Supermarket Sales Project

NTI Capstone Project

Team Members :

- **Abdelrahman Elshimy**
- **Tasneem Abdelshafy**
- **Manar Hamdy**

Summary:

- The dataset represents 1,000 transactions across three supermarket branches, and analyzes key metrics such as sales, customer demographics, product categories and payment methods.

Approach:

- The project approach is to understand the data provided after cleaning it, and to answer some questions using some cards and charts for explaining more information about this data to provide a comprehensive view of supermarket performance across multiple dimensions from which we can make appropriate recommendations for the future.

Quality issues exist in the data:

- The "Tax" column has 9 missing values (can be calculated).
- The "Total" column has 3 missing values (can be calculated).
- There are 6 duplicated rows.
- The "Customer Type" field contains "-" instead of NaN.
- The "Customer Type" field has missing values.
- The "Customer Type" field contains the values ['Member', 'Memberr'], which have the same meaning.
- The "Unit Price" column contains the value "USD" causing the data type of the column to be object instead of float.
- The "Quality" column contains negative values.
- The "Time" column contains the value "8 – 30 pm" instead of "20:30"
- The "Rating" column contains a value of "97.0" instead of "9.7"

Tidiness issues exist in the data:

- The cities "Yangon," "Naypyidaw," and "Mandalay" are each represented in separate columns, but they should be combined into a single column called "City," with each row containing one of the three city names.

Quality issue NO. 1: The "Tax" column has 9 missing values.

Solution: Missing values in the "Tax" column can be calculated from the equation (Unit Price * Quantity) * 0.05

Code:

```
### Create calc_tax Function
def calc_tax(row):
    return np.round((row['Unit price'] * row['Quantity']) * 0.05, 4)
```

```
### Apply calc_tax Funciton
df['Tax 5%'] = df.apply(calc_tax, axis=1)
```

Test:

```
df['Tax 5%'].isnull().sum()
```

```
0
```

Quality issue NO. 2: The "Total" column has 3 missing values.

Solution: Missing values in the "Total" column can be calculated from the equation (Unit Price * Quantity) + Tax

Code:

```
### Create calc_total Function
def calc_total(row):
    return np.round((row['Unit price'] * row['Quantity']) + row['Tax 5%'], 4)
```

```
### Apply calc_total Function
df['Total'] = df.apply(calc_total, axis=1)
```

Test:

```
1 df['Total'].isnull().sum()
```

```
0
```

Quality issue NO. 3: There are 6 duplicated rows

Solution: Remove duplicates

Code:

```
1 ## Drop Duplicates
2 df.drop_duplicates(inplace=True)
3
4 ## Reset Index
5 df.reset_index(drop=True, inplace=True)
```

Test:

```
1 df.duplicated().sum()
```

```
0
```

Quality issue NO. 4 and 5: The "Customer Type" field contains "-" instead of NaN and contains the values ['Member', 'Memberr'], which have the same meaning

Solution: replace "-" with NaN and replace 'Member', 'Memberr'] with value 'Member'

Code:

```
### Create fix_customerType Function
def fix_customerType(v):
    if v == '-':
        return np.nan
    elif v in ['Member', 'Memberr']:
        return 'Member'
    else:
        return v
```

```
### Apply fix_customerType Function
df['Customer type'] = df['Customer type'].apply(fix_customerType)
```

Test:

```
df['Customer type'].unique()
```

```
array(['Normal', nan, 'Member'], dtype=object)
```

Quality issue NO. 6: The "Customer Type" field has missing values.

Solution: Impute missing values with Mode value

Code:

```
### Impute Missing Values With Mode Value
df['Customer type'].fillna(df['Customer type'].mode()[0], inplace=True)
```

Test:

```
df['Customer type'].isnull().sum()
```

```
0
```

Quality issue NO. 7: The "Unit Price" column contains the value "USD" causing the data type of the column to be object instead of float.

Solution: Create function to remove the USD value and convert all values to float

Code:

```
### Create fix_unitPrice Function
def fix_untiPrice(v):
    if "USD" in v:
        return float(v[:-4])
    else:
        return float(v)
```

```
### Apply fix_unitPrice Function
df['Unit price'] = df['Unit price'].apply(fix_untiPrice)
```

Test:

```
df['Unit price'].dtype
```

```
dtype('float64')
```

Quality issue NO. 8: The "Quality" column contains negative values.

Solution: Take Absolute value of all values in Quantity Column

Code:

```
df['Quantity'] = df['Quantity'].apply(lambda x:abs(x))
```

Test:

```
1 df['Quantity'].min()
```

```
1
```

Quality issue NO. 9: The "Time" column contains the value "8 – 30 pm" instead of "20:30"

Solution: Create function to replace 8-30 pm with 20:30

Code:

```
### Create fix_time Function
def fix_time(v):
    if v == '8 - 30 PM':
        return '20:30'
    else:
        return v
```

```
### Apply fix_time Function
df['Time'] = df['Time'].apply(fix_time)
```

Test:

```
1 df['Time'].unique()[3]
```

```
'20:30'
```

Quality issue NO. 10: The "Rating" column contains a value of "97.0" instead of "9.7"

Solution: Create Function to replace "97.0" with "9.7"

Code:

```
### Create fix_rating Function
def fix_rating(v):
    if v == 97.0:
        return 9.7
    else:
        return v
```

```
### Apply fix_rating Function
df['Rating'] = df['Rating'].apply(fix_rating)
```

Test:

```
df['Rating'].max()
```

```
10.0
```

Tidiness issue NO. 1: The cities "Yangon," "Naypyidaw," and "Mandalay" are each represented in separate columns, but they should be combined into a single column called "City," with each row containing one of the three city names.

Solution: Create a column named City that contains a single value from the following options: Yangon, Naypyitaw, or Mandalay

Code:

```
## Create map_city Function
def map_city(row):
    if row['Yangon'] == 1:
        return 'Yangon'
    elif row['Naypyitaw'] == 1:
        return 'Naypyitaw'
    elif row['Mandalay'] == 1:
        return 'Mandalay'
    else:
        return 'None'
```

```
## Apply map_city Function to the dataframe
df['City'] = df.apply(map_city, axis=1)
```

Test:

```
df['City'].unique()
```

```
array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)
```

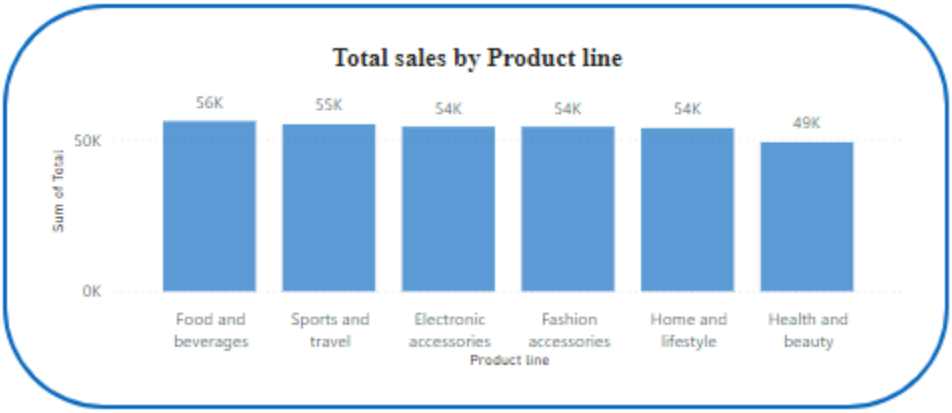
Analysis Part:

Q. NO. 1: Key insights?

Total Sales after tax	Total Sales before tax	Total Quantities	Avg. Rating
322.97K	307.59K	5510	6.97

- These metrics indicate good sales
- The Avg. Rating indicates the need for improvement in customer satisfaction, as the average rating is less than 7.

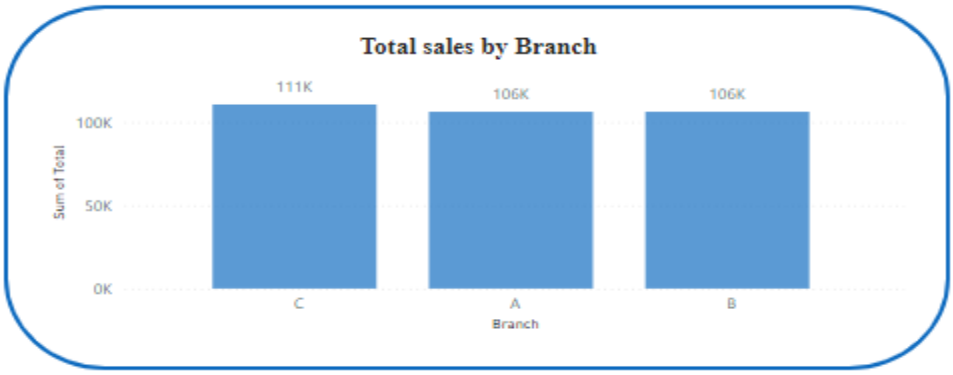
Q. NO. 2: What is the top-selling product line?



This chart indicate that top-selling product line is Food and beverages = 56.14k

Recommendation: Focus on promoting underperforming categories such as Health and Beauty with targeted discounts or product bundling to drive sales.

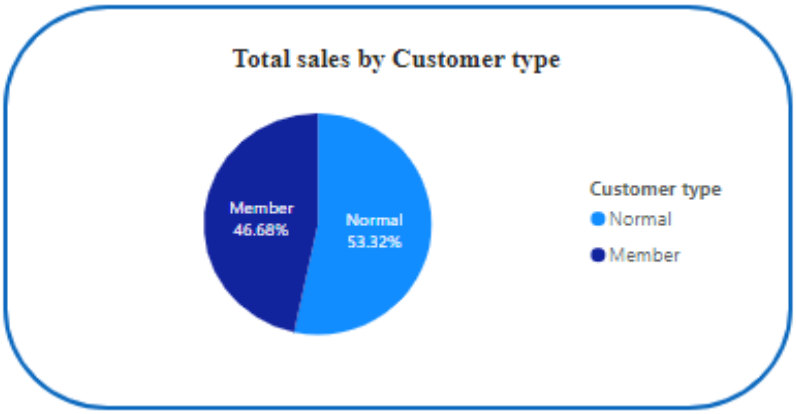
Q. NO. 3: What is the top-selling branch?



This chart indicate that top-selling Branch is C = 110.57k

Recommendation: Investigate what drives higher performance in Branch C, and consider replicating successful strategies in Branches A and B.

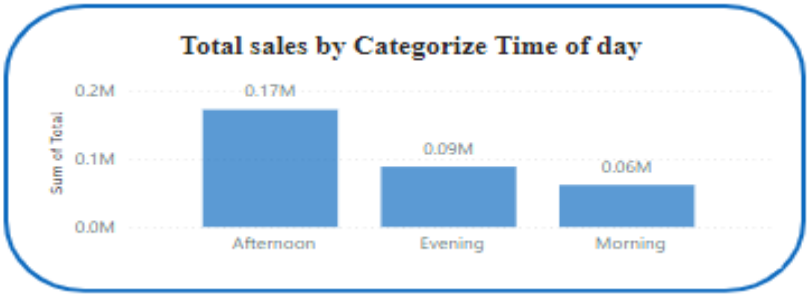
Q. NO. 4: What is the perc. of customer loyalty based on type??



This chart indicates the loyalty of normal customers more than member customers.

Recommendation: introduce program for member customers to boost loyalty as (offer special discounts, promotions, or loyalty rewards).

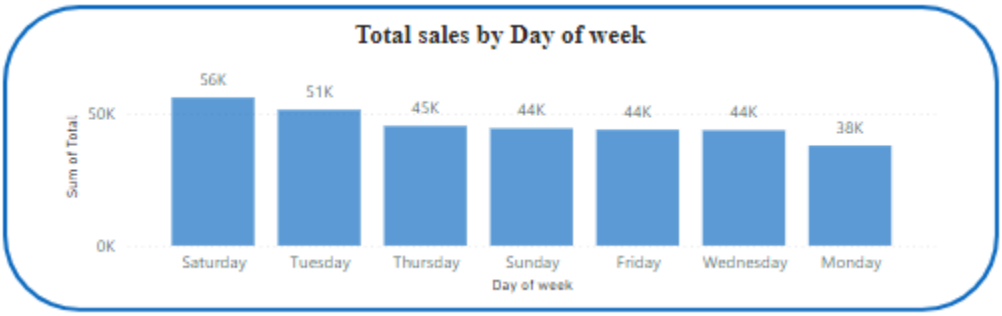
Q. NO. 5: What is the top-selling time of day?



This chart indicates that top-selling time of day is Afternoon = 146.40k.

Recommendation: To boost morning sales, the owner could offer special discounts, promotions, or loyalty rewards for customers who shop in the morning.

Q. NO. 6: What is the top-selling Day of week?



This chart indicates that top-selling Day of week is Saturday = 56.12k. (Customers might be more inclined to shop on weekends (Saturday) due to more free time or special weekend promotions , Popular products might be more available or better stocked on these days, leading to higher sales.)

Recommendation: Try to make popular products available on the days with lower sales.

Q. NO. 7: What is the relation between Product line and Branch based on Quantity?

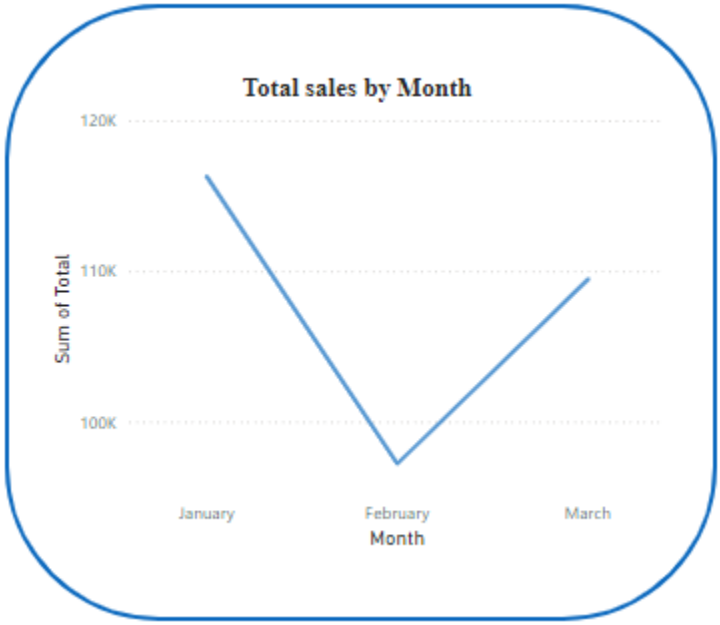
Relation between Product line and Branch by sum of Quantity

Product line	A	B	C
Electronic accessories	322	316	333
Fashion accessories	263	297	342
Food and beverages	313	270	369
Health and beauty	257	320	277
Home and lifestyle	371	295	245
Sports and travel	333	322	265

High values indicate that the branch sells more quantities of this product line, while low values indicate that the branch sells fewer quantities of this product line.

Recommendation: Examine what the successful branch is doing differently or more effectively in their sales approach. Apply similar strategies in the underperforming branch, including marketing tactics, product placement, or customer engagement techniques.

Q. NO. 8: What is the top-selling month?



- January may have benefited from post-holiday sales or new year promotions, while February often sees a drop after holiday shopping peaks. March could mark a return to normal or seasonal increases.
- February is a shorter month (28/29 days), which naturally results in fewer sales days compared to January and March.

Recommendation: In February, introduce special promotions, discounts, or limited-time offers to attract customers.