Loading the Data and Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

raw_2009_2010 = pd.read_excel('customer_transactions_sample.xlsx',
engine='openpyxl')
raw_2010_2011 = pd.read_excel('customer_transactions_sample.xlsx',
engine='openpyxl',sheet_name="Year 2010-2011")

raw_2009_2010.info()
raw_2010_2011.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525461 entries, 0 to 525460
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      525461 non-null  object
 1   StockCode    525461 non-null  object
 2   Description  522533 non-null  object
 3   Quantity     525461 non-null  int64
 4   InvoiceDate  525461 non-null  datetime64[ns]
 5   Price        525461 non-null  float64
 6   Customer ID  417534 non-null  float64
 7   Country      525461 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 32.1+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541910 entries, 0 to 541909
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      541910 non-null  object
 1   StockCode    541910 non-null  object
 2   Description  540456 non-null  object
 3   Quantity     541910 non-null  int64
 4   InvoiceDate  541910 non-null  datetime64[ns]
 5   Price        541910 non-null  float64
 6   Customer ID  406830 non-null  float64
 7   Country      541910 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
raw_2010_2011.isna().sum() # total rows 541910
```

```
Invoice              0
StockCode            0
Description       1454
Quantity             0
InvoiceDate          0
Price                0
Customer ID     135080
Country              0
dtype: int64
```

```
raw_2009_2010.isna().sum() # total rows 525462
```

```
Invoice              0
StockCode            0
Description       2928
Quantity             0
InvoiceDate          0
Price                0
Customer ID     107927
Country              0
dtype: int64
```

Removing the rows with no description.

Why?

- Rows without descriptions lack this essential information, making it difficult to interpret or analyze the transactions accurately.
- Ambiguous data could lead to biased or misleading insights, affecting the validity of our conclusions.
- Removing rows with missing descriptions improves the overall quality of the dataset by eliminating incomplete or unreliable records. This enhances the effectiveness of subsequent analyses and modeling tasks that rely on accurate and complete data.

```
clean_2010_2011 = raw_2010_2011.dropna(subset=['Description'])
clean_2009_2010 = raw_2009_2010.dropna(subset=['Description'])

clean_2010_2011.isna().sum()
clean_2009_2010.isna().sum()
```

```
Invoice              0
StockCode            0
Description          0
Quantity             0
InvoiceDate          0
Price                0
Customer ID     104999
```

```
Country          0
dtype: int64
```

Merging the data from both 2009-2010 and 2010-2011 by adding a new Column in each dataframe for evaluation further.

```
clean_2009_2010['Year'] = '2009-2010'
clean_2010_2011['Year'] = '2010-2011'
clean_df = pd.concat([clean_2009_2010, clean_2010_2011],
ignore_index=True)
clean_df
```

Creating a categorical column for cancellations based on Invoice number

- The quantity for a cancellation is indicated by -ve numbers

```
clean_df['Cancellation'] = clean_df['Invoice'].apply(lambda x: 'Yes'
if str(x).startswith('C') else 'No')
clean_df.loc[clean_df.Cancellation=='Yes']
```


```
{"summary":"{\n  \"name\": \"clean_df\",\n  \"rows\": 19494,\n
\"fields\": [\n    {\n      \"column\": \"Invoice\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 8292,\n        \"samples\": [\n
\"C564940\",\n          \"C492712\",\n          \"C507284\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"StockCode\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2898,\n
\"samples\": [\n          20681,\n          22081,\n          22192\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Description\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3078,\n        \"samples\": [\n          \"FAWN
BLUE HOT WATER BOTTLE\",\n          \"SET OF 60 I LOVE LONDON CAKE
CASES \",\n          \"TEA BAG PLATE RED RETROSPOT\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Quantity\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 805,\n
\"min\": -80995,\n        \"max\": 1,\n        \"num_unique_values\":
207,\n        \"samples\": [\n          -94,\n          -7,\n
-500\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"InvoiceDate\",\n      \"properties\": {\n        \"dtype\":
\"date\",\n        \"min\": \"2009-12-01 10:33:00\",\n        \"max\":
\"2011-12-09 11:58:00\",\n        \"num_unique_values\": 8141,\n
```

\"samples\": [\n   \"2011-10-19 14:19:00\",\n   \"2010-
01-14 11:09:00\",\n   \"2011-11-02 14:43:00\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Price\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 596.3352913716418,\n
\"min\": 0.01,\n   \"max\": 38970.0,\n
\"num_unique_values\": 1005,\n   \"samples\": [\n
33.05,\n   20.55,\n   325.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Customer ID\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1698.413276847186,\n   \"min\": 12346.0,\n   \"max\":
18287.0,\n   \"num_unique_values\": 2572,\n   \"samples\":
[\n   16894.0,\n   14976.0,\n   14141.0\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Country\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 36,\n
\"samples\": [\n   \"European Community\",\n
\"Cyprus\",\n   \"Poland\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Year\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 2,\n
\"samples\": [\n   \"2010-2011\",\n   \"2009-2010\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Cancellation\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n   \"Yes\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

Description of some items have missing names with a '?'.

- The items with description as '?' have missing customer Id and the quantity is also a negative number. It is safe to say we can drop these columns as the ratio of the rows to total data is less as well.

```
mask = clean_df[clean_df['Description'].str.startswith('?',na=False)]
clean_df = clean_df[~clean_df['Description'].str.startswith('?',
na=False)]
clean_df

{"type":"dataframe","variable_name":"clean_df"}
```

Visualization

```
import plotly.express as px

df = clean_df
custom_palette =[
```

```
    'rgb(255, 150, 238)', 'rgb(152, 115, 255)', 'rgb(151, 255, 145)',
    'rgb(0, 225, 255)', 'rgb(255, 150, 140)', 'rgb(255, 196, 48)',
    'rgb(255, 87, 101)', 'rgb(212, 255, 0)', 'rgb(166, 182, 255)',
    'rgb(255, 136, 0)', 'rgb(255, 0, 162)', 'rgb(166, 0, 255)',
    'rgb(216, 255, 209)', 'rgb(255, 194, 202)','rgb(250, 255, 110)'
]
# Group data by country and count the number of unique customers
customer_distribution = df.groupby('Country')['Customer
ID'].nunique().reset_index()
customer_distribution = customer_distribution.sort_values(by='Customer
ID', ascending=False).head(12)
fig = px.bar(customer_distribution, x='Country', y='Customer ID',
             title='Customer Distribution by Country',
             labels={'Country': 'Country', 'Customer ID': 'Number of
Customers'},
              color='Country', color_discrete_sequence=custom_palette)

fig.update_yaxes(range=[0, 1000])
fig.update_layout(
    title=dict(
        text='Customer Distribution by Country',
        x=0.5,
        font=dict(
            size=24
        )
    )
)
fig.show()


product_counts = df['Description'].value_counts().reset_index()
product_counts.columns = ['Description', 'Frequency']


product_counts = product_counts.sort_values(by='Frequency',
ascending=False)

top_products = product_counts.head(10)
fig = px.bar(top_products, x='Description', y='Frequency',
             title='Top 10 Most Frequently Purchased Products',
             labels={'Description': 'Product Description',
'Frequency': 'Frequency'},
             color='Description',
color_discrete_sequence=custom_palette)
fig.update_layout(xaxis_tickangle=-45)
fig.update_layout(
    title=dict(
        text='Top 10 Most Frequently Purchased Products',
        x=0.5,
        font=dict(
```

```
            size=24
        )
    )
)
fig.show()

df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')
df['YearMonth'] = df['YearMonth'].astype(str)

# Group data by year and month and count the number of cancellations
cancellations_over_time = df[df['Cancellation'] ==
'Yes'].groupby('YearMonth').size().reset_index(name='Cancellation
Count')
fig = px.scatter(cancellations_over_time, x='YearMonth',
y='Cancellation Count',
                 title='Frequency of Cancellations Over Time',
                 labels={'YearMonth': 'Year-Month', 'Cancellation
Count': 'Cancellation Count'},
                 size='Cancellation Count',
                 size_max=12,
                 )
fig.add_trace(px.line(cancellations_over_time, x='YearMonth',
y='Cancellation Count').data[0])

fig.update_layout(
    title=dict(
        text='Frequency of Cancellations Over Time',
        x=0.5,
        font=dict(
            size=24
        )
    )
)

fig.show()

revenue_over_time = df.groupby('YearMonth')
['Price'].sum().reset_index()
fig = px.scatter(revenue_over_time, x='YearMonth', y='Price',
                 title='Revenue Trend Over Time',
                 labels={'YearMonth': 'Year-Month', 'Price':
'Revenue'},
                 size='Price',
                 size_max=12,
                  color_discrete_sequence=['springgreen']
                 )
fig.add_trace(px.line(revenue_over_time, x='YearMonth',
y='Price',color_discrete_sequence=['limegreen'] ).data[0])
fig.update_layout(
```

```
    title=dict(
        text='Revenue Trend Over Time',
        x=0.5,
        font=dict(
            size=24
        )
    )
)


fig.show()

# Convert 'InvoiceDate' to datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')
df['YearMonth'] = df['YearMonth'].astype(str)

# Group data by customer and year-month, and calculate the number of
active customers for each period
active_customers = df.groupby(['YearMonth', 'Customer
ID']).size().reset_index(name='Active')


active_customers['PreviousActive'] =
active_customers.groupby('Customer ID')['Active'].shift(1)

active_customers['Retained'] = active_customers['Active'] /
active_customers['PreviousActive']

active_customers['Retained'].fillna(0, inplace=True)

retention_rate = active_customers.groupby('YearMonth')
['Retained'].mean().reset_index()
fig = px.scatter(retention_rate, x='YearMonth', y='Retained',
                 title='Customer Retention Rate Over Time',
                 labels={'YearMonth': 'Year-Month', 'Retained':
'Retention Rate'},
                 size='Retained',
                 size_max=12,
                 )
fig.add_trace(px.line(retention_rate, x='YearMonth',
y='Retained').data[0])

fig.update_layout(
    title=dict(
        text='Customer Retention Rate Over Time',
        x=0.5,
        font=dict(
            size=24
        )
```

```
    )
)

fig.show()
```