Python Exploratory Data Analysis of a Grocery Dataset

This is the third article in a series for beginning data analysis. In the last article I went over using basic and intermediate SQL to analyze our grocery dataset.

For this project I used Anaconda Navigator because it comes with **Jupyter Notebook , Pycharm , R Studio .**

First we want to import our dataset into Jupyter Notebook. I left all the fields as they originally were in order to show you some things you can do to edit your dataset. I used CSV file because pandas in python works only for csv when we are importing the file.

```
import pandas as pd

print('pd')
```

```
---To read the file from local folder

df = pd.read_csv(r"C:\Users\dilee\OneDrive\Documents\Groceries_

---After getting the file please check with head and tail

-----To get the top rows
df.head()

----To get the bottom rows
df.tail()

---- To get the rows and columns
df.info()


----- To add any columns
df['Column name'] = 'usa'


----- To delete any columns

del df['column name']
```

First we need to verify the data

The Dataset and Data Cleaning

Check For Null Values

Check for Null values in Column 1. This did not return anything which is of course what we want. Let’s do the same for the other 2 columns. No results again. We’re golden.

```
----to check null values

null_members = df[df['member_number'].isnull()]
```

Data analysts often come across data stored as text. Especially dates stored as text. Converting this data to dates is important because date data can be very valuable during analysis.

from the data we have to get the exploratory data analysis we need to create some columns from the data we have. i created day, month, year, dow and season column just from date column by using below code

Below we are extracting different column names from Date Column :

```
---to convert date columns to day, month and year column we n
eed to do datetime format
df['Date'] = pd.to_datetime(df['Date'])


-----then we get below columns
df['Day'] = df['Date'].dt.day

df['Month'] = df['Date'].dt.day

df['Year'] = df['Date'].dt.day




----from the day and month column we get the season column us
ing below condition

# Function to map dates to seasons
def get_season(date):
```

```
    month = date.month
    day = date.day

    # Ensure proper indentation for each conditional block
    if (month == 3 and day >= 20) or (month in [4, 5]) or (mo
nth == 6 and day < 21):
        return 'Spring'  # This line needs to be indented
    elif (month == 6 and day >= 21) or (month in [7, 8]) or
(month == 9 and day < 23):
        return 'Summer'  # Indented as well
    elif (month == 9 and day >= 23) or (month in [10, 11]) or
(month == 12 and day < 21):
        return 'Autumn'  # Indented
    else:
        return 'Winter'  # Indented



------to get the weekday names
df['day_name'] = df['Date'].dt.day_name()
```

Most Popular Products by month

```
----max products based on month

month_sales = df.groupby('month')['itemDescription'].count().
reset_index()
max_month = month_sales.sort_values(by='itemDescription',asce
nding=False).iloc[0]
print(f"The month with the maximum items sold is {max_month
['month']}
with {max_month['itemDescription']} items sold.")
```

```
The month with the maximum items sold is 8 with 3496 items so
ld.
```

Most Popular products by Day

```
-----max products based on day

Day_sales = df.groupby('Day')['itemDescription'].count().rese
t_index()
max_Day = Day_sales.sort_values(by='itemDescription',ascendin
g=False).iloc[0]
print(f"The Day with the maximum items sold is {max_Day['Da
y']}
with {max_Day['itemDescription']} items sold.")

The Day with the maximum items sold is 3 with 1393 items sol
d.
```

Most Popular products by year 2014

```
df_2014 = df[df['year'] == 2014]
item_counts = df_2014['itemDescription'].value_counts().reset
_index()
item_counts.columns = ['itemDescription', 'ITEM_COUNT']
top_item = item_counts.nlargest(1, 'ITEM_COUNT')
print(f"The item with the highest sales in 2014
is {top_item.iloc[0]['itemDescription']} with {top_item.iloc
[0]['ITEM_COUNT']} sales.")

The item with the highest sales in 2014 is whole milk with 10
38 sales.
```

Most Popular products by year 2015

```
df_2015 = df[df['year'] == 2015]
item_counts = df_2015['itemDescription'].value_counts().reset
```

```
_index()
item_counts.columns = ['itemDescription', 'ITEM_COUNT']
top_item = item_counts.nlargest(1, 'ITEM_COUNT')
print(f"The item with the highest sales in 2015
is {top_item.iloc[0]['itemDescription']} with {top_item.iloc
[0]['ITEM_COUNT']} sales.")

The item with the highest sales in 2014 is whole milk with 14
64 sales.
```

Most Popular Products bought by Member_number

```
Member_number_sales = df.groupby('Member_number')['itemDescri
ption'].count().reset_index()
Member_number_sales = Member_number_sales.sort_values(by='ite
mDescription', ascending=False)
top_Member = Member_number_sales.iloc[0]
print(f"The member_number with the maximum items bought
is {top_Member['Member_number']} with {top_Member['itemDescri
ption']} items.")

The member_number with the maximum items bought is 3180 with
36 items
```

Most Popular Products by Season

```
import pandas as pd

# Group by 'purchase season' and count the occurrences of 'it
emdescription'
season_sales = df.groupby('season')['itemDescription'].count
().reset_index()

# Sort by 'item_count' (the count of items sold in each seaso
n) in descending order
```

```
season_sales = season_sales.sort_values(by='itemDescription',
ascending=False)

# Select the top row (the season with the most items sold)
top_season = season_sales.iloc[0]

# Print the result
print(f"The season with the maximum items sold is
{top_season['season']} with {top_season['itemDescription']} i
tems sold.")

The season with the maximum items sold is Summer with 10109 i
tems sold.
```

Most Popular Products by DOW

```


# Assuming df is your DataFrame

# Group by 'purchase dow' and count the occurrences of 'itemd
escription'
dow_sales = df.groupby('day_name')['itemDescription'].count
().reset_index()

# Sort by 'item_count' (the count of items sold on each day o
f the week) in descending order
dow_sales = dow_sales.sort_values(by='itemDescription', ascen
ding=False)

# Select the top row (the day of the week with the most items
sold)
top_dow = dow_sales.iloc[0]

# Print the result
print(f"The day of the week with the maximum items sold is
{top_dow['day_name']} with {top_dow['itemDescription']} items
```

```
sold.")


The day of the week with the maximum items sold is Thursday w
ith 5620 items sold.
```