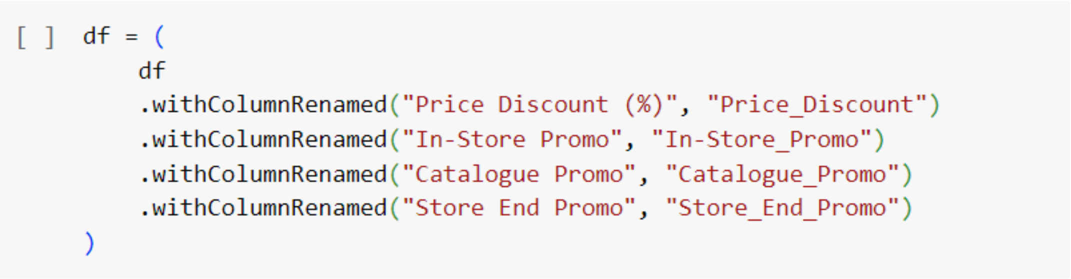
1. Check if there are missing values

   There is no NULL value. So we can surely go ahead with the dataset.



```
[ ] from pyspark.sql.functions import col, isnan, when, count

    missing_counts = df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns if df.schema[c].dataType != 'date'])
    missing_counts.show()

    +-------+----+-----+-------------------+--------------+----------------+----------------+---------------+----------+-----+------+---------+
    |Product|date|Sales|Price Discount (%)|In-Store Promo|Catalogue Promo|Store End Promo|Google_Mobility|Covid_Flag|V_DAY|EASTER|CHRISTMAS|
    +-------+----+-----+-------------------+--------------+----------------+----------------+---------------+----------+-----+------+---------+
    |      0|   0|    0|                  0|             0|              0|              0|              0|         0|    0|     0|        0|
    +-------+----+-----+-------------------+--------------+----------------+----------------+---------------+----------+-----+------+---------+
```

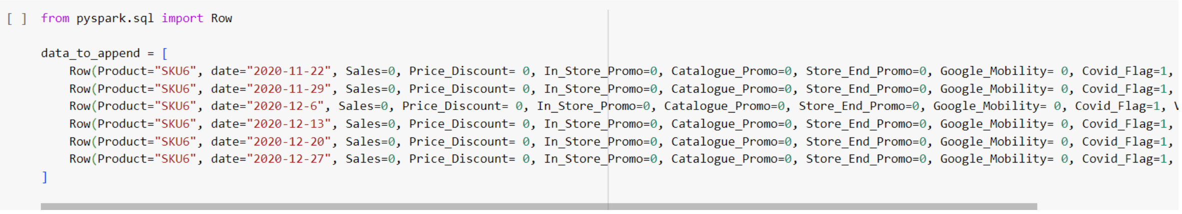
2. Validate the name of columns



```
[ ] df = (
        df
        .withColumnRenamed("Price Discount (%)", "Price_Discount")
        .withColumnRenamed("In-Store Promo", "In-Store_Promo")
        .withColumnRenamed("Catalogue Promo", "Catalogue_Promo")
        .withColumnRenamed("Store End Promo", "Store_End_Promo")
    )
```

3. Zero Sales

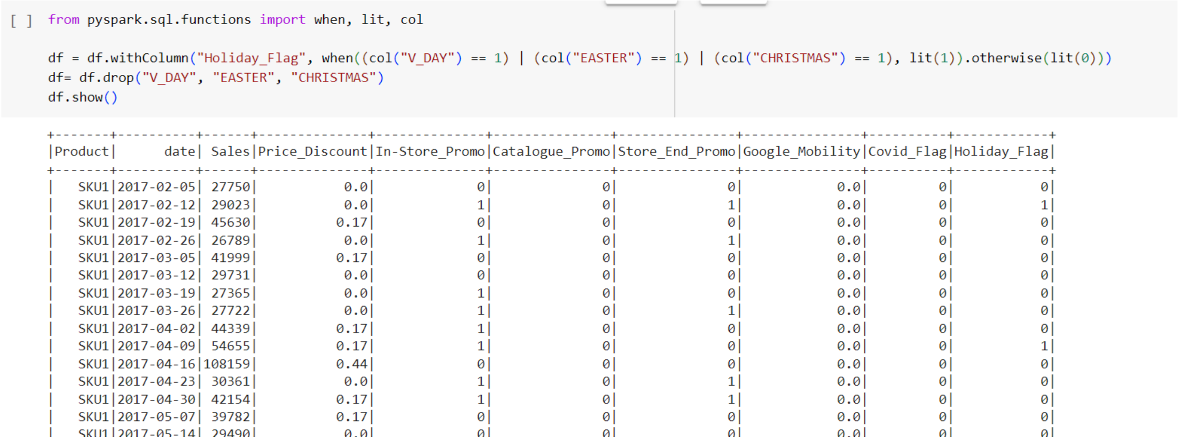
   For 0 Sales of production of SKU1 to SKU5, decided to keep them, not drop. SKU6 of missing data between 2020-11-22 to 2020-12-27, decided to create them to balance with other products.



```
[ ] from pyspark.sql import Row

    data_to_append = [
        Row(Product="SKU6", date="2020-11-22", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1,
        Row(Product="SKU6", date="2020-11-29", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1,
        Row(Product="SKU6", date="2020-12-6", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1, \
        Row(Product="SKU6", date="2020-12-13", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1,
        Row(Product="SKU6", date="2020-12-20", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1,
        Row(Product="SKU6", date="2020-12-27", Sales=0, Price_Discount= 0, In_Store_Promo=0, Catalogue_Promo=0, Store_End_Promo=0, Google_Mobility= 0, Covid_Flag=1,
    ]
```

4. Combine 3 holidays

For the 3 holidays of V_DAY, EASTER, and CHRISTMAS, we decided to combine them into 1 column because the dates of each holiday are different without duplication.

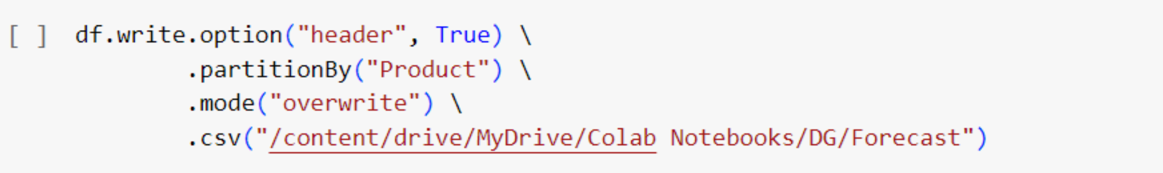


```
[ ] from pyspark.sql.functions import when, lit, col

    df = df.withColumn("Holiday_Flag", when((col("V_DAY") == 1) | (col("EASTER") == 1) | (col("CHRISTMAS") == 1), lit(1)).otherwise(lit(0)))
    df= df.drop("V_DAY", "EASTER", "CHRISTMAS")
    df.show()
```

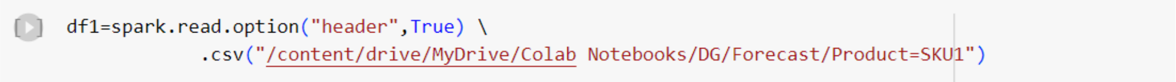
```
+-------+----------+------+--------------+--------------+---------------+---------------+---------------+----------+------------+
|Product|      date| Sales|Price_Discount|In-Store_Promo|Catalogue_Promo|Store_End_Promo|Google_Mobility|Covid_Flag|Holiday_Flag|
+-------+----------+------+--------------+--------------+---------------+---------------+---------------+----------+------------+
|   SKU1|2017-02-05| 27750|           0.0|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-02-12| 29023|           0.0|             1|              0|              1|            0.0|         0|           1|
|   SKU1|2017-02-19| 45630|          0.17|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-02-26| 26789|           0.0|             1|              0|              1|            0.0|         0|           0|
|   SKU1|2017-03-05| 41999|          0.17|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-03-12| 29731|           0.0|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-03-19| 27365|           0.0|             1|              0|              0|            0.0|         0|           0|
|   SKU1|2017-03-26| 27722|           0.0|             1|              0|              1|            0.0|         0|           0|
|   SKU1|2017-04-02| 44339|          0.17|             1|              0|              0|            0.0|         0|           0|
|   SKU1|2017-04-09| 54655|          0.17|             1|              0|              0|            0.0|         0|           1|
|   SKU1|2017-04-16|108159|          0.44|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-04-23| 30361|           0.0|             1|              0|              1|            0.0|         0|           0|
|   SKU1|2017-04-30| 42154|          0.17|             1|              0|              1|            0.0|         0|           0|
|   SKU1|2017-05-07| 39782|          0.17|             0|              0|              0|            0.0|         0|           0|
|   SKU1|2017-05-14| 29490|           0.0|             0|              0|              0|            0.0|         0|           0|
```

5. <u>Divided data into 6</u>

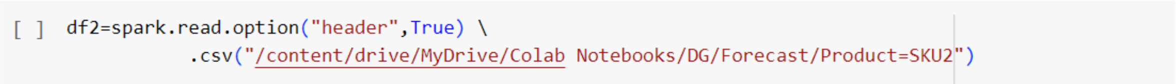
   The dataset has been effectively partitioned into **six** distinct product datasets using **PySpark**. This division allows us to focus on specific product categories individually, streamlining data preparation, exploratory analysis, feature engineering, modeling, and subsequent analysis for each product group. This approach enhances our ability to gain insights, build tailored models, and optimize our analysis for each product category while ensuring efficient and manageable data processing.



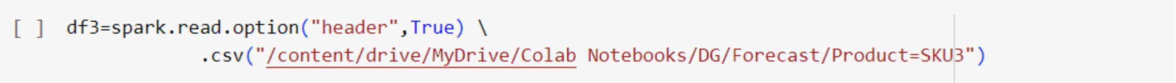
```
[ ] df.write.option("header", True) \
            .partitionBy("Product") \
            .mode("overwrite") \
            .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast")
```



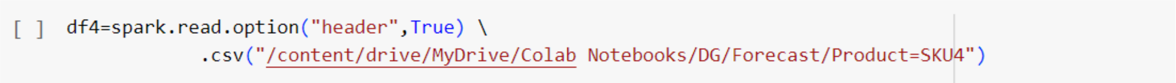
```
df1=spark.read.option("header",True) \
          .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU1")
```



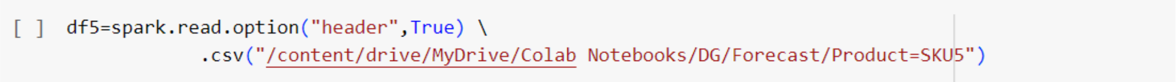
```
[ ] df2=spark.read.option("header",True) \
         .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU2")
```



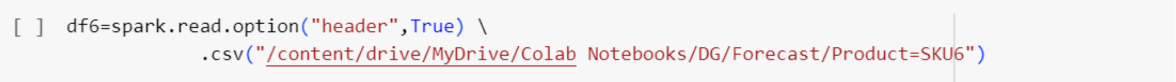
```
[ ] df3=spark.read.option("header",True) \
          .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU3")
```



```
[ ] df4=spark.read.option("header",True) \
          .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU4")
```



```
[ ] df5=spark.read.option("header",True) \
          .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU5")
```



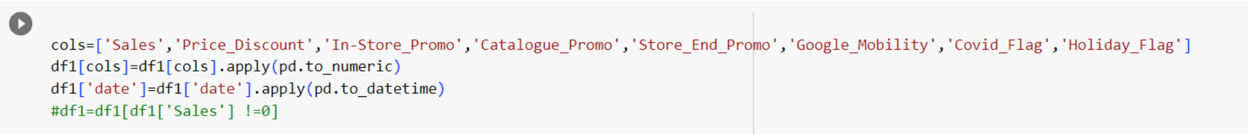
```
[ ] df6=spark.read.option("header",True) \
          .csv("/content/drive/MyDrive/Colab Notebooks/DG/Forecast/Product=SKU6")
```

6. <u>Validate the data type</u>

   We validated the data type as below.

   -Variables with numeric value to int

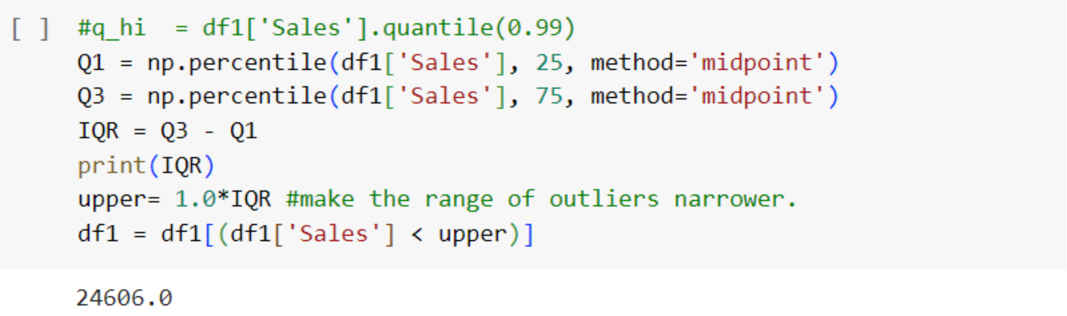
   -date to datetime



```
cols=['Sales','Price_Discount','In-Store_Promo','Catalogue_Promo','Store_End_Promo','Google_Mobility','Covid_Flag','Holiday_Flag']
df1[cols]=df1[cols].apply(pd.to_numeric)
df1['date']=df1['date'].apply(pd.to_datetime)
#df1=df1[df1['Sales'] !=0]
```

7. <u>Remove outlier with narrow range</u>

   To remove outliers using the Interquartile Range (IQR) method, calculate the IQR by finding the difference between the third quartile (Q3) and the first quartile (Q1). Then, define a lower bound (Q1 - 1.5 * IQR) and an upper bound (Q3 + 1.5 * IQR) and filter out data points that fall outside this range. This method helps identify and exclude extreme values from the dataset.



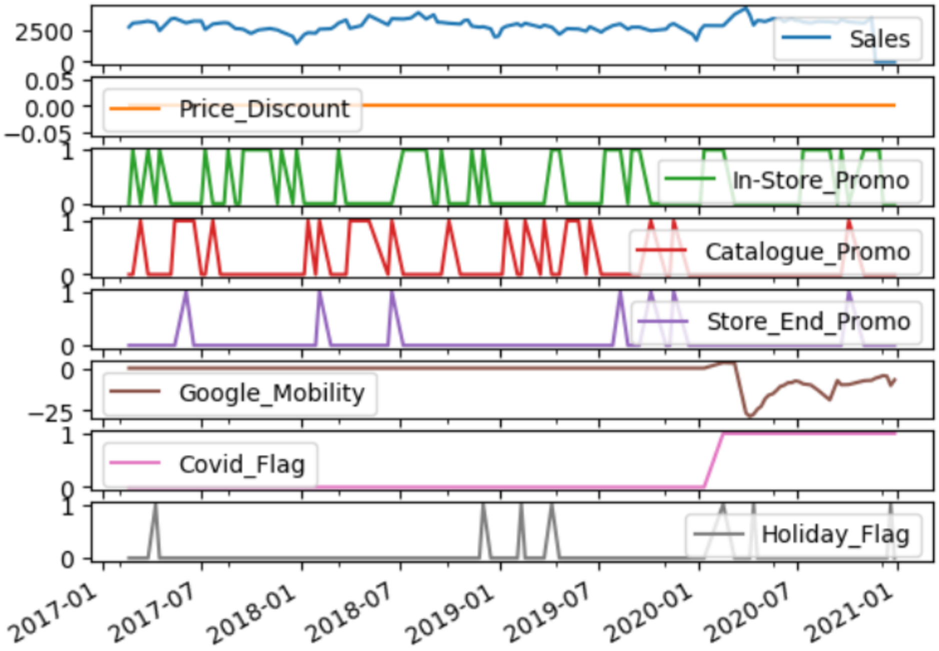
```
[ ]  #q_hi  = df1['Sales'].quantile(0.99)
     Q1 = np.percentile(df1['Sales'], 25, method='midpoint')
     Q3 = np.percentile(df1['Sales'], 75, method='midpoint')
     IQR = Q3 - Q1
     print(IQR)
     upper= 1.0*IQR #make the range of outliers narrower.
     df1 = df1[(df1['Sales'] < upper)]

     24606.0
```

Data Analysis / EDA:

1. Visualization of each product

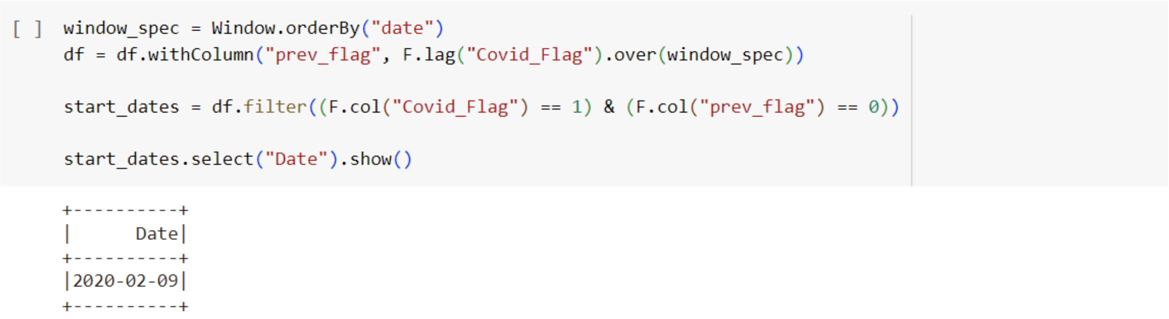
   Below is product SKU2. Each product has different sales and characteristics.

```
array([<Axes: xlabel='date'>, <Axes: xlabel='date'>,
       <Axes: xlabel='date'>, <Axes: xlabel='date'>,
       <Axes: xlabel='date'>, <Axes: xlabel='date'>,
       <Axes: xlabel='date'>, <Axes: xlabel='date'>], dtype=object)
```



2. Covid_Flag

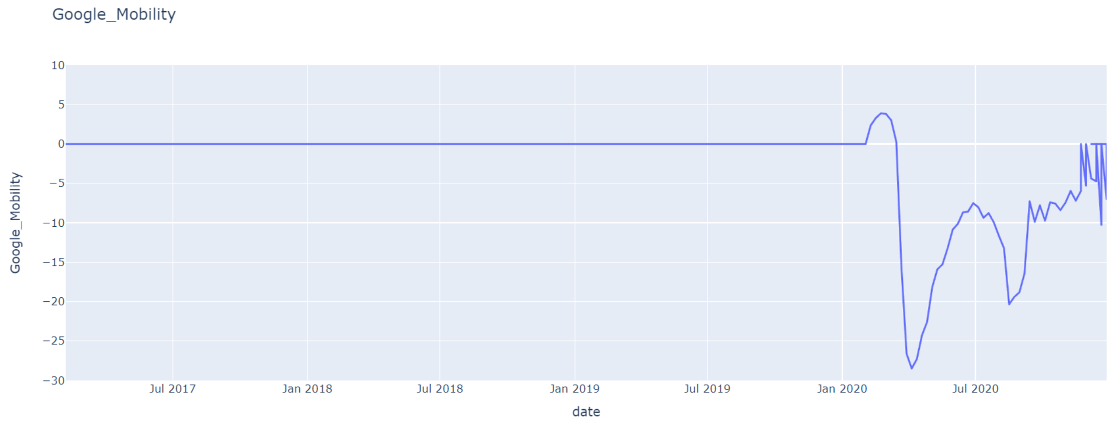
   Covid Flag started from February 09, 2020.



```
[ ]  window_spec = Window.orderBy("date")
     df = df.withColumn("prev_flag", F.lag("Covid_Flag").over(window_spec))

     start_dates = df.filter((F.col("Covid_Flag") == 1) & (F.col("prev_flag") == 0))

     start_dates.select("Date").show()
```

```
+----------+
|      Date|
+----------+
|2020-02-09|
+----------+
```

3. <u>Google Mobility</u>

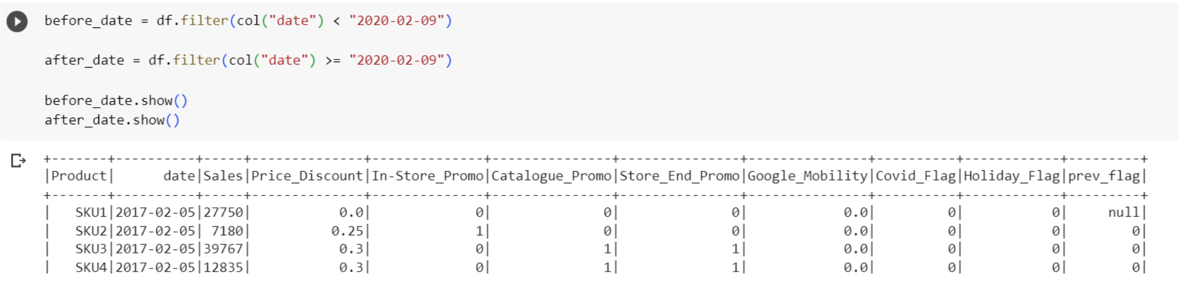


-Google Mobility is related Covid19. This is because the line is flat until February 2, 2020 above the plot. The flat line means there are no activities and no existing record.

-After February 9, 2020, it started fluctuating and keeps changing. According to the variable of Covid Flag, it started being recorded as 1 after February 9, 2020. The timing between Google Mobility and Covid Flag is exactly coinciding.

-Google Mobility data tracks travel patterns in detail, such as how often people go to public places and how much time they spend commuting or shopping. This will allow us to assess the risk of spread of infection and predict the spread of infection in a particular region or city.

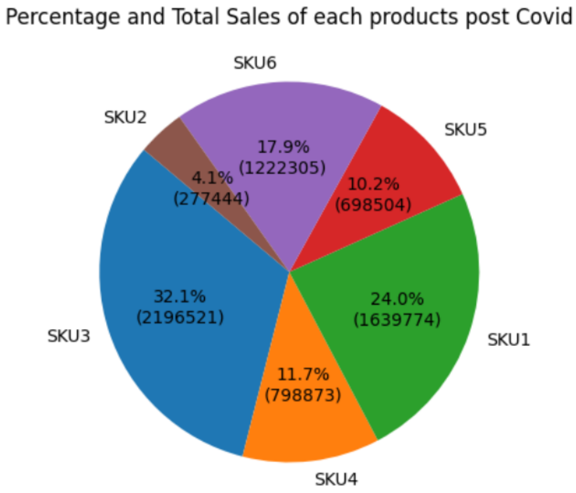
4. <u>To analyze the data pre-Covid and post-Covid, we divide the data to 2</u>

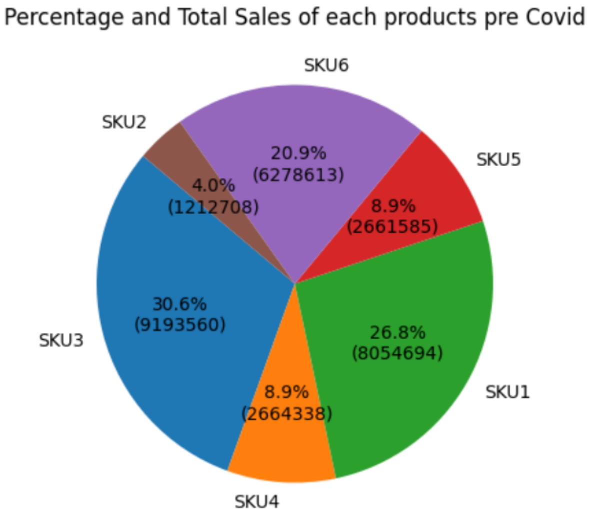


```
before_date = df.filter(col("date") < "2020-02-09")

after_date = df.filter(col("date") >= "2020-02-09")

before_date.show()
after_date.show()
```

```
+-------+----------+-----+--------------+--------------+---------------+---------------+---------------+----------+------------+---------+
|Product|      date|Sales|Price_Discount|In-Store_Promo|Catalogue_Promo|Store_End_Promo|Google_Mobility|Covid_Flag|Holiday_Flag|prev_flag|
+-------+----------+-----+--------------+--------------+---------------+---------------+---------------+----------+------------+---------+
|   SKU1|2017-02-05|27750|           0.0|             0|              0|              0|            0.0|         0|           0|     null|
|   SKU2|2017-02-05| 7180|          0.25|             1|              0|              0|            0.0|         0|           0|        0|
|   SKU3|2017-02-05|39767|           0.3|             0|              1|              1|            0.0|         0|           0|        0|
|   SKU4|2017-02-05|12835|           0.3|             0|              1|              1|            0.0|         0|           0|        0|
```

Comparison of percentage of sharing each product before Covid and after Covid.


Percentage and Total Sales of each products post Covid

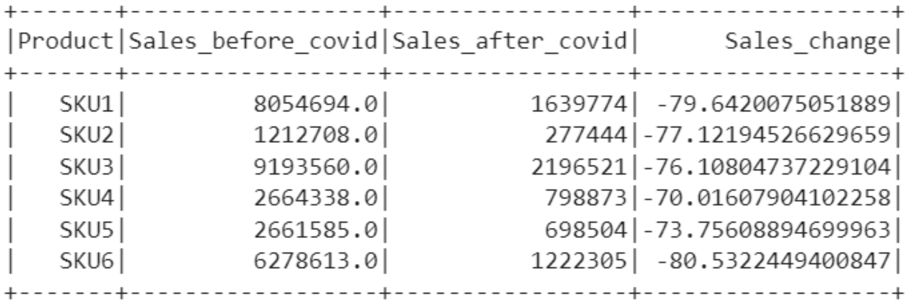

Percentage and Total Sales of each products pre Covid

In terms of the percentages of the products to sales, there is no very big difference between pre covid and post covid.

In general both before Covid and after Covid, SKU3 is the most popular product. SKU1 is the secondest popular. And SKU6 is the 3rd.

SKU4 and SKU5 have the same sales amounts. SKU2 is the least popular product.

-Comparing the sales amount before Covid and after Covid



```
+-------+------------------+-----------------+------------------+
|Product|Sales_before_covid|Sales_after_covid|      Sales_change|
+-------+------------------+-----------------+------------------+
|   SKU1|         8054694.0|          1639774| -79.6420075051889|
|   SKU2|         1212708.0|           277444|-77.12194526629659|
|   SKU3|         9193560.0|          2196521|-76.10804737229104|
|   SKU4|         2664338.0|           798873|-70.01607904102258|
|   SKU5|         2661585.0|           698504|-73.75608894699963|
|   SKU6|         6278613.0|          1222305| -80.5322449400847|
+-------+------------------+-----------------+------------------+
```

Sales of all of the products significantly reduced between 70% and 80% minus after Covid compared to before Covid.

Recommended Models:

We use multivariate time series for this dataset.

Proposal methods are below.

-ARIMA

-Prophet

-XGBoost

-LSTM

-SVM

-VAR

What are the problems in the data (number of NA values, outliers , skewed etc)?

Data Assessment Summary:

1. Zero Sales Observation:

- Based on our research, we have observed instances where sales data contains zero values for each product.
- We will investigate the context of these zero sales observations, as they could be legitimate data points, or they may require special handling.
- Understanding the reasons behind zero sales can help refine our analysis.

2. Outliers:

- Outliers have been identified in the data.
- We recognize that the presence of outliers can distort our analysis and results.

Proposed Approach:

1. Partitioning by Product:

- To gain a more granular understanding of the data and to address the unique sales characteristics of each product, we plan to partition the dataset by product category.
- This partitioning will enable us to perform data quality checks and analyses specific to each product, which can yield more meaningful insights.

2. Handling Missing Values:

- After partitioning the data by product, we will examine each product's dataset for missing values.
- Our goal is to implement data imputation techniques or strategies that are tailored to each product's sales behavior, thus mitigating the impact of missing data.

3. Outlier Treatment:

- For each product category, we will assess and address outliers individually.
- Techniques such as outlier removal, transformation, or the use of robust statistical methods will be employed to manage outliers effectively.
- Our aim is to ensure that our analysis and modeling are not unduly influenced by extreme data points.

4. Zero Sales Observation:

- Based on our research, we have observed instances where sales data contains zero values for each product.
- We will investigate the context of these zero sales observations, as they could be legitimate data points, or they may require special handling.

By adopting this systematic approach of partitioning the data by product, addressing missing values, managing outliers, and examining zero sales data, we aim to enhance the quality and relevance of our analysis, ultimately leading to more accurate and actionable insights.

Data Preprocessing:

1. Zero Values Removed:
    - Zero sales values have been successfully removed from the dataset.
    - This step helps ensure that our analysis focuses on meaningful sales data points.
2. Outliers Removed:
    - Outliers have been identified and removed from the dataset.
    - The removal of outliers aids in creating a more robust and representative dataset for analysis.

By eliminating zero values and outliers, we are now working with a cleaner dataset that is better suited for our analytical goals.

What approaches are you trying to apply on your data set to overcome problems like NA value, outlier etc and why?

1. Partitioning by Product:

- To address the variability in sales for different products, we partitioned the dataset by product category.
- This partitioning allows for tailored analysis and treatment of issues specific to each product.

2. Handling Missing Values:

- Within each product category, we removed rows with missing sales values.
- This step ensures that our analysis focuses on complete and relevant sales data for each product.

3. Outlier Detection and Removal:

- Box plots were utilized to visualize the sales distribution for each product.
- The Interquartile Range (IQR) formula was applied to identify and remove outliers specific to each product category.
- Managing outliers ensures that our analysis and modeling are not unduly influenced by extreme data points.

4. Data Cleanliness Achieved:

- Following these steps, our dataset is now free of missing sales values and outliers.
- This cleanliness enhances the dataset's suitability for further analysis and modeling.

Next Steps:

- With clean data in hand, we can now explore further transformations and feature engineering to prepare the data for modeling.
- Additional steps might include normalization, encoding categorical variables, or creating new features to improve predictive performance.