4.10: Coding Etiquette & Excel Reporting

Step 1: Importing Libraries

```
import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

import scipy
```

Step 2. Importing Data Sets

```
# Defining path

path = r'C:\Users\Admin\Documents\18-07-2023 Instacart Basket Analysis'

# Import orders_products_all.pkl

ords_prods_all = pd.read_pickle(os.path.join(path, '02 Data', 'Prepared Data', 'orders_products_all.pkl'))

# Display max rows and columns

pd.options.display.max_rows = None

pd.options.display.max_columns = None

# Check the data

ords_prods_all.head()
```

```
ords_prods_all.shape
```

Step 3. Data Security Implications

Our data consists names of customers. To ensure our data GDPR requirements, we will drop these columns from the dataframe.

```
# Dropping the First Name column

ords_prods_all_1 = ords_prods_all.drop(columns = ['First Name'])

# Dropping the Surname columns

ords_prods_all_1 = ords_prods_all_1.drop(columns = ['Surname'])

# Check the new dataframe

ords_prods_all_1.head()
```

Step 4. Regional Segmentation & Analysis¶

```
# We're going to use states to create a new region column. We will do this with the loc() function

# First, we will do the Midwest states

ords_prods_all_1.loc[ords_prods_all_1['STATE'].str.contains('Wisconsin|Michigan|Illinois|Indiana|Ohio|North Dakota|South Dakota|Nebraska|Kansas|Minnesota|Iowa|Missouri'), 'Region'] = 'Midwest'

# Flag the West states

ords_prods_all_1.loc[ords_prods_all_1['STATE'].str.contains('Idaho|Montana|Wyoming|Nevada|Utah|Colorado|Arizona|New Mexico|Alaska|Washington|Oregon|California|Hawaii'), 'Region'] = 'West'
```

```
# Flag the Northeast states

ords_prods_all_1.loc[ords_prods_all_1['STATE'].str.contains('Maine|New Hampshire|Vermont|Massachusetts|Rhode Island|Connecticut|New York|Pennsylvania|New Jersey'), 'Region'] = 'Northeast'

# Flag the South states

ords_prods_all_1.loc[ords_prods_all_1['STATE'].str.contains('Delaware|Maryland|District of Columbia|Virginia|West Virginia|North Carolina|South Carolina|Georgia|Florida|Kentucky|Tennessee|Mississippi|Alabama|Oklahoma|Texas|Arkansas|Louisiana'), 'Region'] = 'South'

# Check the dataframe

ords_prods_all_1.head(100)

# Check the frequency of the Region column

ords_prods_all_1['Region'].value_counts(dropna = False)

# Now we'll analyze spending habits by region with a crosstab

crosstab = pd.crosstab(ords_prods_all_1['Region'], ords_prods_all_1['spending_flag'], dropna = False)

crosstab.to_clipboard()
```

Upon analyzing the results of the crosstab, we will get to know that whaich region has the lowest spenders a nd which region has the highest spenders.

Step 5. Creating An Exclusion Flag for Customer Activity

```
# We want to remove all users with less than 5 orders. To do so, we will create an exclusion flag using the max_order

# column.

ords_prods_all_1.loc[ords_prods_all_1['max_order'] >= 5, 'exclusion_flag'] = 'High activity customer'

ords_prods_all_1.loc[ords_prods_all_1['max_order'] < 5, 'exclusion_flag'] = 'Low activity customer'

# Check the frequency results of our exclusion flag

ords_prods_all_1['exclusion_flag'].value_counts(dropna = False)

# Create a dataframe of the low activity customers

df_low = ords_prods_all_1[ords_prods_all_1['exclusion_flag'] == 'Low activity customer']

# Check that we have the right amount of rows in our low activity dataframe

df_low.shape

# Double-check the dataframe

df_low.head(30)

# Export this dataframe

df_low.to_pickle(os.path.join(path, '02 Data','Prepared Data', 'low_activity_customers.pkl'))
```

```
# Remove the low activity customers from the main dataframe

ords_prods_all_2 =
ords_prods_all_1[ords_prods_all_1['exclusion_flag'] == 'High activity
customer']

# Check that we have the right amount of rows in our high activity
dataframe

ords_prods_all_2.shape
```

Step 6. Creating Groupings for Customer Income¶

```
# We want to group customers by income. We will create a low
income, middle income, and upper income flag based on

# income. First, we create the low income group.

ords_prods_all_2.loc[ords_prods_all_2['income'] <= 52000,
'income_group'] = 'Low Income'

# Next, we create the middle income group

ords_prods_all_2.loc[(ords_prods_all_2['income'] > 52000) &
(ords_prods_all_2['income'] <= 150000), 'income_group'] = 'Middle
Income'

#Lastly, we create the upper income group

ords_prods_all_2.loc[ords_prods_all_2['income'] > 150000,
'income_group'] = 'Upper Income'

ords_prods_all_2.head()

# Check the frequency of the income_group column

ords_prods_all_2['income_group'].value_counts(dropna = False)
```

Step 7. Creating Groupings for Customer Age¶

We want to group customers by age. We will create a young adult, middle aged, and elderly flag based on age

First, we create the young adult group.

```
ords_prods_all_2.loc[ords_prods_all_2['Age'] <= 30, 'age_group'] = 'Young Adult'
```

Next, we create the middle aged group.

```
ords_prods_all_2.loc[(ords_prods_all_2['Age'] > 30) & (ords_prods_all_2['Age'] <= 60), 'age_group'] = 'Middle Aged'
```

Lastly, we create the elderly group.

```
ords_prods_all_2.loc[ords_prods_all_2['Age'] > 60, 'age_group'] = 'Elderly'
```

```
ords_prods_all_2.head()
```

Check the frequency of the age_group column

```
ords_prods_all_2['age_group'].value_counts(dropna = False)
```

Step 8. Creating Groupings for Dependants¶

We want to group customers dependants. We will create a no dependants and has dependents grouping.

First, we create the no dependants group.

```
ords_prods_all_2.loc[ords_prods_all_2['num_of_dependants'] == 0, 'dependants_flag'] = 'No Dependants'
```

#Next, we'll great the has dependants group.

ords_prods_all_2.loc[ords_prods_all_2['num_of_dependants'] > 0, 'dependants_flag'] = 'Has Dependants'

ords_prods_all_2.head()

Check the frequency of the dependants_flag column

ords_prods_all_2['dependants_flag'].value_counts(dropna = False)

Step 9. Merge Department Names To Main Dataframe¶

Import departments data set

df_depts = pd.read_csv(os.path.join(path, '02 Data', 'Prepared Data', 'departments_wrangled.csv'), index_col = False)

Check

df_depts.head()

Renamed unnamed: 0 column to department_id

df_depts = df_depts.rename(columns={'Unnamed: 0': 'department_id'})

Check

df_depts.head(22)

Merge with main dataframe

df_merged_1 = df_depts.merge(ords_prods_all_2, on = 'department_id')

Check

df_merged_1.head(50)

```
# Check frequency of the department column

df_merged_1['department'].value_counts(dropna = False)

#Rename df_merged_1 to ords_prods_all

ords_prods_all = df_merged_1

# Check

ords_prods_all.head(50)
```

Step 10. Creating Customer Profiles

```
# We want to create customer profiles based on certain demographics to help inform our marketing efforts. Using

# various criteria, we will create profiles for young parents, middle-aged parents, elderly parents,

# young adults w/ no children, middle-aged w/ no children, elderly w/ no children

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'Has Dependants') & (ords_prods_all['age_group'] == 'Young Adult'), 'customer_profile'] = 'Young parent'

# Creating the middle-aged parents profile

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'Has Dependants') & (ords_prods_all['age_group'] == 'Middle Aged'), 'customer_profile'] = 'Middled-aged parents'

# Creating the Elderly parents profile

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'Has Dependants') & (ords_prods_all['age_group'] == 'Elderly'), 'customer_profile'] = 'Elderly parents'
```

```
# Creating the young adults w/ no children profile

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'No Dependants') & (ords_prods_all['age_group'] == 'Young Adult'), 'customer_profile'] = 'Young adults w/ no children'

# Creating the middle-aged w/ no children profile

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'No Dependants') & (ords_prods_all['age_group'] == 'Middle Aged'), 'customer_profile'] = 'Middle-aged w/ no children'

# Creating the elderly w/ no children profile

ords_prods_all.loc[(ords_prods_all['dependants_flag'] == 'No Dependants') & (ords_prods_all['age_group'] == 'Elderly'), 'customer_profile'] = 'Elderly w/ no children'

# Check frequency of the customer_profile column

ords_prods_all['customer_profile'].value_counts(dropna = False)

# Check

ords_prods_all.head(100)
```

Step 11. Visualizing Distribution of Customer Profiles

```
# Create a pie chart to visualize the distribution of customer profiles

pie_cust_prof = ords_prods_all['customer_profile'].value_counts().plot.pie(figsize = (8,8), colors = sns.color_palette('Oranges', 6), autopct = '%1.1f%%')

plt.ylabel('') plt.title('Distribution of Customer Profiles', fontsize = 13)

plt.show()
```

By seeing this Piechart we will get to know that which age group people has highest number of customers.

Export this chart

pie_cust_prof.figure.savefig(os.path.join(path, '04 Analysis','Visualizations', 'dist_cust_profs.png'), bbox_inches = "tight")

Step 12. Creating Aggregations

Aggregating max, mean, and min of customer_profile by expenditure

ords_prods_all.groupby('customer_profile').agg({'prices': ['mean', 'min', 'max']})

Aggregating max, mean, and min of customer_profile by usage frequency

ords_prods_all.groupby('customer_profile').agg({'days_since_prior_order': ['mean', 'min', 'max']})

Step 13. Comparing Customer Profiles w/ Regions & Departments

We want to compare customer profiles across each region. We will do so with 100% stacked charts.

First, we create a normalized cross tab that we'll use for the data

```
cross_cust_region_1 = pd.crosstab(index=ords_prods_all['Region'],
                                  columns=ords_prods_all['customer_profile'],
                                  normalize="index")
```

```
# Next, we create a cross tab to display the data labels on the plotcross_cust_region_labels =
pd.crosstab(index=ords_prods_all['Region'],
                    columns=ords_prods_all['customer_profile'])

# Now, we create the stacked chart

cust_prof_region = cross_cust_region_1.plot(kind='bar',
                    stacked=True,
                    colormap='tab10',
                    figsize=(10, 6))
plt.title('Percent of Customer Profiles by Region, Normalized')
plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.xlabel("Region")
plt.ylabel("Proportion")
plt.xticks(rotation = 0, fontsize=10)
for n, x in enumerate([*cross_cust_region_labels.index.values]):
   for (proportion, y_loc) in zip(cross_cust_region_1.loc[x],
                        cross_cust_region_1.loc[x].cumsum()):
plt.text(x=n - 0.17,y=(y_loc - proportion) + (proportion / 2),
            s=f'{np.round(proportion * 100, 1)}%', color="black",
            fontsize=12, fontweight="bold") plt.show()
```

By seeing this stacked chart, we will get to know that which percentage has having highest customer profiles by region.

```
# Export this chart

cust_prof_region.figure.savefig(os.path.join(path, '04 Analysis','Visualizations', 'cust_prof_region.png'), bbox_inches = "tight")

# We want to compare customer profiles across each department. We will do so with 100% stacked charts.

# First, we create a normalized cross tab that we'll use for the data

cross_cust_dept = pd.crosstab(index=ords_prods_all['department'],

                              columns=ords_prods_all['customer_profile'],

                              normalize="index")

# Next, we create a cross tab to display the data labels on the plot

cross_cust_dept_labels = pd.crosstab(index=ords_prods_all['department'],

                              columns=ords_prods_all['customer_profile'])

# Now, we create the stacked chart

cust_prof_dept = cross_cust_dept.plot(kind='bar',

                 stacked=True,

                 colormap='tab10',

                 figsize=(10, 6))
```

```
plt.title('Percent of Customer Profiles by Department, Normalized')

plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')

plt.xlabel("Department")

plt.ylabel("Proportion")

plt.show()
```

By seeing this chart we will get to know that which department has highest and lowest percentage of customer profiles.

```
# Export this chart

cust_prof_dept.figure.savefig(os.path.join(path, '04 Analysis','Visualizations', 'cust_prof_dept.png'), bbox_inches = "tight")

# compare the customer profiles per department

cross_cust_dept_2 = pd.crosstab(index=ords_prods_all['department'],

                                columns=ords_prods_all['customer_profile'])

# Bar chart for department sales from elderly parents

bar_elderly_par = cross_cust_dept_2.sort_values(['Elderly parents']).plot.barh(y='Elderly parents',


color="Purple").legend(loc='lower right')

plt.title('Department Sales by Family Status',fontsize=13)

plt.xlabel('')

plt.ylabel('')
```

By seeing this chart, we can see that Department sales by Elderly parents

```
# Export this chart

bar_elderly_par.figure.savefig(os.path.join(path, '04 Analysis','Visualizations', 'bar_elderly_kids.png'), bbox_inches = "tight")

# Bar chart for department sales from middle-aged parents

bar_mid_par = cross_cust_dept_2.sort_values(['Middled-aged parents']).plot.barh(y='Middled-aged parents',


color="Pink").legend(loc='lower right')

plt.title('Department Sales by Family Status',fontsize=13)

plt.xlabel('')

plt.ylabel('')
```

By seeing this chart, we can see that Department sales by middled aged parents

```
# Export this chart

bar_mid_par.figure.savefig(os.path.join(path, '04 Analysis','Visualizations', 'bar_middle_kids.png'), bbox_inches = "tight")

# Bar chart for department sales from young parents

bar_young_par = cross_cust_dept_2.sort_values(['Young parent']).plot.barh(y='Young parent',


color="Red").legend(loc='lower right')
```

```
plt.title('Department Sales by Family Status',fontsize=13)

plt.xlabel('')

plt.ylabel('')
```

By plotting graphs of all kind of people we will get to know that the department sales by every group of people. So as a Data analyst we can analyze according to that.