Data Assessment Report

Supermarket Sales Project

NTI Capstone Project

Team Members :

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

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 "Quantity" 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"

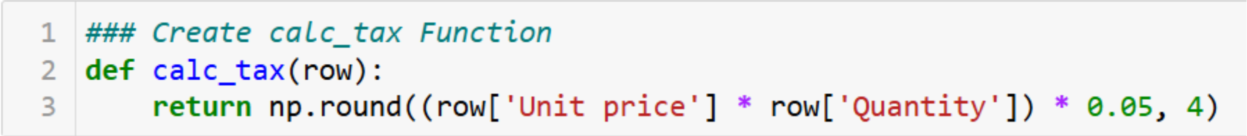
Tidiness issues exist in the data:

- 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.

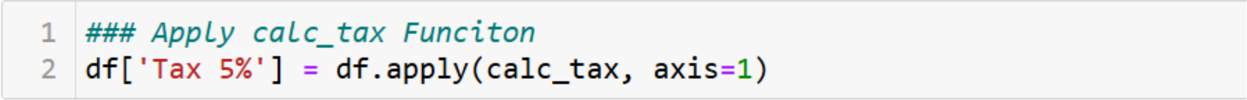
Quality issue NO. 1: The "Tax" column has 9 missing values.

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

Code:

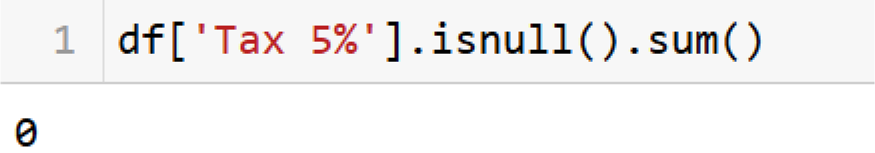


```python
### Create calc_tax Function
def calc_tax(row):
    return np.round((row['Unit price'] * row['Quantity']) * 0.05, 4)
```



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

Test:



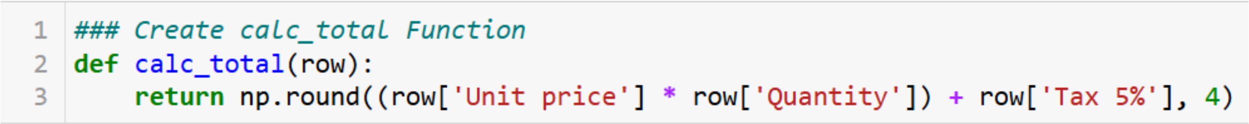
```python
df['Tax 5%'].isnull().sum()
```

```
0
```

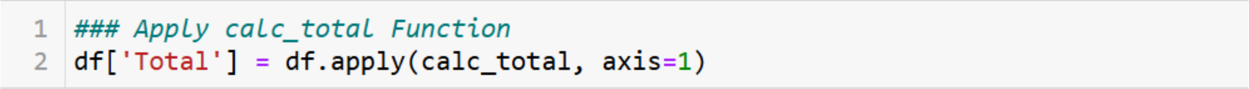
Quality issue NO. 2: The "Total" column has 3 missing values.

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

Code:

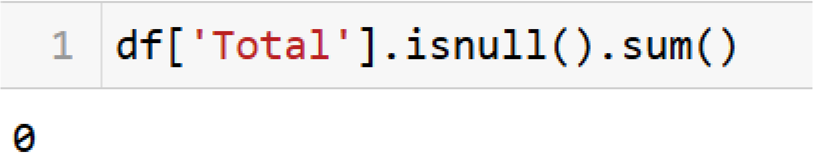


```python
### Create calc_total Function
def calc_total(row):
    return np.round((row['Unit price'] * row['Quantity']) + row['Tax 5%'], 4)
```



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

Test:



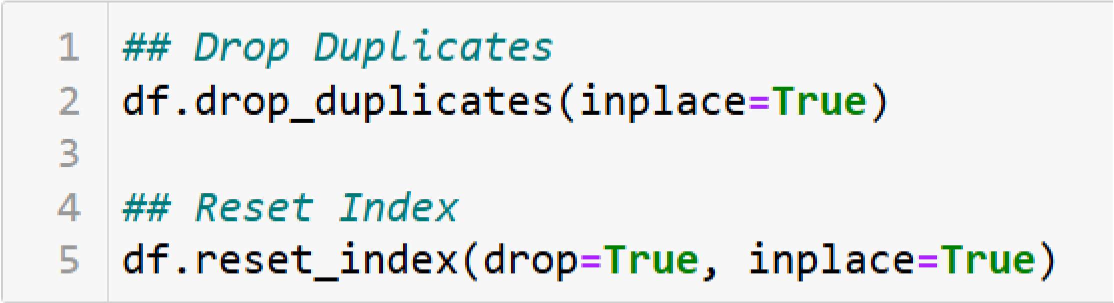
```
df['Total'].isnull().sum()
```

```
0
```

Quality issue NO. 3: There are 6 duplicated rows

Solution: Remove duplicates

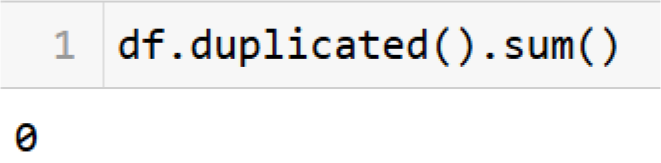
Code:



```
## Drop Duplicates
df.drop_duplicates(inplace=True)

## Reset Index
df.reset_index(drop=True, inplace=True)
```

Test:



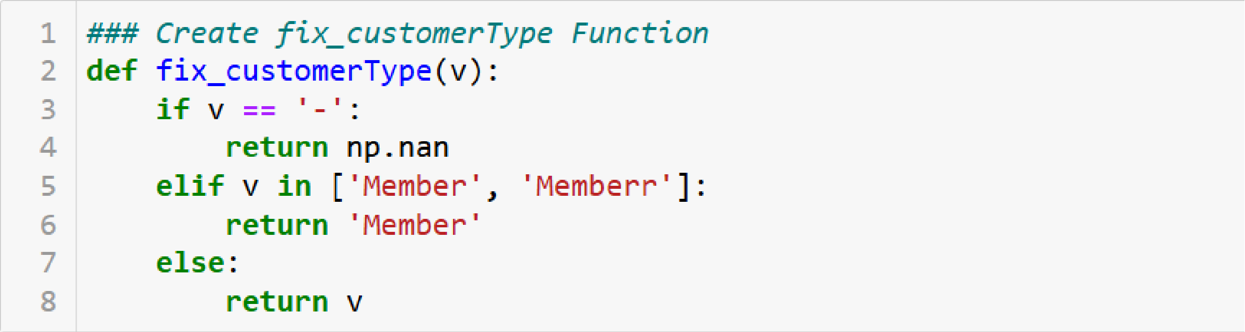
```
df.duplicated().sum()
```

```
0
```

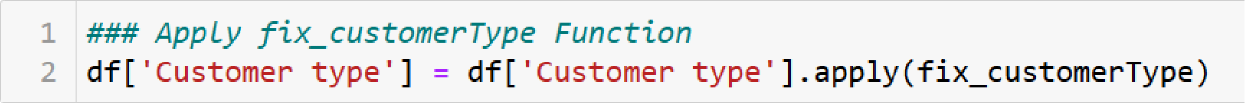
Quality issue NO. 4 and 5: The "Customer Type" field contains "-" instead of NaN and contains the values ['Member', 'Memberr'], which have the same meaning

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

Code:

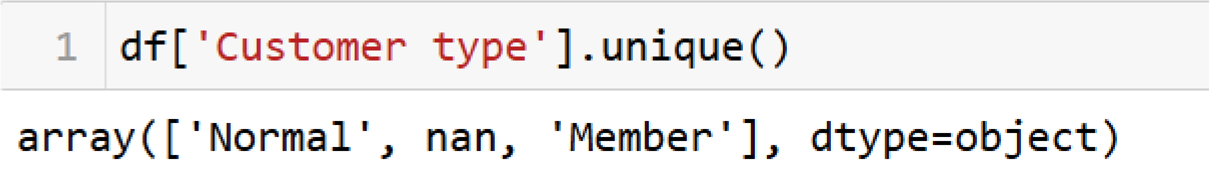


```python
### Create fix_customerType Function
def fix_customerType(v):
    if v == '-':
        return np.nan
    elif v in ['Member', 'Memberr']:
        return 'Member'
    else:
        return v
```



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

Test:



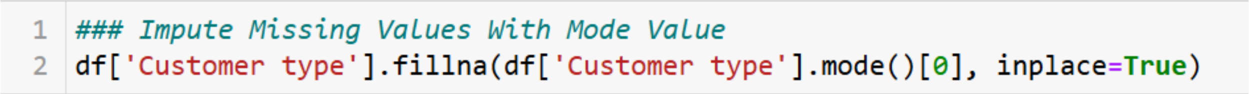
```python
df['Customer type'].unique()
```

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

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

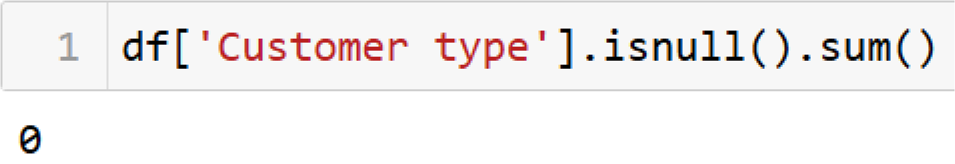
Solution: Impute missing values with Mode value

Code:



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

Test:



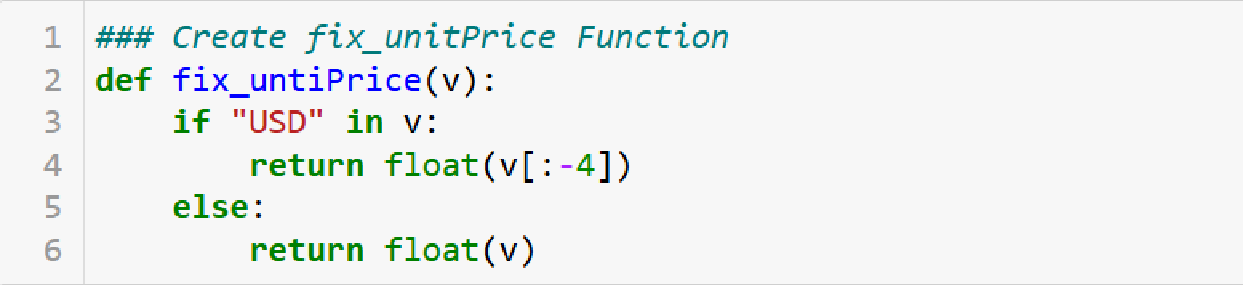
```python
df['Customer type'].isnull().sum()
```

```
0
```

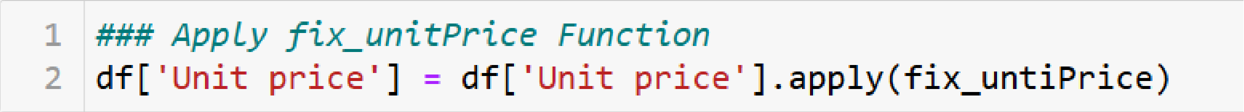
Quality issue NO. 7: The "Unit Price" column contains the value "USD" causing the data type of the column to be object instead of float.

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

Code:

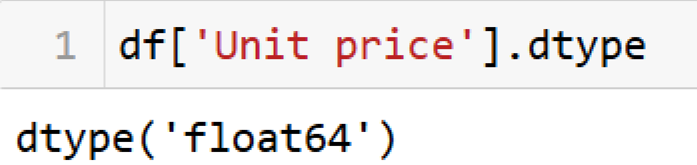


```python
### Create fix_unitPrice Function
def fix_untiPrice(v):
    if "USD" in v:
        return float(v[:-4])
    else:
        return float(v)
```



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

Test:



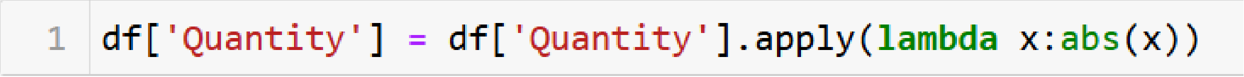
```python
df['Unit price'].dtype
```

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

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

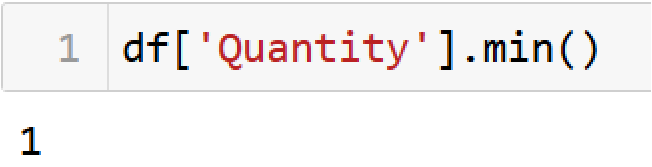
Solution: Take Absolute value of all values in Quantity Column

Code:



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

Test:



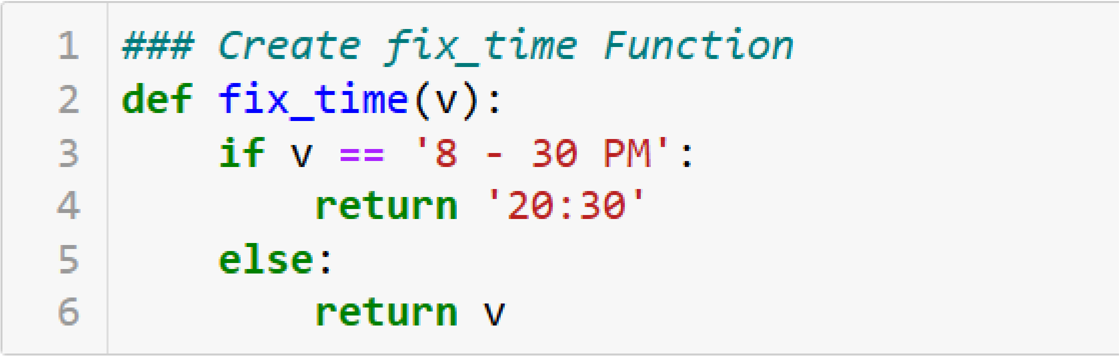
```python
df['Quantity'].min()
```

```
1
```

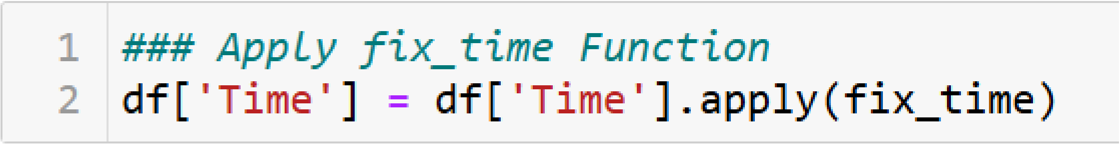
Quality issue NO. 9: The "Time" column contains the value "8 – 30 pm" instead of "20:30"

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

Code:

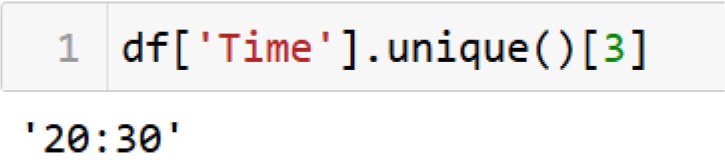


```python
### Create fix_time Function
def fix_time(v):
    if v == '8 - 30 PM':
        return '20:30'
    else:
        return v
```



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

Test:



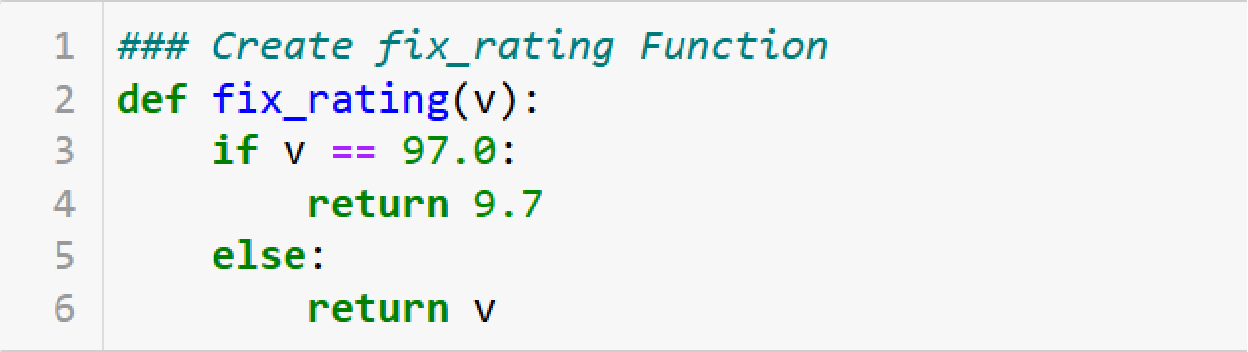
```python
df['Time'].unique()[3]
```

```
'20:30'
```

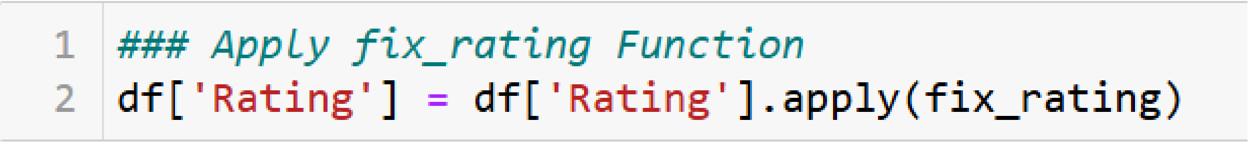
Quality issue NO. 10: The "Rating" column contains a value of "97.0" instead of "9.7"

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

Code:

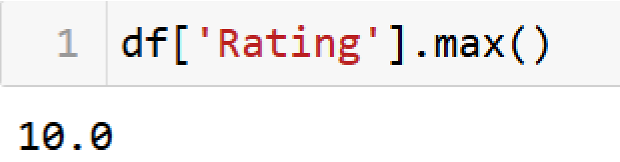


```python
### Create fix_rating Function
def fix_rating(v):
    if v == 97.0:
        return 9.7
    else:
        return v
```



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

Test:



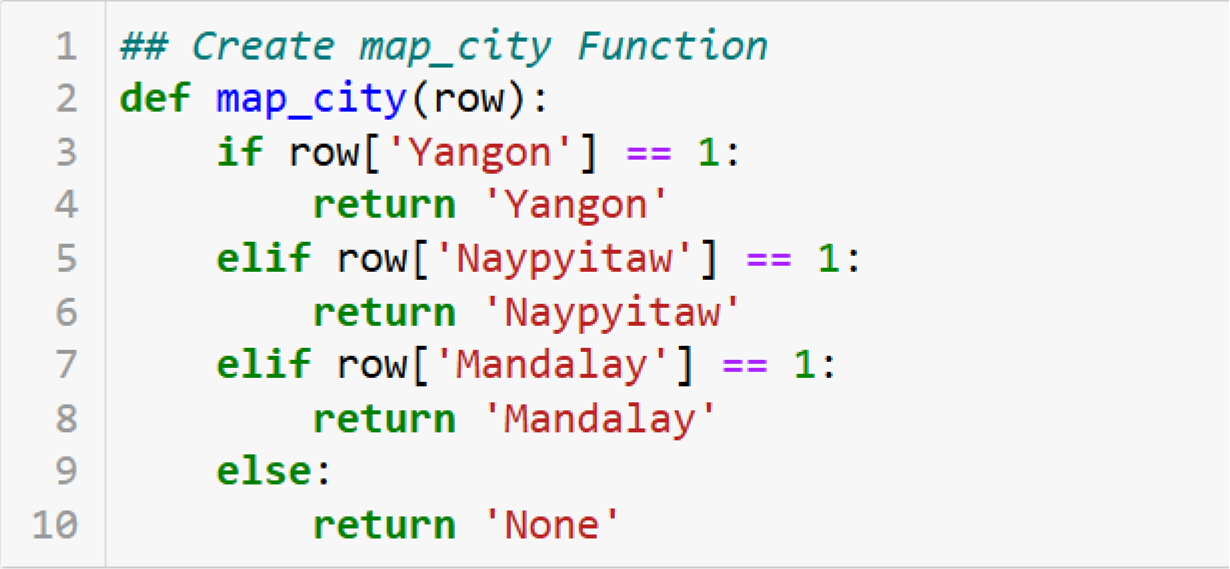
```python
df['Rating'].max()
```

```
10.0
```

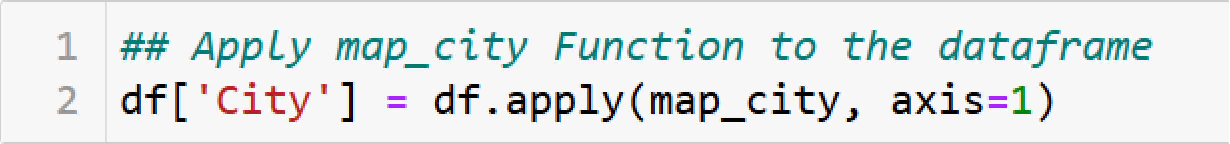
Tidiness issue NO. 1: The cities "Yangon," "Naypyidaw," and "Mandalay" are each represented in separate columns, but they should be combined into a single column called "City," with each row containing one of the three city names.

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

Code:

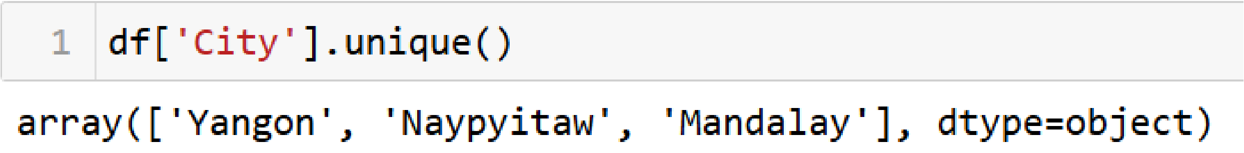


```python
## Create map_city Function
def map_city(row):
    if row['Yangon'] == 1:
        return 'Yangon'
    elif row['Naypyitaw'] == 1:
        return 'Naypyitaw'
    elif row['Mandalay'] == 1:
        return 'Mandalay'
    else:
        return 'None'
```



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

Test:



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

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