7NEXT DATA ANALYSIS CHALLENGE

Warmup

Understanding each columns

1. Firstly, explore the data by looking at the provided columns.

a. What do you understand about each column from a quick scan of column names and types

From a quick scan of the columns, I can see that the dataset is composed of continuous and categorical variables. The continuous variables having data type int64, and float64, while categorical variables have data type object. Also, the quick scan of the data show that the price column is the ratio of value to the Qty.

GRP is the products specification present on each store.

The Bill_id column have some duplicated since it's generated base on store number it can have multiple similar id's base on the store, but it's a unique to a specific store.

The month column have three moths worth of sale from represent by M1, M2, and M3.

The storecode column present the three stores N1….N10

The day column represent each of the month the transaction take place from the 1st till the 31st of the month.

The bill Amount column represent the amount of each transaction at the specific store.

b. Columns of interest – What is the relationship between columns "BILL_AMT", "QTY", "VALUE" and "PRICE" in the given data?

"BILL_AMT", "QTY", "VALUE" and "PRICE", are all continuous variables columns.

Bill_AMT is nothing but the overall transaction at the given store, while PRICE is the ratio of VALUE to the Qty, and QTY is the overall quantity of GRP, and VALUE is the product of PRICE multiply by QTY.

The dataset is loaded into the Python environment.

```python
#loading and check data data
store_dataset=pd.read_csv("store_dataset.csv")
```

```python
print(type(store_dataset))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```python
store_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26985 entries, 0 to 26984
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   MONTH      26985 non-null  object 
 1   STORECODE  26985 non-null  object 
 2   DAY        26985 non-null  int64  
 3   BILL_ID    26985 non-null  object 
 4   BILL_AMT   26985 non-null  float64
 5   QTY        26985 non-null  float64
 6   VALUE      26985 non-null  float64
 7   PRICE      26985 non-null  float64
 8   GRP        26985 non-null  object 
dtypes: float64(4), int64(1), object(4)
memory usage: 1.9+ MB
```

The datset has multiple columns such as Month, STORECODE, DAY, BILL_ID, BILL_AMT and so on. The attributes belong to numerical as well as categorical data type such as object as well as float. The dataset has 26985 samples with 9 attributes.

```python
store_dataset.head()
```

	MONTH	STORECODE	DAY	BILL_ID	BILL_AMT	QTY	VALUE	PRICE	GRP
0	M1	N1	4	T375	225.0	1.0	225.0	225.0	BUTTER MARGR (4/94)
1	M1	N1	4	T379	95.0	1.0	95.0	95.0	CONFECTIONERY - ECLAIRS
2	M1	N1	4	T381	10.0	1.0	10.0	10.0	CHOCOLATE
3	M1	N1	4	T382	108.0	1.0	108.0	108.0	PACKAGED TEA
4	M1	N1	4	T384	19.0	1.0	19.0	19.0	ALL IODISED SALT

The dataset has 10 stores.

```python
# Print the total number of null values in the data
print(f"Missing values :  {store_dataset.isnull().sum().values.sum()}")
```

```
Missing values :  0
```

The dataset does not have any missing values.

```
] # For each column, print the number of unique values
  print(f"Unique values :  {store_dataset.nunique()}")
```

```
Unique values :  MONTH           3
STORECODE      10
DAY            31
BILL_ID      6424
BILL_AMT     1453
QTY            46
VALUE         640
PRICE         492
GRP            80
dtype: int64
```

The dataset has 10 stores. It involves 31 days and aslo 80 category of products.

```
#Descriptive statistics for continuous variables
store_dataset.describe().T
```

	count	mean	std	min	25%	50%	75%	max
DAY	26985.0	15.167019	8.956057	1.0	7.0	14.0	23.0	31.0
BILL_AMT	26985.0	278.754206	541.398504	0.0	40.0	111.0	280.0	7292.0
QTY	26985.0	4.104984	95.666949	0.0	1.0	1.0	2.0	12000.0
VALUE	26985.0	67.808551	118.005978	0.0	10.0	30.0	80.0	3150.0
PRICE	26985.0	52.812982	84.987730	0.0	10.0	22.0	64.0	3150.0

The descriptive statistics illustrates the count, mean, standard deviation, min, max, 25%, 50% as well as 75% values.

Column of interest

```
#correlation
corr = store_dataset.corr()
print(type(corr))
print(corr)
```

```
<class 'pandas.core.frame.DataFrame'>
               DAY  BILL_AMT       QTY     VALUE     PRICE
DAY       1.000000 -0.048808  0.008432 -0.027509 -0.021367
BILL_AMT -0.048808  1.000000  0.027484  0.460631  0.350307
QTY       0.008432  0.027484  1.000000  0.067245 -0.018326
VALUE    -0.027509  0.460631  0.067245  1.000000  0.791834
PRICE    -0.021367  0.350307 -0.018326  0.791834  1.000000
```

The correlation analysis illustrates the colleration among attributes. The attributes “value” and “Day” is having negative correlation. Bill_AMT and Value are correlated positively. Value and PRICE is higly correlated with each other.

Technical Analysis

Sales by store

Total sales amount made by each store

```
store_new1=store_new[['STORECODE', 'VALUE']]
```

```
store_new1=store_new1.groupby('STORECODE').sum('VALUE')
```

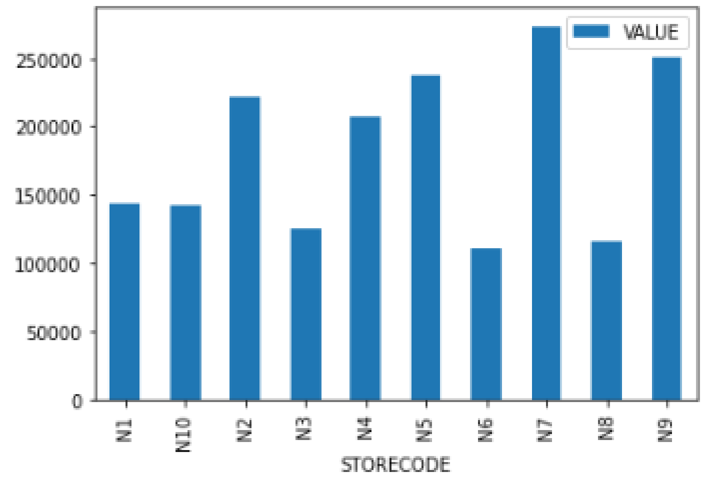
```
print(store_new1)
```

```
               VALUE
STORECODE
N1         144206.93
N10        142433.00
N2         221355.00
N3         125528.79
N4         206874.99
N5         238057.00
N6         110187.00
N7         273787.15
N8         116466.86
N9         250917.03
```

Total sales amount made by each store is illustrated above.

```
store_new1.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fe1cb5e90>
```



The plot illustrates that, total sales amount is highest for the store “N7.

Sales by category

Store N1

```
store_new1=store_new[store_new["STORECODE"]=='N1']
```

```
store_new2=store_new1[['GRP', 'QTY']]
```

```
store_new2=store_new2.groupby('GRP').sum('QTY')
```

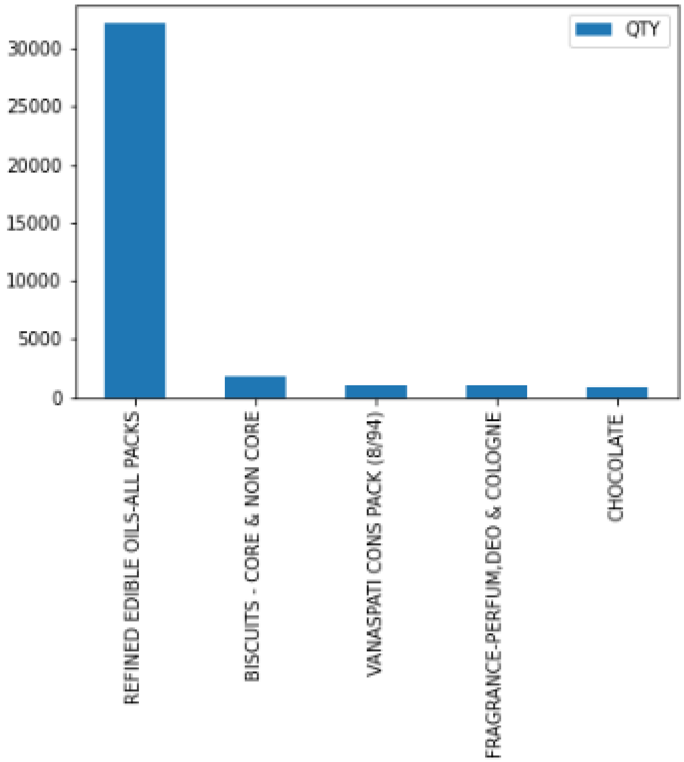
```
print(store_new2)
```

```
                               QTY
GRP
AGARBATTI & DHOOPBATTI        15.0
ALL AIR FRESHNERS(01/03)       3.0
ALL IODISED SALT              56.0
ANTACIDS                      24.0
ANTISEPTIC LIQUIDS (4/97)      1.0
...                            ...
TOOTH PASTES                  51.0
TWIN BLADES                    5.0
VANASPATI CONS PACK (8/94)  1019.0
VERMICELLI & NOODLE          518.0
WASHING POWDERS/LIQUIDS      209.0

[68 rows x 1 columns]
```

```
df.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf9b4e50>
```

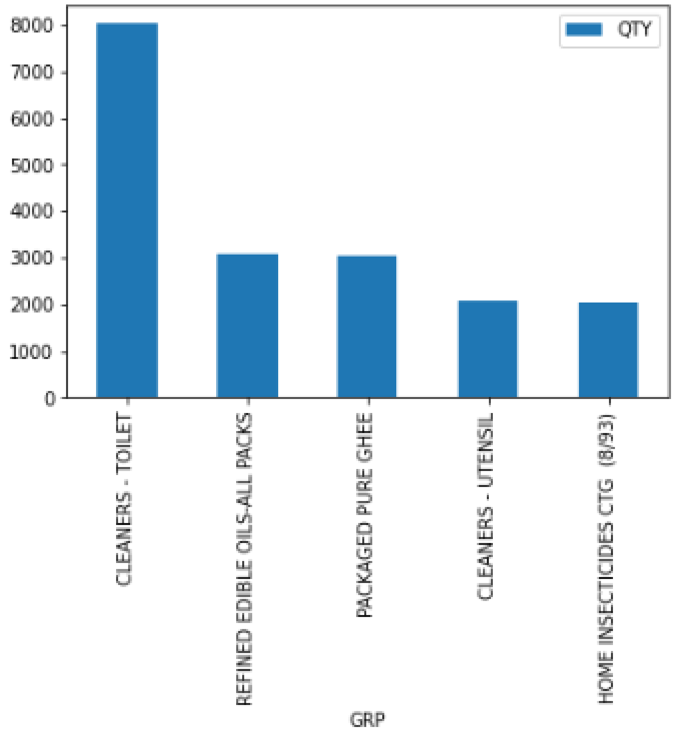


For N1, refined edible oils all packs is sold high in terms of quantity.

Store N2:

```
store_N2=store_new[store_new["STORECODE"]=='N2']
store_N2=store_N2[['GRP', 'QTY']]
store_N2=store_N2.groupby('GRP').sum('QTY')
print(store_N2)
store_N2.sort_values(by=['QTY'],inplace=True,ascending=False)
print(store_N2.head())
store_N2.sort_values(by=['QTY'],inplace=True,ascending=False)
df_N2=store_N2.head()
df_N2.plot.bar()
```

```
HOME INSECTICIDES CTG  (8/93)  2023.0
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fe1535090>
```

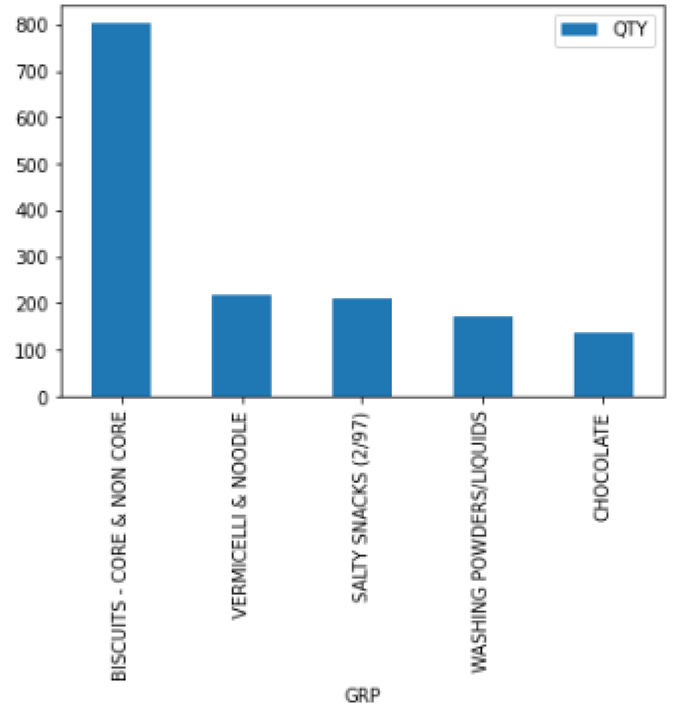


In store N2, the product “cleaners_Toilet” is sold high.

Store N3

```
store_N3=store_new[store_new["STORECODE"]=='N3']
store_N3=store_N3[['GRP', 'QTY']]
store_N3=store_N3.groupby('GRP').sum('QTY')
print(store_N3)
store_N3.sort_values(by=['QTY'],inplace=True,ascending=False)
print(store_N3.head())
store_N3.sort_values(by=['QTY'],inplace=True,ascending=False)
df_N3=store_N3.head()
df_N3.plot.bar()
```

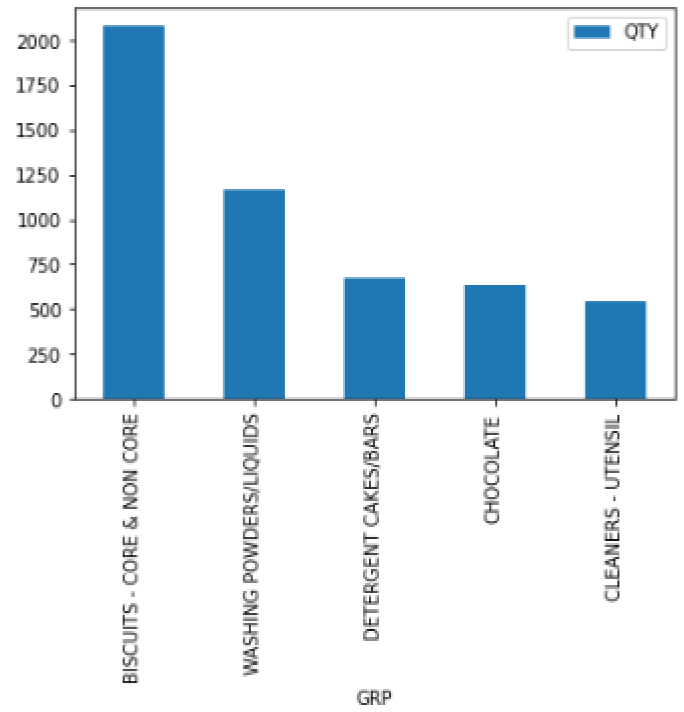
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf76a8d0>
```



In store N3, BISCUITS-CORE & NON CORE is sold.

Store N4

```
CLEANERS - UTENSIL           548.0
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf6f3ed0>
```

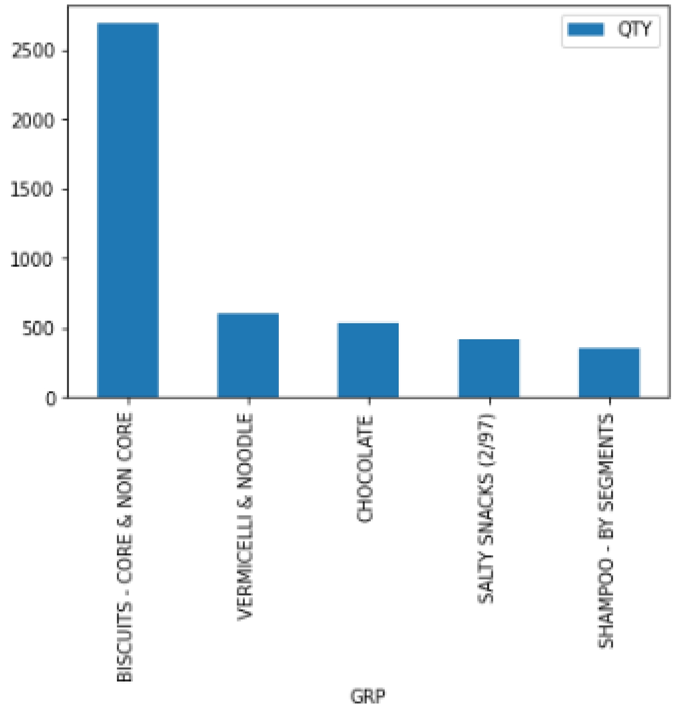


In store N4 also, the product such as Biscuit – Core & Non Core is sold high in terms of quantity.

Store : N5

```
store_N5=store_new[store_new["STORECODE"]=='N5']
store_N5=store_N5[['GRP', 'QTY']]
store_N5=store_N5.groupby('GRP').sum('QTY')
print(store_N5)
store_N5.sort_values(by=['QTY'],inplace=True,ascending=False)
print(store_N5.head())
store_N5.sort_values(by=['QTY'],inplace=True,ascending=False)
df_N5=store_N5.head()
df_N5.plot.bar()
```

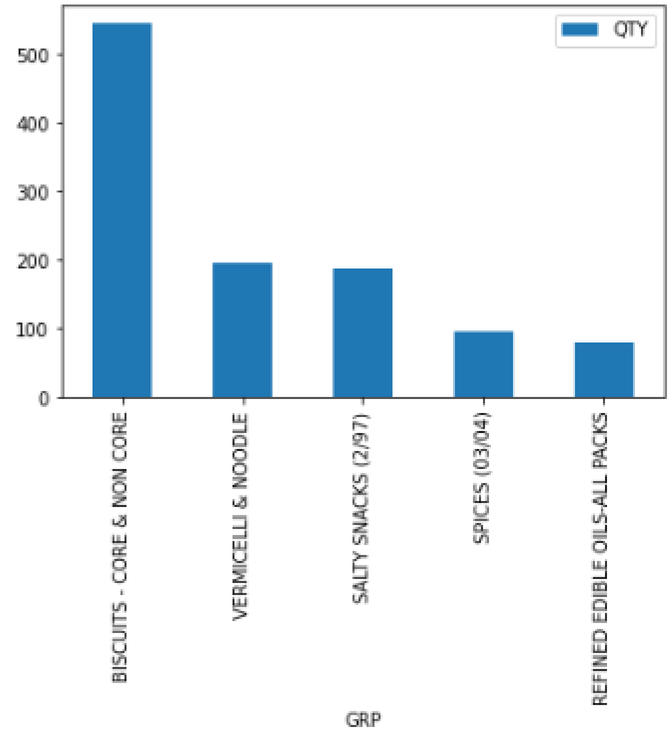
```
SHAMPOO - BY SEGMENTS        352.0
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf735e10>
```



In store N5, Biscuits-core & non-core is sold high.

Store : N6

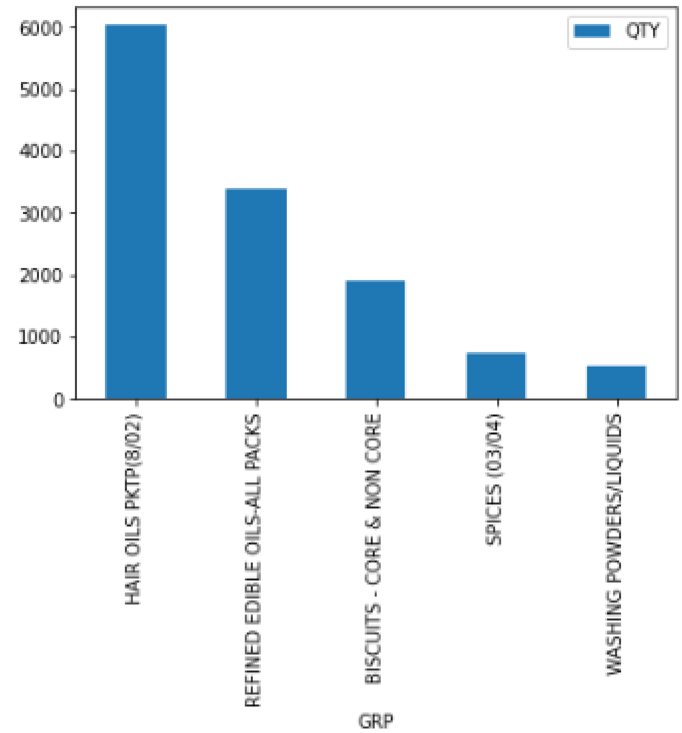
```
REFINED EDIBLE OILS-ALL PACKS    79.0
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf5cf490>
```



In store N6, the product “Biscuit – core & Non-core” is sold high.

Store : N7

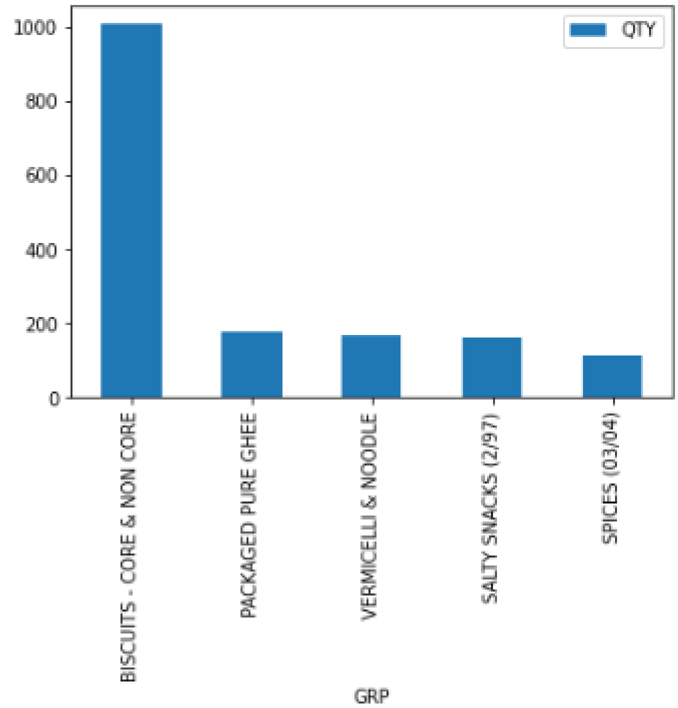
```
WASHING POWDERS/LIQUIDS          529.0
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf55a290>
```



In store N7, Hair oils are sold high.

Store N8

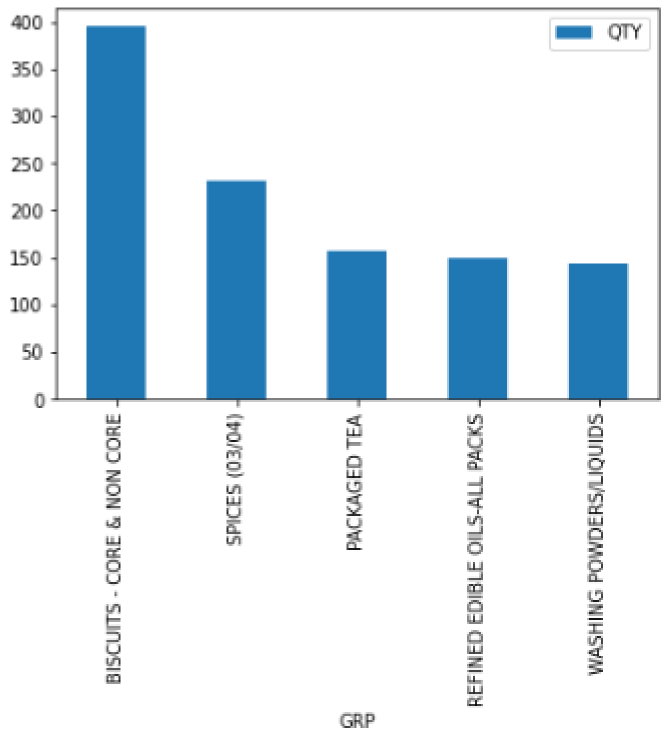
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf4d1490>
```



In store N8, Biscuits – core & Non core is sold high

Store N9

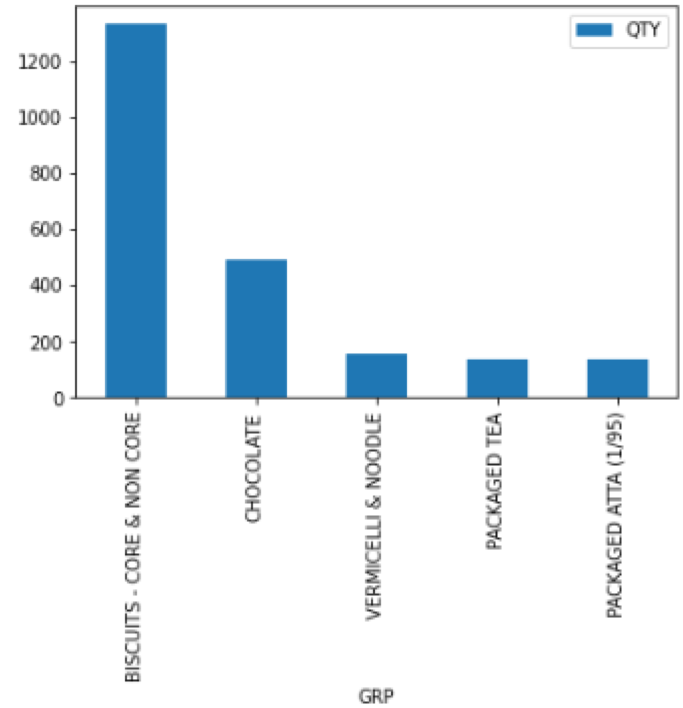
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf453810>
```



In store N9, Biscuits –core & Non core is sold high

Store N10

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf432a90>
```

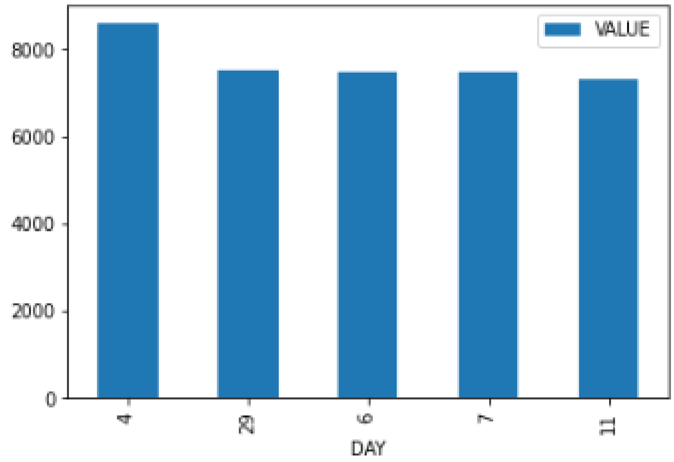


In store N10 also Biscuits –core & Non core is sold high

Sales by day

```
store_N1_day=store_new[store_new["STORECODE"]=='N1']
store_N1_day=store_N1_day[['DAY', 'VALUE']]
store_N1_day=store_N1_day.groupby('DAY').sum('VALUE')
print(store_N1_day)
store_N1_day.sort_values(by=['VALUE'],inplace=True,ascending=False)
print(store_N1_day.head())
store_N1_day.sort_values(by=['VALUE'],inplace=True,ascending=False)
df_N1_day=store_N1_day.head()
df_N1_day.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf80efd0>
```

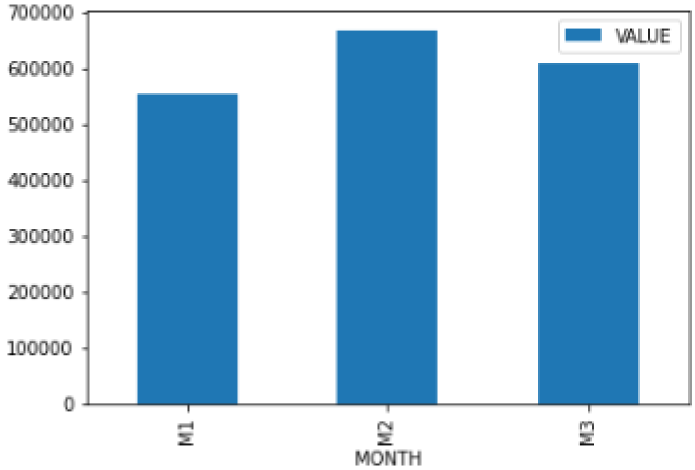


The above analysis illustrates the best day for the store N1 in terms of sales of amount. According to this, the best day for the store N1 in terms of sales of amount is 4.

Sales by month

```
store_month=store_new[['MONTH', 'VALUE']]
store_month=store_month.groupby('MONTH').sum('VALUE')
print(store_month)
store_month.plot.bar()
```

```
            VALUE
MONTH
M1      552545.32
M2      667850.67
M3      609417.76
<matplotlib.axes._subplots.AxesSubplot at 0x7f0fdf7f6c90>
```



The analysis is about calculating the total sales amount across all stores per month for finding the best month across the given three months. The best month based on total sales amount across all store is M2.

References

https://www.w3schools.com/python/

https://www.tutorialspoint.com/python-descending-order-sort-grouped-pandas-dataframe-by-group-size