

รายงาน

เรื่อง

การวิเคราะห์กลุ่มลูกค้า

(Customer Personality Analysis)

จัดทำโดย

- นายกัญจน์ ทิรัญมาศสุวรรณ รหัส 6209610234
- นางสาวอธิตา ชลathem รหัส 6209610358
- นางสาวสุภัสสรา พิมพันธ์ รหัส 6209680013

เสนอ

ผศ.ดร.วิรัตน์ จาเรืองศ์เพบูลย์

รายวิชา คพ.345 การเรียนรู้ของเครื่องจักรและทำเหมืองข้อมูลเชิงประยุกต์

โครงการปริญญาตรีภาคปกติ ภาคเรียนที่ 1 ปีการศึกษา 2564

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต

สารบัญ

บทที่ 1 บทนำ

1

- ทีมและความสำคัญ
- วัตถุประสงค์
- คำอ่านและปัญหาที่น่าสนใจ

บทที่ 2 เอกสารที่เกี่ยวข้อง

2 - 4

- Data Exploration

บทที่ 3 วิธีการจัดทำและผลการศึกษา

5 -

- Data Preparation
- Data Preprocessing and Data Representation
- Modeling
- Model Evaluation

บทที่ 4 สรุปและขอเสนอแนะ

64 - 65

บทที่ 1

บทนำ

1. ที่มาและความสำคัญ

เมื่อพูดถึง “ลูกค้า” เราต้องชัดเจน และรู้ให้ลึกมากที่สุด ลูกค้ากำลังมองหาอะไร มีเหตุผลใดบ้างที่ ลูกค้าใช้ในการตัดสินใจเลือกหรือไม่เลือกสินค้าหรือบริการ เป็นที่ทราบกันในวงการการตลาดปัจจุบันว่า การรู้ ว่าลูกค้าของเรามีใคร? จะทำให้การตอบโจทย์ลูกค้าด้วยกลยุทธ์การตลาดต่าง ๆ ทำได้อย่างมีประสิทธิภาพ ทั้งในด้านการพัฒนาสินค้า และราคา ที่จะทำให้ลูกค้ายอมรับ และตัดสินใจ “ซื้อ” สินค้าได้ง่ายขึ้น หรือในมิติ ด้านช่องทาง และการส่งเสริมการตลาด จะทำให้รู้ว่าควรสื่อสารอย่างไร? ให้เข้าถึง Insight จนสามารถกระตุ้น ให้ลูกค้า “ซื้อ” สินค้าในที่สุด

ทางคณะผู้จัดทำจึงสนใจที่จะศึกษาเรื่อง การวิเคราะห์กลุ่มลูกค้า (Customer Personality Analysis) เพื่อหาว่าเรา สามารถแบ่งกลุ่มลูกค้าได้เป็นกี่ประเภท เพื่อที่จะตอบสนองความต้องการของลูกค้า มากที่สุด ปัจจัยที่มีผลต่อการซื้อสินค้าของลูกค้า

2. วัตถุประสงค์

2.1 เพื่อต้องการศึกษาเกี่ยวกับการวิเคราะห์ข้อมูลของกลุ่มลูกค้า

2.2 เพื่อศึกษาและทำความเข้าใจเกี่ยวกับ การเรียนรู้ของเครื่องจักร หรือ Machine Learning ได้ อย่างเข้าใจมากยิ่งขึ้น

2.3 เพื่อทำการคำตوبของปัญหาจากชุดข้อมูลที่สังสัย และต้องการคำตوب

3. คำถามที่เกี่ยวกับข้อมูลที่สนใจ

3.1 สามารถแบ่งกลุ่มลูกค้าได้เป็นกี่ประเภท เพื่อที่จะตอบสนองความต้องการของลูกค้ามากที่สุด

3.2 ลูกค้ากลุ่มอายุเท่าไรที่มีการซื้อสินค้าเป็นจำนวนมากที่สุด

3.3 ปัจจัยที่มีผลต่อการซื้อสินค้าของลูกค้า

3.4 ลูกค้าที่มาใช้บริการ มีการตอบรับ Campaign ทั้งหมด ส่วนใหญ่เป็นลูกค้ากลุ่มไหน

3.5 สินค้าที่เป็นที่นิยมที่ลูกค้าซื้อมากที่สุดคือสินค้าประเภทอะไร

บทที่ 2

เอกสารที่เกี่ยวข้อง

1. Data Exploration

- ทำการ import ไลบรารี ที่จำเป็นต้องใช้สำหรับการทำรายงาน

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

- ทำการ import ไลบรารีต่างๆ ที่ต้องใช้ในการให้สิทธิ์การเข้าถึง Google Drive ผ่านทาง Colaboratory

```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

- ทำการ download ไฟล์ข้อมูล dataset ที่จำเป็นต้องใช้ โดยชุดข้อมูลที่ใช้มาจากการ

<https://www.kaggle.com/imakash3011/customer-personality-analysis>

```
downloaded = drive.CreateFile({'id': '1MqsMH0kAjT_vRsnDavdaSmT-smJWlQkH'})
downloaded.GetContentFile('marketing_campaign.csv')
```

4. ทำการอ่านไฟล์และสร้างตัวแปร customer เพื่อใช้สำหรับทำขั้นตอนถัดไป

```
customer = pd.read_csv('marketing_campaign.csv', sep='\t')
customer
```

ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	NumWebPurchases	
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	88	546	172	88	88	3	8
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	1	6	2	1	6	2	1
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	49	127	111	21	42	1	8
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	4	20	10	3	5	2	2
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	43	118	46	27	15	5	5
...
2235	10870	1967	Graduation	Married	61223.0	0	1	13-06-2013	46	709	43	182	42	118	247	2	9
2236	4001	1946	PhD	Together	64014.0	2	1	10-06-2014	56	406	0	30	0	0	8	7	8
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25-01-2014	91	908	48	217	32	12	24	1	2
2238	8235	1956	Master	Together	69245.0	0	1	24-01-2014	8	426	30	214	80	30	61	2	6
2239	9405	1954	PhD	Married	52869.0	1	1	15-10-2012	40	84	3	61	2	1	21	3	3

2240 rows × 29 columns

5. ทำการ print ค่า shape ของข้อมูลของมาเพื่อดูว่ามี กี่ ข้อมูล และ feature ซึ่งชุดข้อมูลมีทั้งหมด 2240 instances และมี 29 features

```
print("Shape of the DataFrame is :",customer.shape)
```

Shape of the DataFrame is : (2240, 29)

6. ทำการ print ค่า columns ของ customer ออกมาดู โดยใช้ฟังก์ชัน .columns เพื่อดู feature ของ customer

```
print("\nColumns in DataFrame is :\n",customer.columns)
```

```
Columns in DataFrame is :
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
```

7. ໃຊ້ພັກສັນ .info() ເພື່ອຄູ່ຂໍ້ມູນ information ກາພຣວມ ຂອງ customer

```
print("Print a Summary of a Dataframe is : ",customer.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               2240 non-null    int64  
 1   Year_Birth        2240 non-null    int64  
 2   Education         2240 non-null    object  
 3   Marital_Status    2240 non-null    object  
 4   Income            2216 non-null    float64 
 5   Kidhome           2240 non-null    int64  
 6   Teenhome          2240 non-null    int64  
 7   Dt_Customer       2240 non-null    object  
 8   Recency           2240 non-null    int64  
 9   MntWines          2240 non-null    int64  
 10  MntFruits         2240 non-null    int64  
 11  MntMeatProducts   2240 non-null    int64  
 12  MntFishProducts   2240 non-null    int64  
 13  MntSweetProducts  2240 non-null    int64  
 14  MntGoldProds      2240 non-null    int64  
 15  NumDealsPurchases 2240 non-null    int64  
 16  NumWebPurchases   2240 non-null    int64  
 17  NumCatalogPurchases 2240 non-null    int64  
 18  NumStorePurchases 2240 non-null    int64  
 19  NumWebVisitsMonth 2240 non-null    int64  
 20  AcceptedCmp3      2240 non-null    int64  
 21  AcceptedCmp4      2240 non-null    int64  
 22  AcceptedCmp5      2240 non-null    int64  
 23  AcceptedCmp1      2240 non-null    int64  
 24  AcceptedCmp2      2240 non-null    int64  
 25  Complain          2240 non-null    int64  
 26  Z_CostContact     2240 non-null    int64  
 27  Z_Revenue          2240 non-null    int64  
 28  Response          2240 non-null    int64
```

บทที่ 3

วิธีการจัดทำและผลการศึกษา

ขั้นตอนที่ 1: Data Preparation

1. Data Cleansing

- ทำการหาค่า Missing value ของ dataset customer โดยใช้ฟังก์ชัน .isnull() เพื่อหาค่า Missing value และ ฟังก์ชัน .sum() เพื่อร่วมค่า Missing value ทั้งหมด โดย feature ที่มีค่า Missing value คือ Income 24 ตัว

```
customer.isnull().sum()
# 24 missing values -> income

ID              0
Year_Birth       0
Education        0
Marital_Status   0
Income           24
Kidhome          0
Teenhome         0
Dt_Customer      0
Recency          0
MntWines         0
MntFruits        0
MntMeatProducts  0
MntFishProducts  0
MntSweetProducts 0
MntGoldProducts  0
NumDealsPurchases 0
NumWebPurchases  0
NumCatalogPurchases 0
NumStorePurchases 0
NumWebVisitsMonth 0
AcceptedCmp3      0
AcceptedCmp4      0
AcceptedCmp5      0
AcceptedCmp1      0
AcceptedCmp2      0
Complain         0
Z_CostContact    0
Z_Revenue         0
Response          0
dtype: int64
```

- ทำการกำจัดค่า Missing value โดยเติม ค่า median ของข้อมูลเข้าไป จากนั้นเช็ควิเคราะห์เพื่อดูว่า มีค่า Missing value หรือไม่

```

customer['Income'] = customer['Income'].fillna(customer['Income'].median())
customer.isnull().sum()

ID          0
Year_Birth   0
Education    0
Marital_Status 0
Income       0
Kidhome     0
Teenhome    0
Dt_Customer 0
Recency     0
MntWines    0
MntFruits   0
MntMeatProducts 0
MntFishProducts 0
MntSweetProducts 0
MntGoldProds 0
NumDealsPurchases 0
NumWebPurchases 0
NumCatalogPurchases 0
NumStorePurchases 0
NumWebVisitsMonth 0
AcceptedCmp3 0
AcceptedCmp4 0
AcceptedCmp5 0
AcceptedCmp1 0
AcceptedCmp2 0
Complain    0
Z_CostContact 0
Z_Revenue    0
Response    0
dtype: int64

```

3. ทำการตรวจสอบว่า record ซ้ำกันหรือไม่ โดยสามารถใช้ฟังก์ชัน .duplicated() ได้ผลลัพธ์ false
คือข้อมูลไม่ซ้ำกัน

```
customer.duplicated().any()
```

False

4. ทำการดูค่า unique ของแต่ละคอลัมน์ โดยใช้ ฟังก์ชัน .nunique()

```
customer.nunique()
```

ID	2240
Year_Birth	59
Education	5
Marital_Status	8
Income	1975
Kidhome	3
Teenhome	3
Dt_Customer	663
Recency	100
MntWines	776
MntFruits	158
MntMeatProducts	558
MntFishProducts	182
MntSweetProducts	177
MntGoldProds	213
NumDealsPurchases	15
NumWebPurchases	15
NumCatalogPurchases	14
NumStorePurchases	14
NumWebVisitsMonth	16
AcceptedCmp3	2
AcceptedCmp4	2
AcceptedCmp5	2
AcceptedCmp1	2
AcceptedCmp2	2
Complain	2
Z_CostContact	1
Z_Revenue	1
Response	2
dtype: int64	

2. Data Analysis & Data Visualization

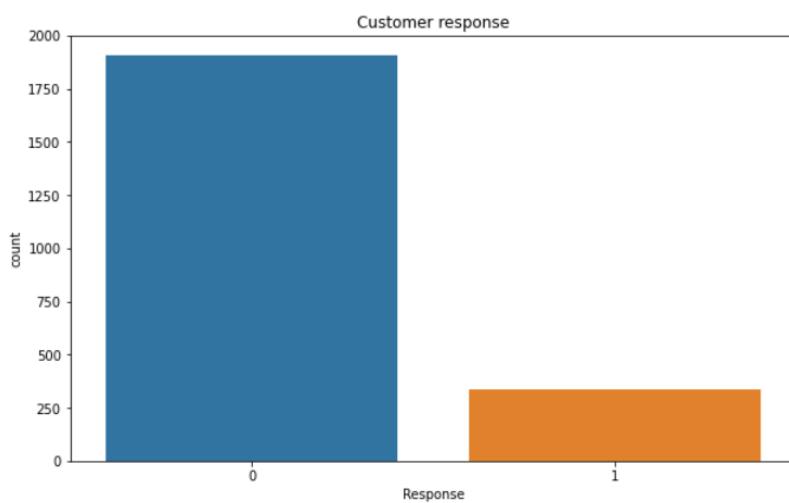
- ทำการดู จำนวน Customer ที่ Response โดย ใช้ฟังก์ชัน .value_counts() โดย 0 คือไม่รับโปรโมชั่น และ 1 คือตอบรับโปรโมชั่น

```
customer['Response'].value_counts()
```

```
0    1906
1    334
Name: Response, dtype: int64
```

- ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพธ์ แผนภูมิแท่งของ Response ได้ผลลัพธ์ ดังภาพด้านล่าง จะเห็นได้ว่า มีจำนวนลูกค้าไม่ตอบรับโปรโมชั่นมากกว่า จำนวนลูกค้าที่ตอบรับโปรโมชั่น

```
plt.figure(figsize=(10,6))
plt.title("Customer response")
sns.countplot(x="Response", data = customer)
plt.show()
```



3. ทำการคำนวณหาค่า rate_web โดยนำ feature Response ไปหารกับ NumWebVisitMonth ทำการดู ผลลัพธ์ โดย ใช้ฟังก์ชัน .value_counts()

```
[ rate_web = customer['Response'] / customer['NumWebVisitsMonth']
rate_web = pd.DataFrame(rate_web)

rate_web.rename(columns={0: 'Rate_web'}, inplace=True)
rate_web.value_counts()
```

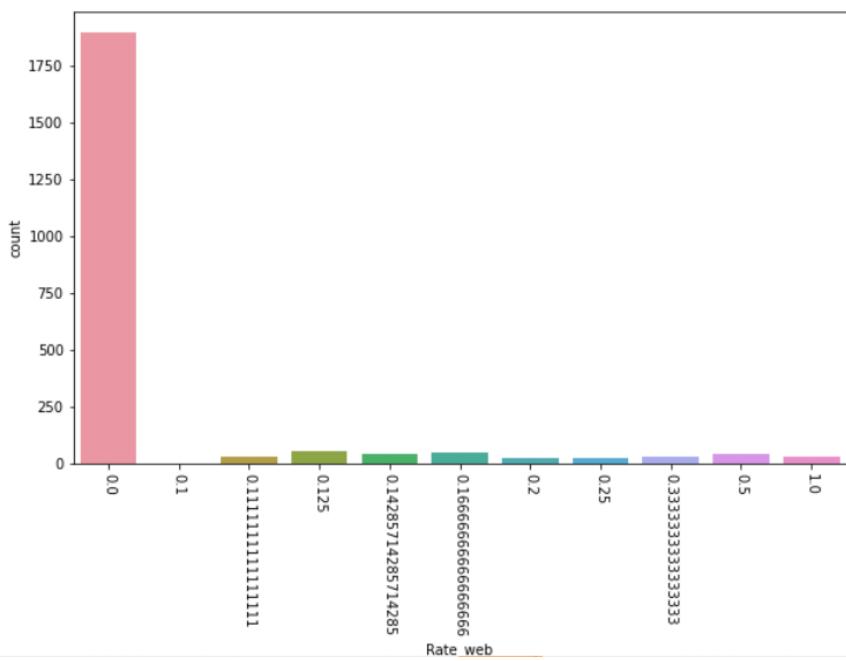
Rate_web	count
0.000000	1895
0.125000	57
0.166667	48
0.142857	45
0.500000	40
0.333333	33
1.000000	30
0.111111	29
0.250000	26
0.200000	25
0.100000	1

dtype: int64

4. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อพล็อต แผนภูมิแท่งของ rate_web ผลลัพธ์แสดงผลตามภาพด้านล่าง

```
# plot visitor rate (web) จากข้อมูลข้างบน
plt.figure(figsize=(10,6))
sns.countplot(x="Rate_web", data = rate_web)
plt.xticks(rotation=270)
```

(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]),
 <a list of 11 Text major ticklabel objects>)



5. เปลี่ยนข้อมูลใน feature Dt_Customer ให้เป็นวันที่ โดยใช้ฟังก์ชัน .to_datetime() ใน pandas และ print ค่า เพื่อดูวันที่ลูกค้าเป็นสมาชิก จากนั้นใช้ ฟังก์ชัน dt.year เพื่อนำค่าปีที่เป็นสมาชิก มาเก็บไว้ที่ตัวแปร since_year ทำให้ได้ทราบว่าลูกค้าเป็นสมาชิกตั้งแต่ปีที่เท่าไร

```
[ customer['Dt_Customer'] = pd.to_datetime(customer['Dt_Customer'])
print(customer['Dt_Customer'])

since_year = customer['Dt_Customer'].dt.year
print(since_year)

0      2012-04-09
1      2014-08-03
2      2013-08-21
3      2014-10-02
4      2014-01-19
...
2235    2013-06-13
2236    2014-10-06
2237    2014-01-25
2238    2014-01-24
2239    2012-10-15
Name: Dt_Customer, Length: 2240, dtype: datetime64[ns]
0      2012
1      2014
2      2013
3      2014
4      2014
...
2235    2013
2236    2014
2237    2014
2238    2014
2239    2012
Name: Dt_Customer, Length: 2240, dtype: int64
```

6. ทำการดู ปีที่ลูกค้าเริ่มเป็นสมาชิก โดย ใช้ฟังก์ชัน .value_counts() โดยลูกค้าเริ่มเป็นสมาชิก ตั้งแต่ปี 2012 มีจำนวน 494 คน , 2013 มีจำนวน 1189 คน และ 2014 มีจำนวน 557 คน

```
since_year.value_counts()
## plot since_year เป็นลูกค้าตั้งแต่ปีไป
```

ปี	จำนวน
2013	1189
2014	557
2012	494

Name: Dt_Customer, dtype: int64

7. นำ since_year ที่ได้มาใส่ Dataframe และเก็บในตัวแปร since_year1

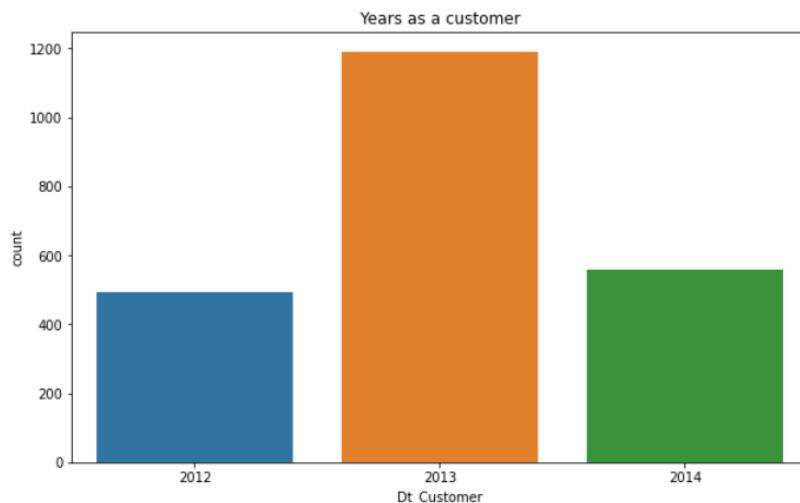
```
since_year1 = pd.DataFrame(since_year)
since_year1
```

Dt_Customer	
0	2012
1	2014
2	2013
3	2014
4	2014
...	...
2235	2013
2236	2014
2237	2014
2238	2014
2239	2012

2240 rows × 1 columns

8. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อพล็อต
แผนภูมิแท่งของ since_year1 ได้ผลลัพธ์ ดังภาพด้านล่าง จะเห็นได้ว่าปี ค.ศ. 2013 เป็นปีที่มีจำนวนลูกค้า
เป็นสมาชิกมากที่สุด

```
plt.figure(figsize=(10,6))
plt.title("Years as a customer")
sns.countplot(x='Dt_Customer', data = since_year1)
```



9. คำนวณอายุของลูกค้าตอนเป็นสมาชิก โดยนำ since_year มาลบกับ ข้อมูล Year_Birth จากนั้นใช้ฟังก์ชัน .describe() เพื่อดูค่าสถิติต่างๆ เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

```
customer['Age'] = since_year - customer['Year_Birth']
customer['Age'].describe()
```

```
count    2240.000000
mean     44.222321
std      12.022855
min     16.000000
25%    36.000000
50%    43.000000
75%    54.000000
max    121.000000
Name: Age, dtype: float64
```

10. ทำการดู จำนวนอายุของ Customer ตอนที่เป็นสมาชิก โดย ใช้ฟังก์ชัน .value_counts() ว่ามีจำนวนกี่คน โดยอายุของลูกค้าตอนที่เป็นสมาชิกที่มีจำนวนมากที่สุดคือ 41 ปี มีจำนวน 91 คน

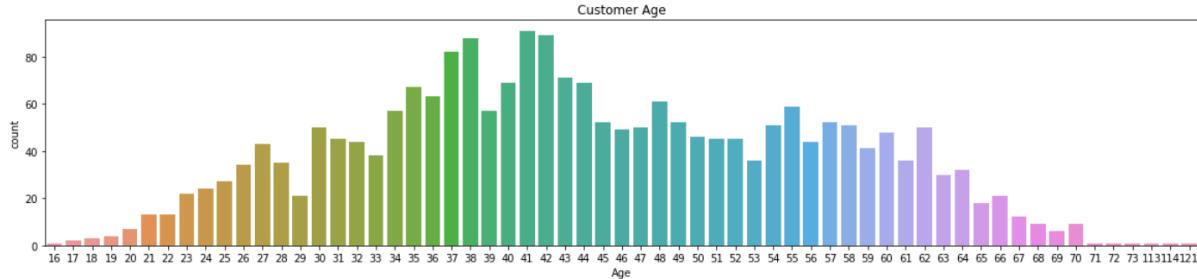
```
print(customer['Age'].value_counts())
```

```
41     91
42     89
38     88
37     82
43     71
..
114    1
113    1
71     1
73     1
16     1
Name: Age, Length: 61, dtype: int64
```

11. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพต์แผนภูมิแท่งของ อายุของลูกค้า จากภาพด้านล่าง แสดงให้เห็นถึง อายุของลูกค้าที่เป็นสมาชิกที่มีจำนวนมากที่สุดส่วนใหญ่จะอยู่ในช่วง 35 – 50 ปี

```
#ผลลัพธ์ของลูกค้าจาก customer['Age']
plt.figure(figsize=(20,4))
customer['Age']
sns.countplot(x='Age', data=customer)
plt.title("Customer Age")
```

Text(0.5, 1.0, 'Customer Age')



12. ประกาศตัวแปรอาเรย์ที่ชื่อว่า dates จากนั้นใช้ loop for เก็บข้อมูลวันที่ Dt_Customer ไปเก็บไว้ในตัวแปร dates แล้วทำการ print ค่าเพื่อดู วันที่ลูกค้าเป็นสมาชิกล่าสุด และวันที่ลูกค้าเป็นสมาชิกที่เก่าที่สุด

```
dates = []
for i in customer["Dt_Customer"]:
    i = i.date()
    dates.append(i)
print("The newest customer's enrolment date in therecords:", max(dates))
print("The oldest customer's enrolment date in the records:", min(dates))
```

The newest customer's enrolment date in therecords: 2014-12-06
The oldest customer's enrolment date in the records: 2012-01-08

13. หาจำนวนวันที่ลูกค้าเป็นสมาชิก โดยเก็บค่าใส่ตัวแปรอาเรย์ที่ชื่อว่า days จากนั้นเพิ่ม feature Customer_for ของcustomer โดยนำข้อมูลจาก days ไปใส่ใน Customer_for จากนั้นใช้ฟังก์ชัน dt.days เลือกเก็บแค่จำนวนวันที่ลูกค้าเป็นสมาชิก

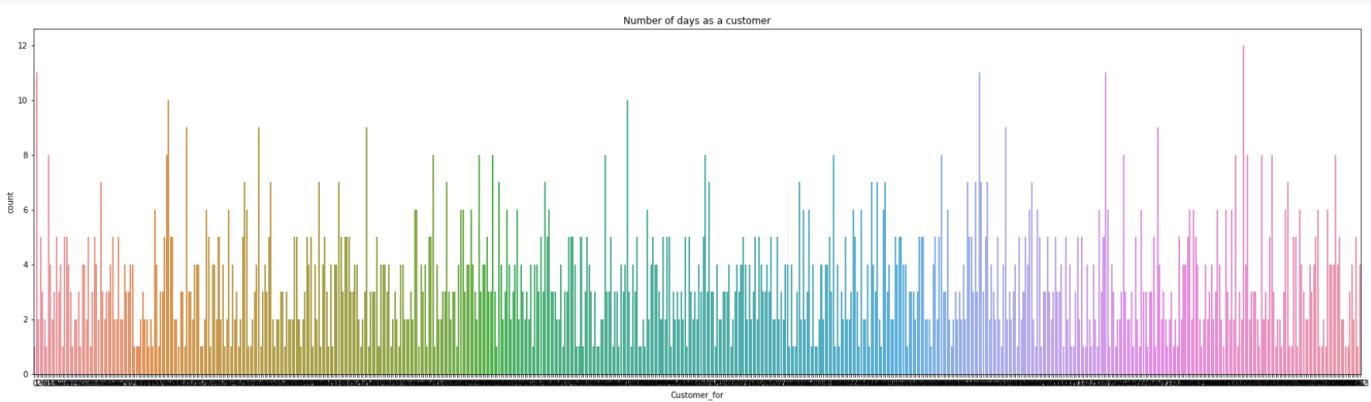
```
days = []
recent_day = max(dates)
for i in dates:
    delta = recent_day - i
    days.append(delta)

customer['Customer_for'] = days
customer['Customer_for'] = customer['Customer_for'].dt.days
customer['Customer_for']
```

```
0      971
1      125
2      472
3       65
4      321
...
2235     541
2236      61
2237     315
2238     316
2239     782
Name: Customer_for, Length: 2240, dtype: int64
```

14. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพธ์
แผนภูมิแท่งของ จำนวนวันที่ลูกค้าเป็นสมาชิก

```
plt.figure(figsize=(30, 8))
plt.title("Number of days as a customer")
sns.countplot(x="Customer_for", data = customer)
plt.show()
```



15. นำ feature kidhome และ Teenhome มารวมกันเป็น feature เดียวที่ชื่อว่า Children เพื่อ
แสดงถึงจำนวนเด็กภายในบ้านทั้งหมด และทำการ drop คอมลัมน์ kidhome และ Teenhome ทั้ง
จำนวนนี้ใช้ฟังก์ชัน value_counts() เพื่อถูくるจำนวนลูกค้าที่มีเด็กหรือบุตรอยู่ในบ้าน เป็นเท่าไรโดย ลูกค้าส่วน
ใหญ่มีจำนวนเด็กในบ้าน 1 คนสูงที่สุด

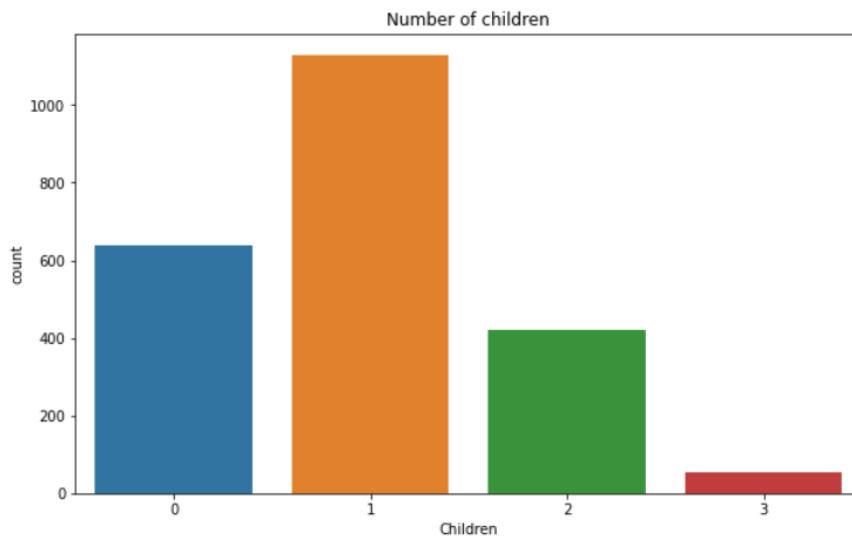
```
customer['Children'] = customer['Kidhome'] + customer['Teenhome']
customer.drop(['Kidhome', 'Teenhome'], axis='columns', inplace=True)
customer['Children'].value_counts()
```

```
1    1128
0     638
2     421
3      53
Name: Children, dtype: int64
```

16. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพธ์
แผนภูมิแท่งของ จำนวนเด็กภายในบ้าน จากภาพด้านล่าง แสดงให้เห็นถึง ลูกค้าที่มีบุตร 1 คนมีจำนวนมาก
ที่สุด และ ถึง ลูกค้าที่มีบุตร 3 คนมีจำนวนน้อยที่สุด

```
plt.figure(figsize=(10,6))
sns.countplot(x='Children', data=customer)
plt.title("Number of children")
```

Text(0.5, 1.0, 'Number of children')



17. ทำการรวม ข้อมูลที่ เป็น Basic และ 2n Cycle ใน feature Education แทนที่ ด้วย Undergraduate แทนที่ข้อมูลที่เป็น Graduation ด้วย Graduate และ แทนที่ข้อมูลที่เป็น Master และ PhD เป็น Postgraduate จากนั้นใช้ พงก์ชัน .value_counts() เพื่อดูค่า ของระดับการศึกษาต่างๆของลูกค้า

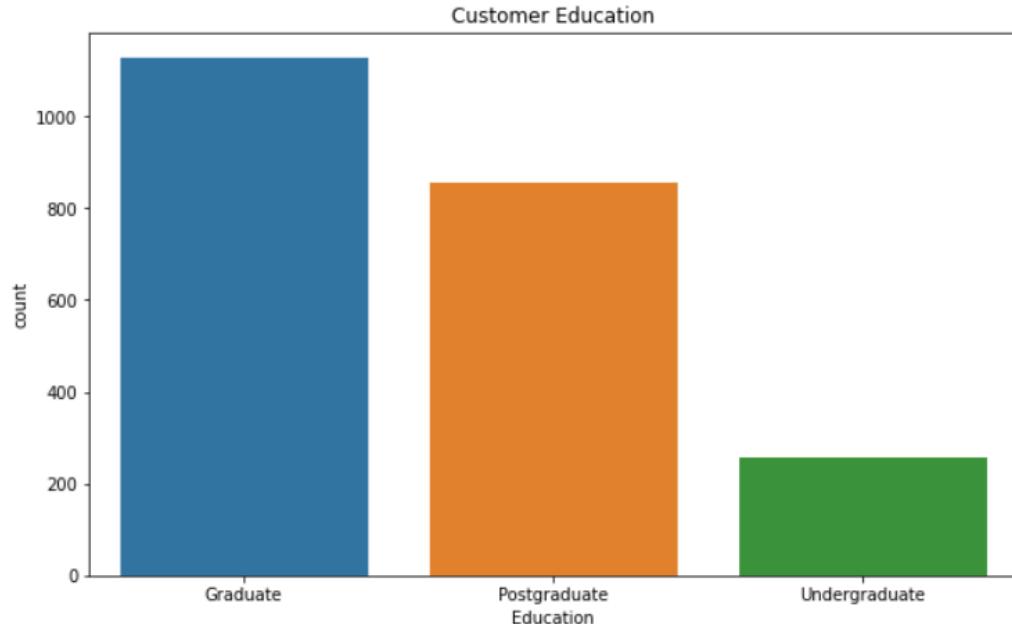
```
print(customer['Education'].value_counts())
customer['Education'] = customer['Education'].replace({"Basic":"Undergraduate","2n Cycle":"Undergraduate",
                                                       "Graduation":"Graduate", "Master":"Postgraduate", "PhD":"Postgraduate"})
print(customer['Education'].value_counts())
```

```
Graduation    1127
PhD          486
Master        370
2n Cycle     203
Basic         54
Name: Education, dtype: int64
Graduate     1127
Postgraduate  856
Undergraduate 257
Name: Education, dtype: int64
```

18. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพธ์
แผนภูมิแท่งของ ระดับการศึกษาของลูกค้า จากภาพด้านล่าง แสดงให้เห็นถึงระดับที่มีจำนวนลูกค้ามากที่สุด
คือ ระดับ Graduate รองลงมาคือ Postgraduate และ Undergraduate คืออันดับสุดท้าย

```
plt.figure(figsize=(10,6))
plt.title("Customer Education")
sns.countplot(x="Education", data = customer)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c71ab42d0>
```



19. ทำการรวม ข้อมูล ที่เป็น Divorced, Widow, Alone, YOLO, Absurd ใน feature Marital_Status รวมกันโดยการแทนที่เป็น Single และ ข้อมูลที่เป็น Married กับ Together รวมกันเป็น Relationship จากนั้นใช้ ฟังก์ชัน .value_counts() เพื่อดูค่า สถานภาพของลูกค้า

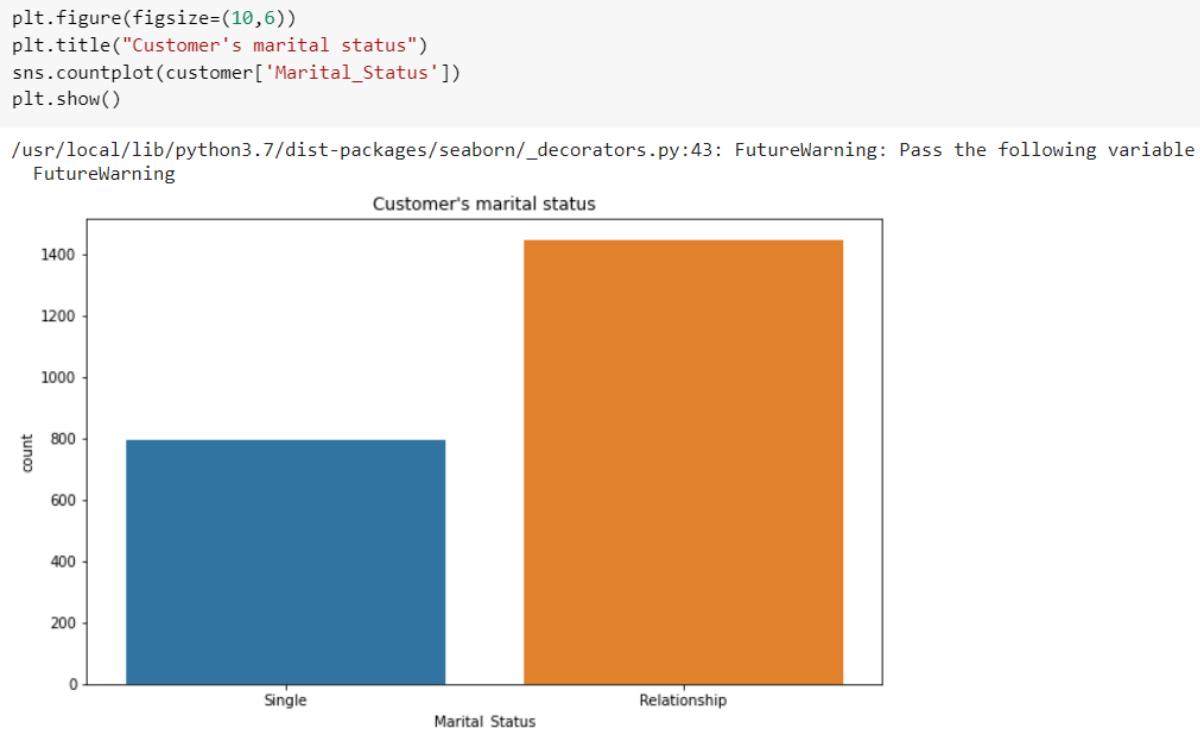
```
print(customer['Marital_Status'].value_counts())
```

```
customer['Marital_Status'] = customer['Marital_Status'].replace(['Divorced', 'Widow', 'Alone', 'YOLO', 'Absurd'], 'Single')
customer['Marital_Status'] = customer['Marital_Status'].replace(['Married', 'Together'], 'Relationship')
```

```
print(customer['Marital_Status'].value_counts())
```

```
Married      864
Together     580
Single       480
Divorced    232
Widow        77
Alone         3
YOLO          2
Absurd        2
Name: Marital_Status, dtype: int64
Relationship  1444
Single        796
Name: Marital_Status, dtype: int64
```

20. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อผลลัพธ์
แผนภูมิแท่งของ สถานภาพของลูกค้า จากภาพด้านล่าง แสดงให้เห็นว่า มีลูกค้าที่แต่งงาน มีจำนวนมากกว่า
จำนวนลูกค้าที่โสด



21. ใช้ ฟังก์ชัน .values() เพื่อนำค่าใน feature ต่างๆ ของ customer ไปเก็บไว้ในตัวแปรที่สร้าง
ขึ้นมา โดยตัวแปรที่สร้างขึ้นคือ wines, fruits, meat_prod, fish_prod, sweet_prod และ gold_prod

```
wines = customer["MntWines"].values
fruits = customer["MntFruits"].values
meat_prod = customer["MntMeatProducts"].values
fish_prod = customer["MntFishProducts"].values
sweet_prod = customer["MntSweetProducts"].values
gold_prod = customer["MntGoldProds"].values
```

22. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .plot() เพื่อผลลัพธ์ดังภาพ กราฟ ของค่า wines, fruits, meat_prod, fish_prod, sweet_prod และ gold_prod ได้ผลลัพธ์ดังภาพ ด้านล่าง

```

plt.subplots()
plt.plot(wines,color ='purple')
plt.title("Wine")

plt.subplots()
plt.plot(fruits,color ='blue')
plt.title("Fruits")

plt.subplots()
plt.plot(meat_prod,color ='green')
plt.title("Meat_prod")

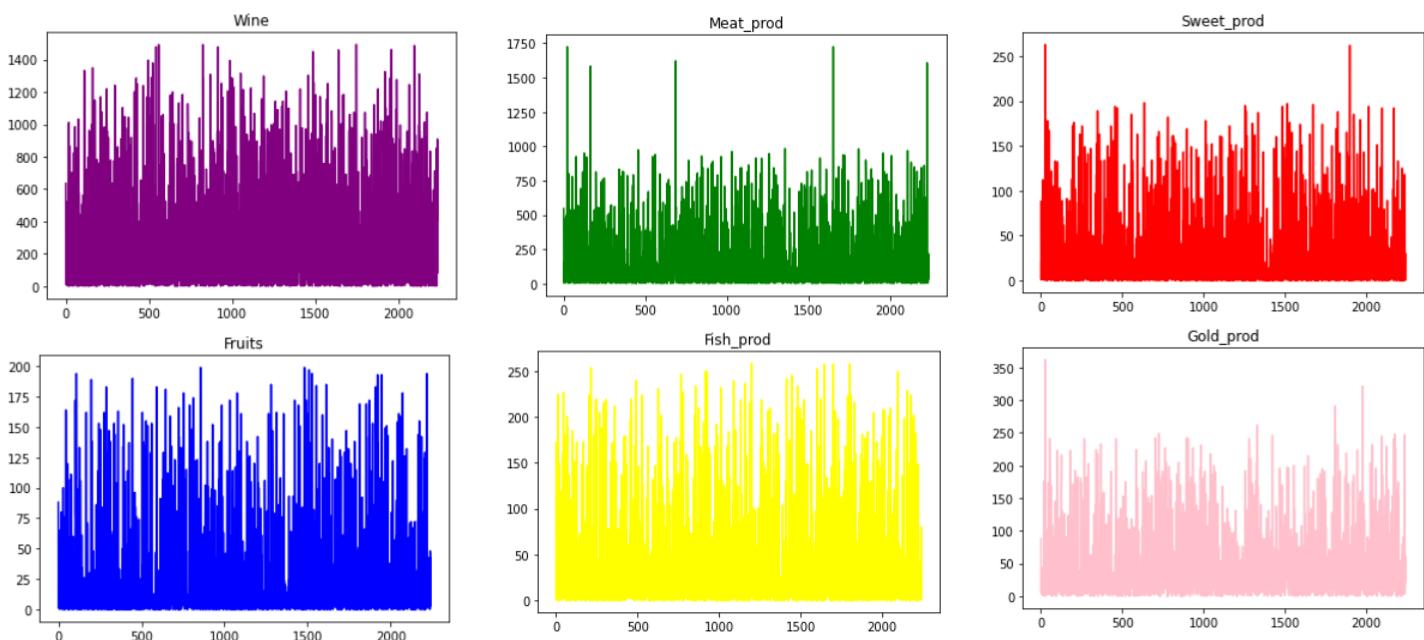
plt.subplots()
plt.plot(fish_prod,color ='yellow')
plt.title("Fish_prod")

plt.subplots()
plt.plot(sweet_prod,color ='red')
plt.title("Sweet_prod")

plt.subplots()
plt.plot(gold_prod,color ='pink')
plt.title("Gold_prod")

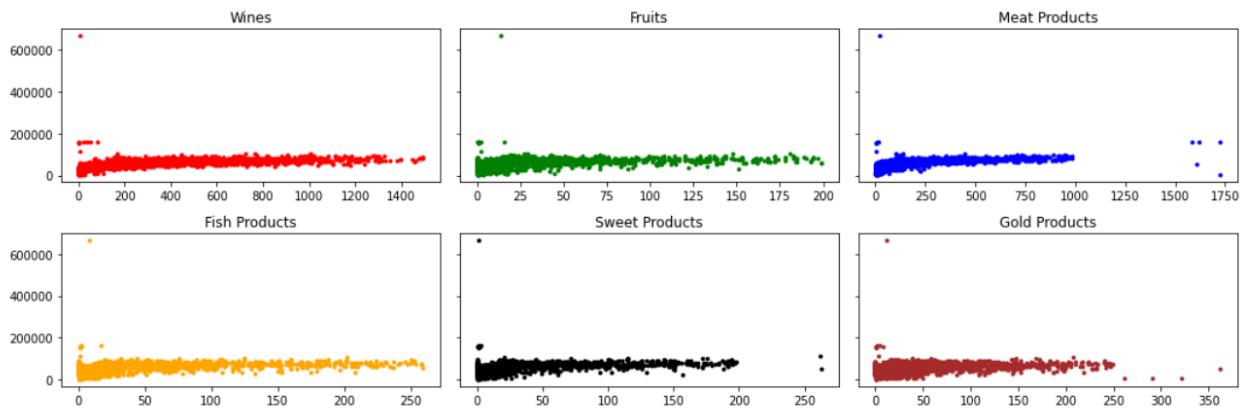
```

23. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .scatter() เพื่อ



ผลลัพธ์ดังภาพด้านล่าง
ได้ผลลัพธ์ดังภาพด้านล่าง

```
fig, axes = plt.subplots(2, 3, figsize=(15, 5), sharey=True)
axes[0,0].scatter(customer["MntWines"], customer["Income"] , label= "stars", color= "red",marker= ".", s=30)
axes[0,0].set_title('Wines')
axes[0,1].scatter(customer["MntFruits"], customer["Income"] , label= "stars", color= "green",marker= ".", s=30)
axes[0,1].set_title('Fruits')
axes[0,2].scatter(customer["MntMeatProducts"], customer["Income"] , label= "stars", color= "blue",marker= ".", s=30)
axes[0,2].set_title('Meat Products')
axes[1,0].scatter(customer["MntFishProducts"], customer["Income"] , label= "stars", color= "orange",marker= ".", s=30)
axes[1,0].set_title('Fish Products')
axes[1,1].scatter(customer["MntSweetProducts"], customer["Income"] , label= "stars", color= "black",marker= ".", s=30)
axes[1,1].set_title('Sweet Products')
axes[1,2].scatter(customer["MntGoldProds"], customer["Income"] , label= "stars", color= "brown",marker= ".", s=30)
axes[1,2].set_title('Gold Products')
fig.tight_layout()
```

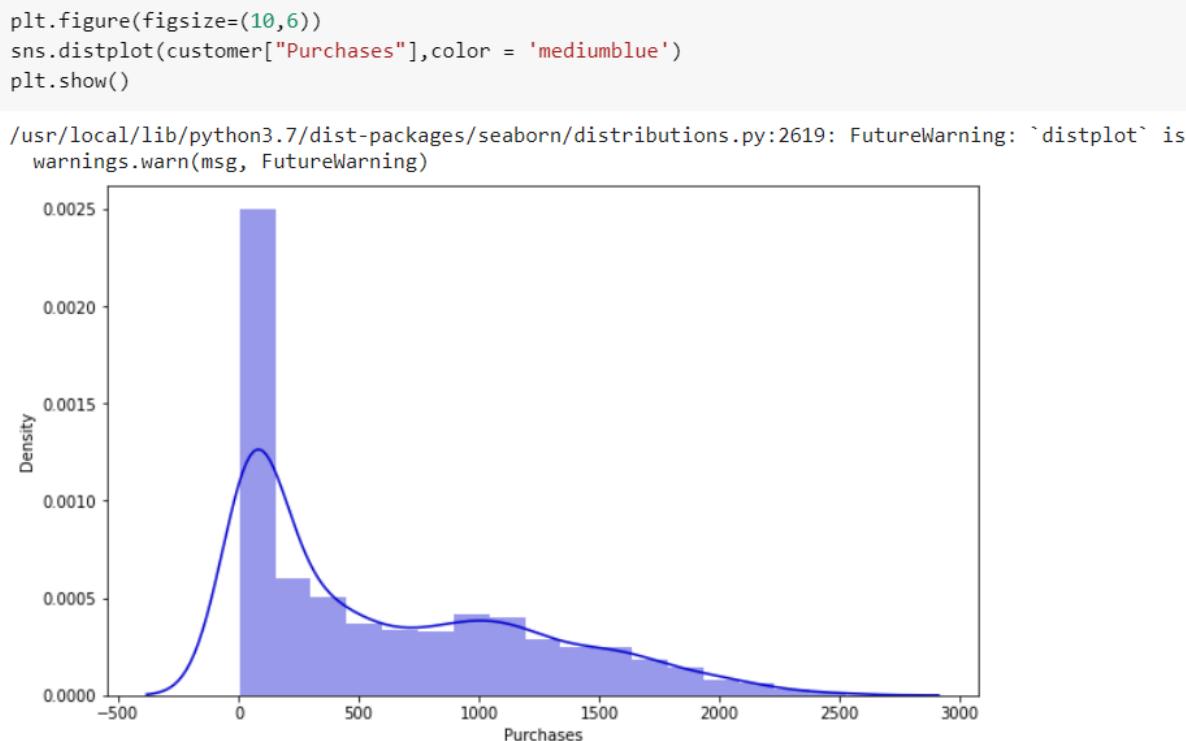


24. สร้าง feature Purchases โดยการรวมค่าของ feature MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, และ MntGoldProds เป็น feature ที่บอกจำนวนสินค้าทั้งหมดที่ลูกค้าซื้อ

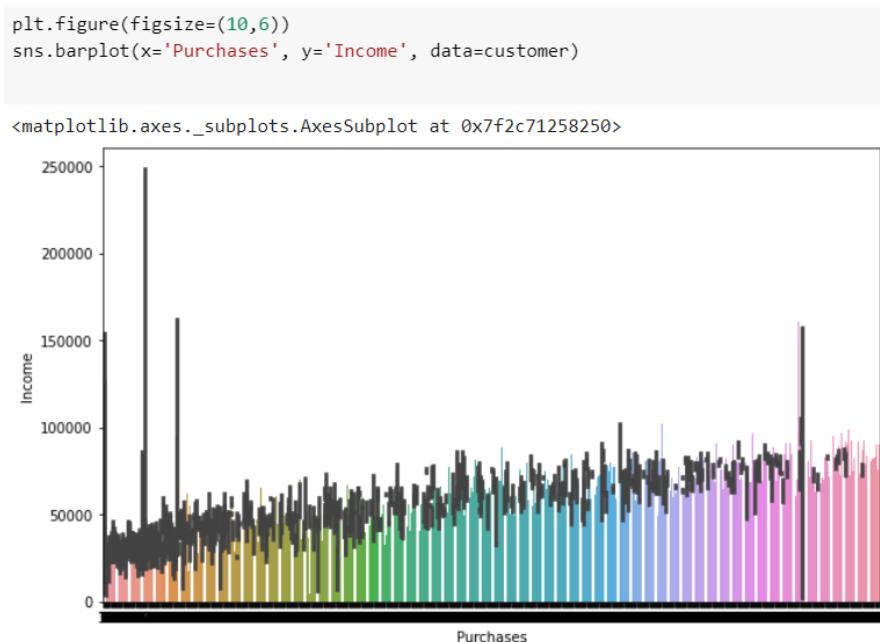
```
customer['Purchases'] = customer["MntWines"] + customer["MntFruits"] + customer["MntMeatProducts"]
+ customer["MntFishProducts"] + customer["MntSweetProducts"] + customer["MntGoldProds"]
customer['Purchases']

0      1617
1        27
2      776
3       53
4      422
...
2235    1341
2236     444
2237    1241
2238     843
2239     172
Name: Purchases, Length: 2240, dtype: int64
```

25. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี seaborn ใช้ฟังก์ชัน `.distplot()` เพื่อใช้สร้าง กราฟ histogram เป็นการประมาณของการกระจายของข้อมูล Purchases จำนวนการซื้อสินค้าต่างๆที่แสดงในเห็นในภาพด้านล่าง



26. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี seaborn ใช้ฟังก์ชัน `.barplot()` แสดงแผนภูมิแท่งค่าระหว่าง Purchases และ Income จากภาพด้านล่างแสดงให้เห็นว่า ลูกค้าที่มีรายได้สูงส่วนใหญ่จะซื้อของมากกว่าลูกค้าที่มีรายได้น้อยกว่า



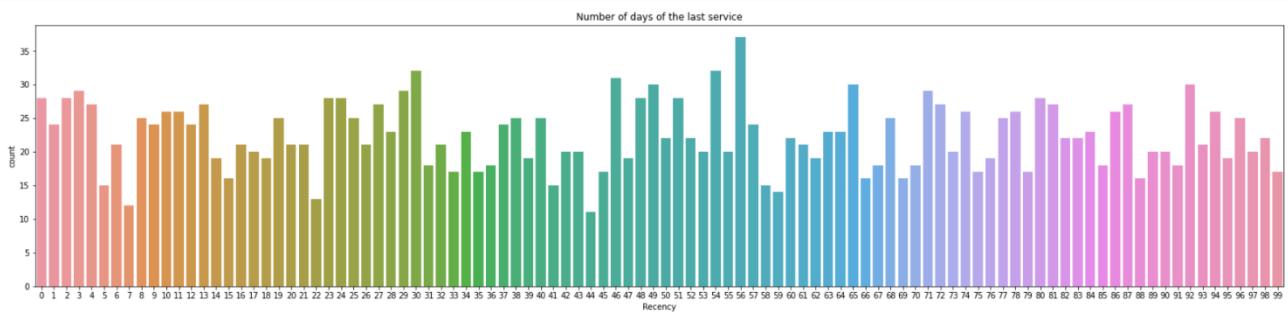
27. ใช้ฟังก์ชัน .describe() เพื่อดูค่าสถิติต่างๆ ของ feature Recency ของ customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

```
customer['Recency'].describe()

count    2240.000000
mean     49.109375
std      28.962453
min      0.000000
25%     24.000000
50%     49.000000
75%     74.000000
max     99.000000
Name: Recency, dtype: float64
```

28. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .countplot() เพื่อพื้นที่ แผนภูมิแท่ง ของ Recency หรือจำนวนวันที่ลูกค้ามาใช้บริการล่าสุด

```
plt.figure(figsize=(28, 6))
plt.title("Number of days of the last service")
sns.countplot(x="Recency", data = customer)
plt.show()
```



29. ทำการ print ค่า Income ของ customer ออกมาเพื่อดูรายได้ของลูกค้า แต่ละคน

```
print(customer['Income'])
```

```
0      58138.0
1      46344.0
2      71613.0
3      26646.0
4      58293.0
      ...
2235    61223.0
2236    64014.0
2237    56981.0
2238    69245.0
2239    52869.0
Name: Income, Length: 2240, dtype: float64
```

30. ใช้ฟังก์ชัน .describe() เพื่อดูค่าสถิติต่างๆ ของ feature Income ใน customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุด ในคอมลัมນ์

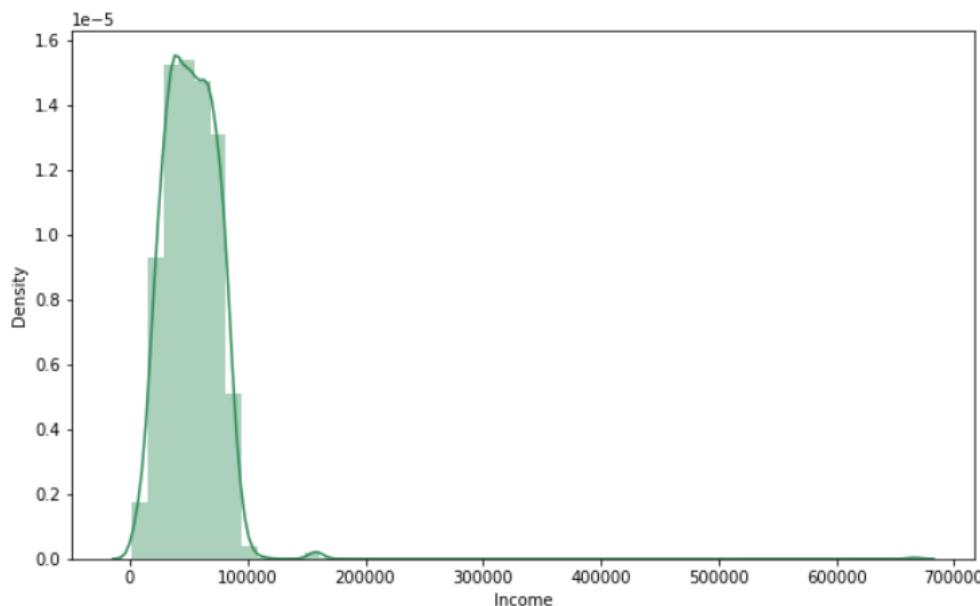
```
customer['Income'].describe()
```

count	2240.000000
mean	52237.975446
std	25037.955891
min	1730.000000
25%	35538.750000
50%	51381.500000
75%	68289.750000
max	666666.000000
Name:	Income, dtype: float64

31. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี seaborn ใช้ฟังก์ชัน .distplot() เพื่อใช้สร้าง กราฟ histogram เป็นการประมาณของการกระจายของข้อมูล รายได้ของลูกค้าแสดงผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
sns.distplot(customer["Income"], color = 'Seagreen')
plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot`  
warnings.warn(msg, FutureWarning)
```



32. ทำการสร้าง feature AcceptedCmp_total โดยการนำ ข้อมูลการตอบรับของ Campaign ต่างๆของลูกค้า และ ข้อมูลการ Response ใช้ฟังก์ชัน .value_counts() เก็บค่าใส่ตัวแปร total_cmp และทำการแสดงผลมีข้อมูลดังนี้ 0 คือไม่ตอบรับสัก Campaign, 1 ตอบรับ 1 Campaign, 2 ตอบรับ 2 Campaign, ไปจนถึง ตอบรับมากสุดคือ 5 Campaign ว่ามีจำนวนทั้งหมดกี่คน

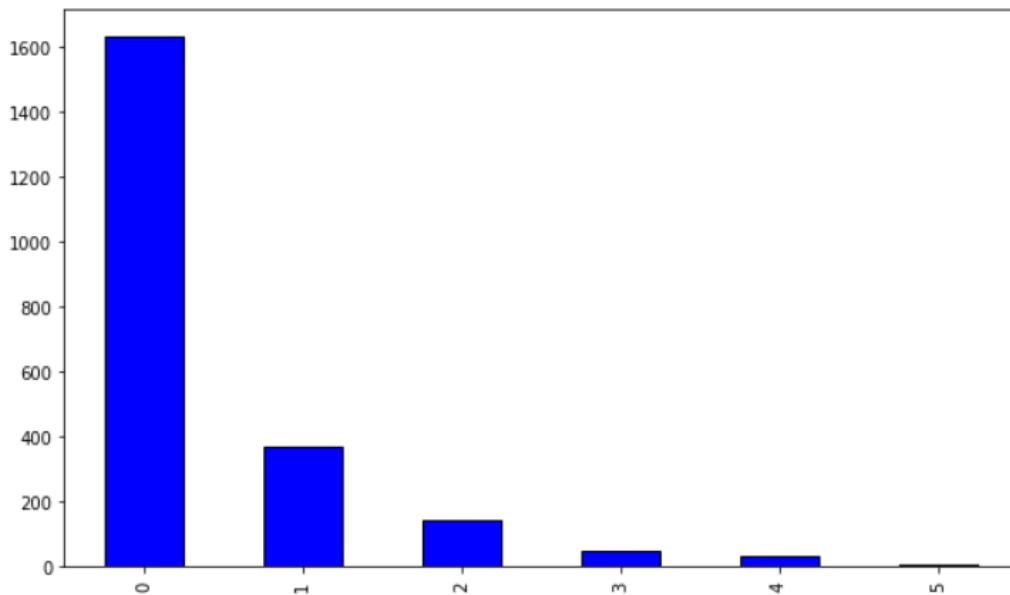
```
customer['AcceptedCmp_total'] = customer['AcceptedCmp1'] + customer['AcceptedCmp2'] + customer['AcceptedCmp3'] + customer['AcceptedCmp4'] + customer['AcceptedCmp5'] + customer['Response']
total_cmp = customer['AcceptedCmp_total'].value_counts()
total_cmp
## 0 = ไม่รับสักอัน, 1 = รับ 1, 2 = รับ 2 ...
0    1631
1     370
2     142
3      51
4      36
5      10
Name: AcceptedCmp_total, dtype: int64
```

33. ทำการนำข้อมูลมา visualize โดยใช้ฟังก์ชัน .plot() แสดงผลเป็น แผนภูมิแท่งแสดงการตอบรับของลูกค้า โดยลูกค้าส่วนใหญ่เลือกไม่ตอบรับสัก Campaign และ ตอบรับ 1, 2, 3, 4 และ 5 เรียงลำดับตามลงมา

```
## ผลลัพธ์ bar AcceptedCmp_total 4 แห่ง
plt.figure(figsize=(10,6))
total_cmp.plot(kind='bar', color = 'blue', edgecolor = "black", linewidth = 1)
plt.title("Frequency Of Each Category in the TotalAcceptedCmp Variable \n")
```

Text(0.5, 1.0, 'Frequency Of Each Category in the TotalAcceptedCmp Variable \n')

Frequency Of Each Category in the TotalAcceptedCmp Variable



34. ทำการสร้าง feature Purchases_total ใน customer โดยการนำข้อมูลของ feature NumWebPurchases, NumCatalogPurchases, NumStorePurchases, และ NumDealsPurchases มารวมกัน และแสดงค่าออมมาดู โดยใช้ฟังก์ชัน .value_counts()

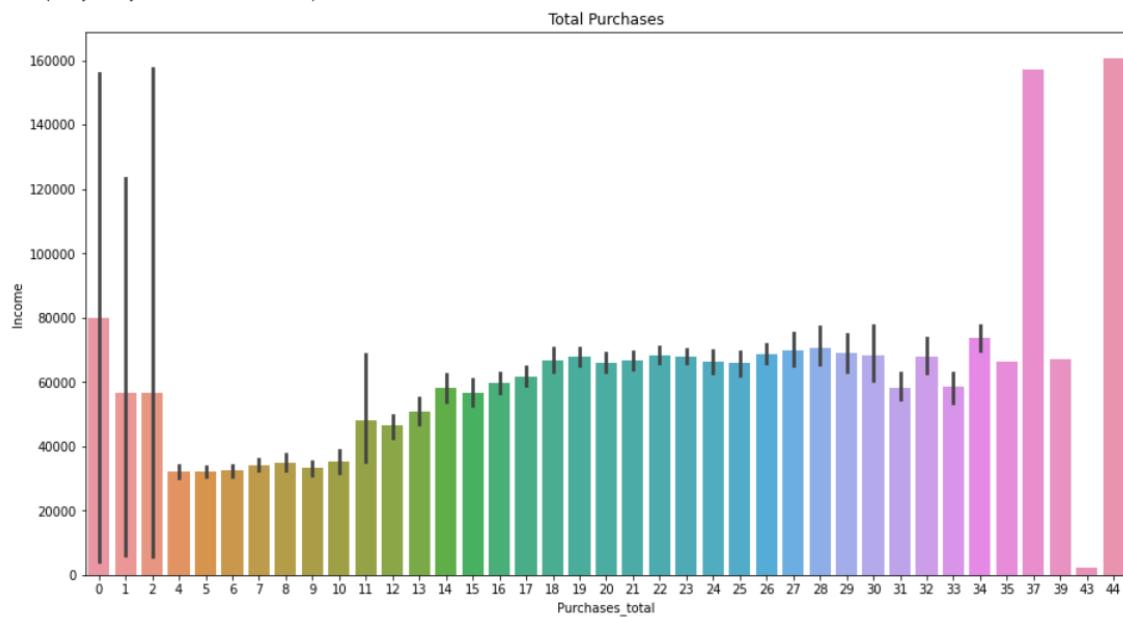
```
customer['Purchases_total'] = customer['NumWebPurchases'] + customer['NumCatalogPurchases'] + customer['NumStorePurchases'] + customer['NumDealsPurchases']
customer['Purchases_total'].value_counts()
```

```
7    149
5    145
4    128
6    123
17   116
9    102
16   101
19   101
21   95
8    94
20   94
22   94
23   87
10   80
18   79
15   74
12   70
25   68
26   67
11   67
24   56
14   55
13   44
27   39
28   35
29   19
32   12
```

35. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี seaborn ใช้ฟังก์ชัน .barplot() แสดงแผนภูมิแท่งค่าระหว่าง Purchase_total และ Income ได้ผลลัพธ์แสดงตามภาพด้านล่าง

```
plt.figure(figsize=(15,8))
sns.barplot(x='Purchases_total', y='Income', data=customer)
plt.title("Total Purchases")
```

```
Text(0.5, 1.0, 'Total Purchases')
```



36. ใช้ฟังก์ชัน `.describe()` เพื่อดูค่าสถิติต่างๆ ของ feature `Purchases_total` ใน customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

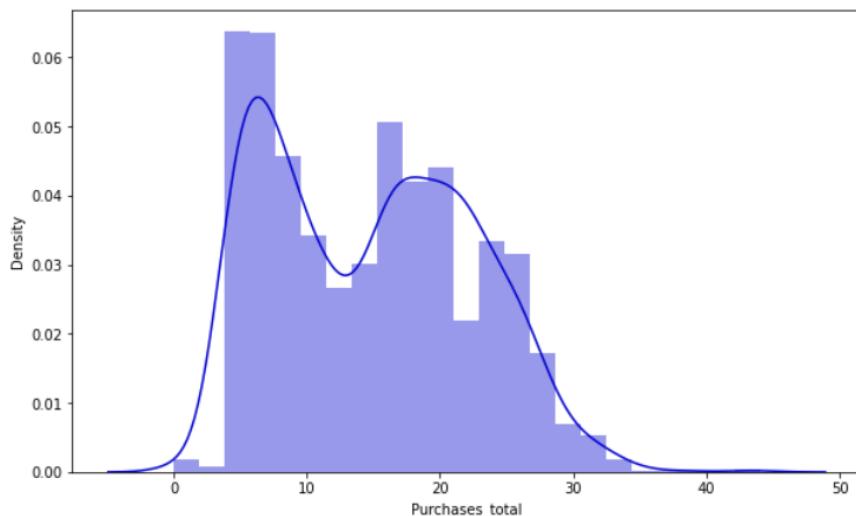
```
customer['Purchases_total'].describe()

count    2240.00000
mean     14.862054
std      7.677173
min      0.000000
25%     8.000000
50%    15.000000
75%    21.000000
max     44.000000
Name: Purchases_total, dtype: float64
```

37. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน `.distplot()` เพื่อใช้สร้าง กราฟ histogram เป็นการประมาณของการกระจายของข้อมูล ของจำนวนการซื้อสินค้ารวมทั้งหมดของลูกค้า

```
plt.figure(figsize=(10,6))
sns.distplot(customer["Purchases_total"], color = 'mediumblue')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is
warnings.warn(msg, FutureWarning)
```



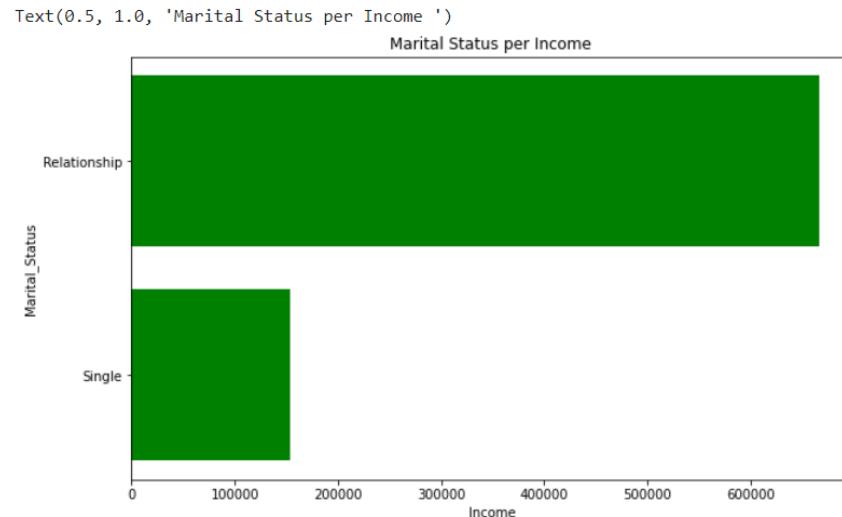
38. ทำการสร้าง feature `Avg_Purchase` โดยการนำ feature `Purchases` มาหาร feature `Purchases_total` จากนั้นใน ฟังก์ชัน `round()` เพื่อปัดเลขให้เป็นจำนวนเต็ม และใช้ฟังก์ชัน `.describe()` เพื่อแสดงค่าทางสถิติ

```
customer['Avg_Purchase'] = round((customer['Purchases'] / customer['Purchases_total']), 1)
customer['Avg_Purchase'].describe()
```

```
count    2240.000
mean      inf
std       NaN
min      0.500
25%     9.700
50%    23.450
75%    45.525
max      inf
Name: Avg_Purchase, dtype: float64
```

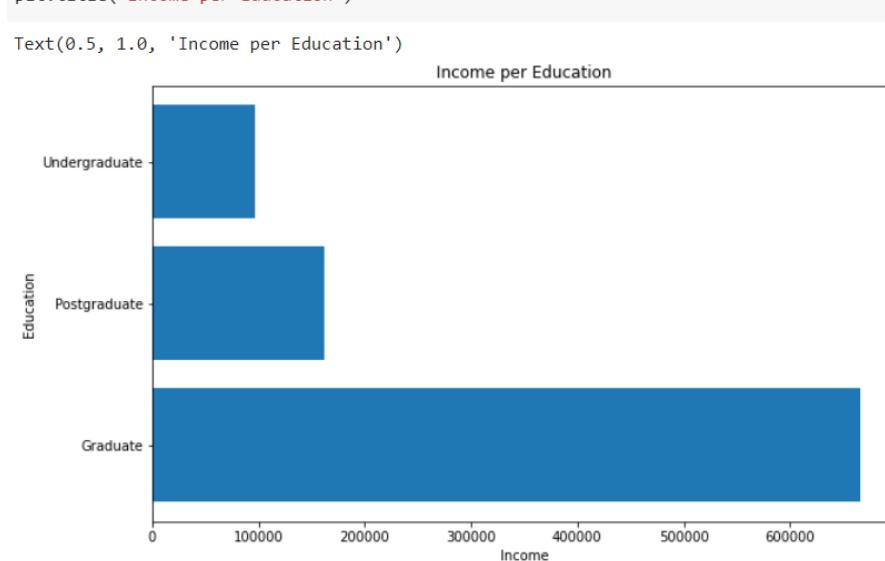
39. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงผลระหว่าง Income และ Marital Status จากภาพด้านล่างแสดงให้เห็นว่า ลูกค้าที่มี Relationship มีรายได้มากกว่า ลูกค้าที่ Single

```
## plot Income VS ทุก feature
plt.figure(figsize=(10,6))
plt.barh(customer["Marital_Status"],customer["Income"],color = 'green')
plt.xlabel("Income")
plt.ylabel("Marital_Status")
plt.title("Marital Status per Income ")
```

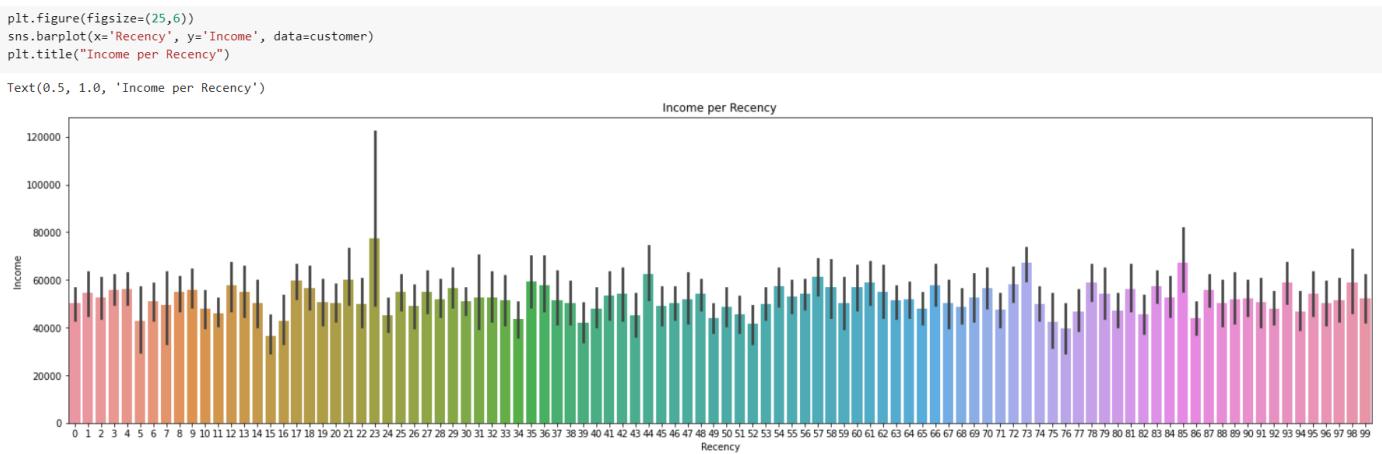


40. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Income และ Education ได้ผลลัพธ์ตามภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer["Education"],customer["Income"])
plt.xlabel("Income")
plt.ylabel("Education")
plt.title("Income per Education")
```



41. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .barplot() และแสดงแผนภูมิแท่งค่าระหว่าง Recency และ Income ได้ผลลัพธ์แสดงดังภาพด้านล่าง



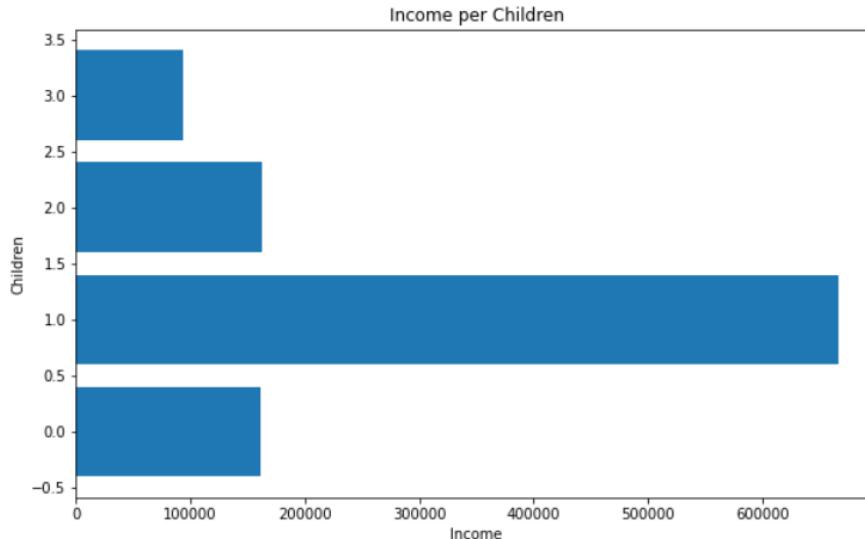
42. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .scatter() เพื่อพื้นที่ต่อของ Customer_for (จำนวนวันที่เป็นลูกค้า) และ Income ได้ผลลัพธ์แสดงดังภาพด้านล่าง



43. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Income และ Children ได้ผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer["Children"],customer["Income"])
plt.xlabel("Income")
plt.ylabel("Children")
plt.title("Income per Children")
```

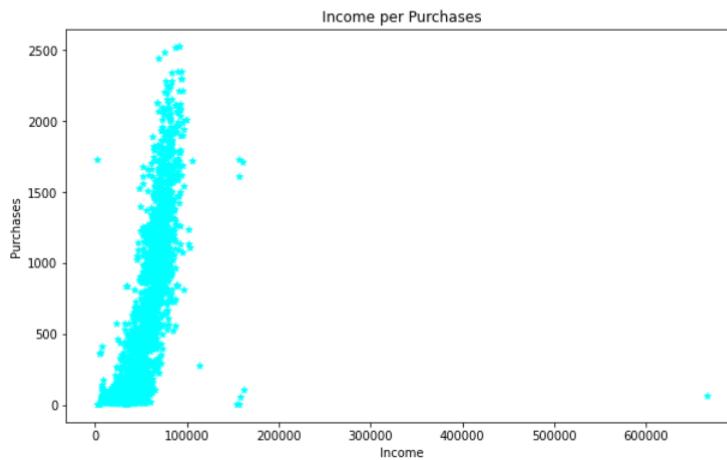
Text(0.5, 1.0, 'Income per Children')



44. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .scatter() เพื่อพิจารณาค่าของ Purchases และ Income ได้ผลลัพธ์แสดงดังภาพด้านล่าง

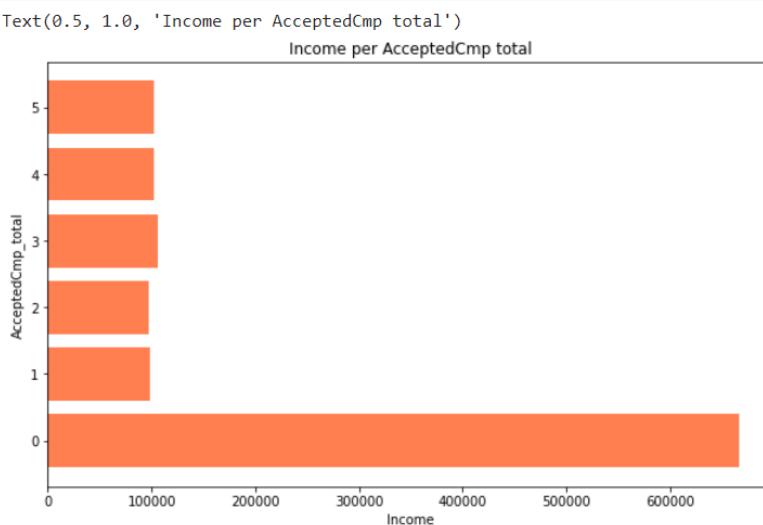
```
plt.figure(figsize=(10,6))
plt.scatter(customer["Income"],customer["Purchases"], label= "stars", color= "cyan",marker= "*", s=30)
plt.xlabel("Income")
plt.ylabel("Purchases")
plt.title("Income per Purchases")
```

Text(0.5, 1.0, 'Income per Purchases')



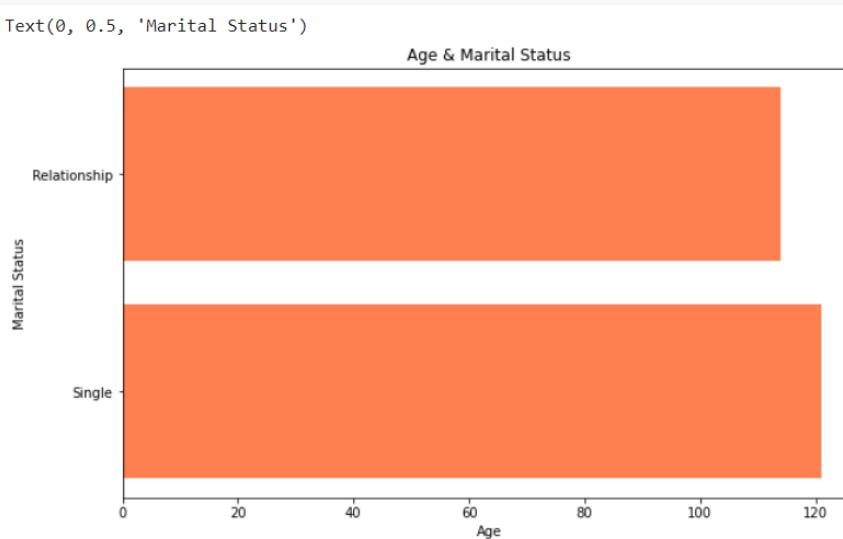
45. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดง แผนภูมิแท่งระหว่าง Income และ AcceptedCmp_total (จำนวน campaign ที่ลูกค้าตอบรับ) ได้ผลลัพธ์ ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer["AcceptedCmp_total"],customer["Income"],color ='coral')
plt.xlabel("Income")
plt.ylabel("AcceptedCmp_total")
plt.title("Income per AcceptedCmp total")
```



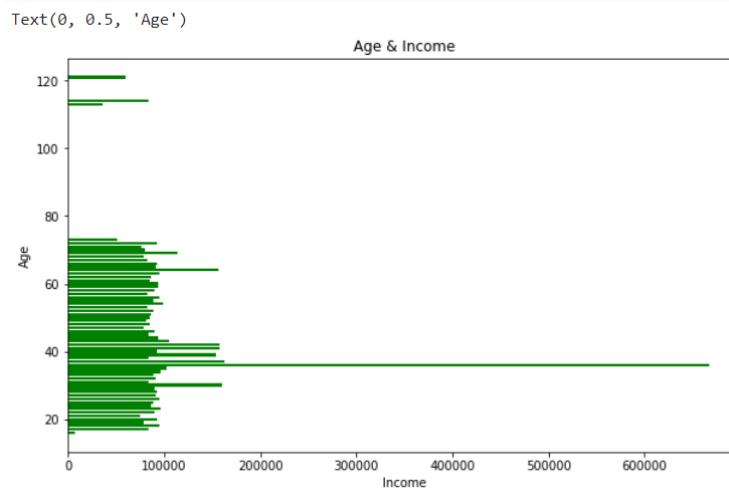
46. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดง แผนภูมิแท่งระหว่าง Age และ Marital Status ได้ผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer['Marital_Status'], customer['Age'],color ='coral')
plt.title('Age & Marital Status')
plt.xlabel('Age')
plt.ylabel('Marital Status')
```



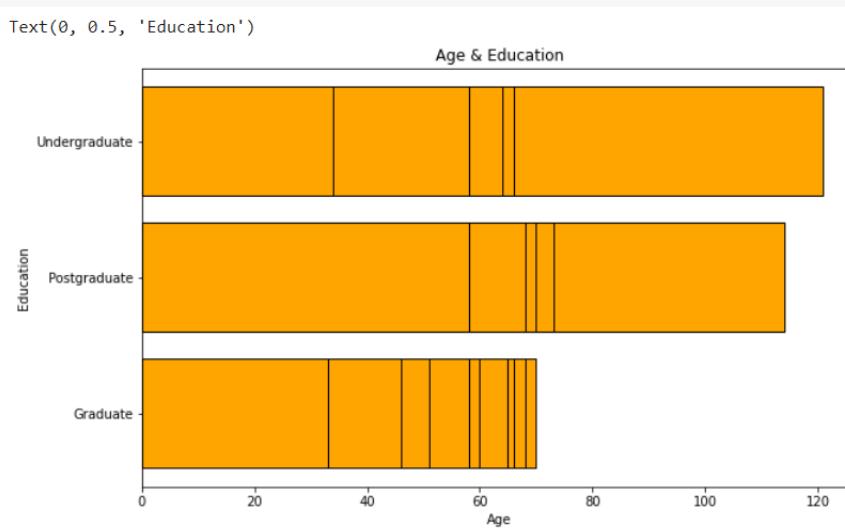
47. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Age และ Income ได้ผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer['Age'], customer['Income'], color = 'green')
plt.title('Age & Income')
plt.xlabel('Income')
plt.ylabel('Age')
```



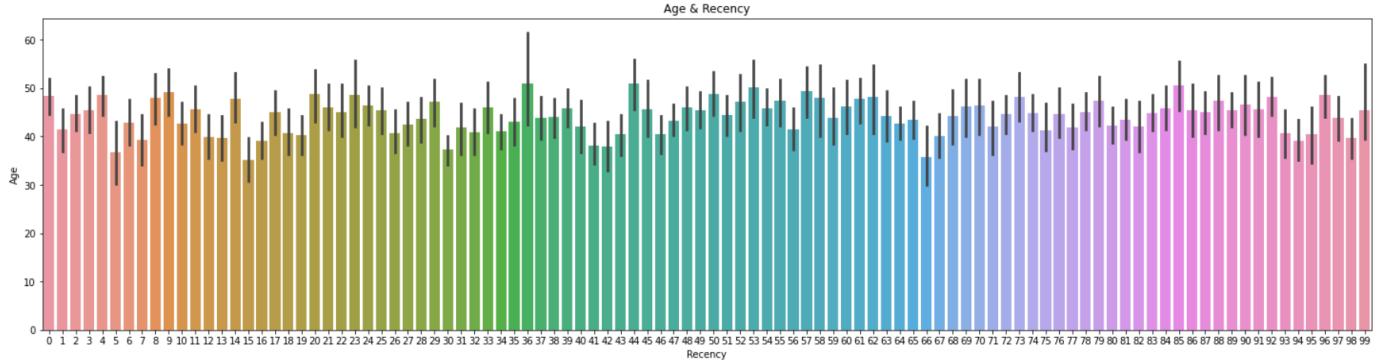
48. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Age และ Education ได้ผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.barh(customer['Education'], customer['Age'], color="orange")
plt.title('Age & Education')
plt.xlabel('Age')
plt.ylabel('Education')
```



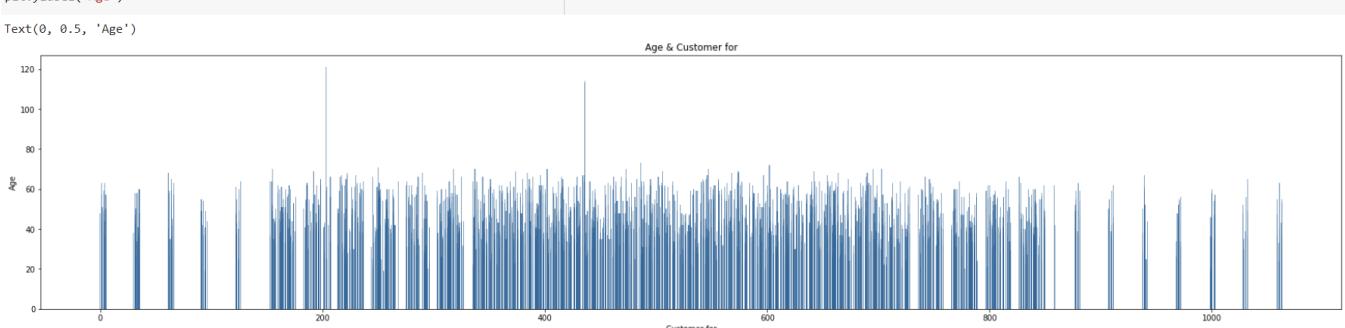
49. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .barplot() แสดงแผนภูมิแท่งค่าระหว่าง Age และ Recency ได้ผลลัพธ์แสดงดังภาพด้านล่าง

```
plt.figure(figsize=(25,6))
sns.barplot(x='Recency', y='Age', data=customer)
plt.title('Age & Recency')
Text(0.5, 1.0, 'Age & Recency')
```



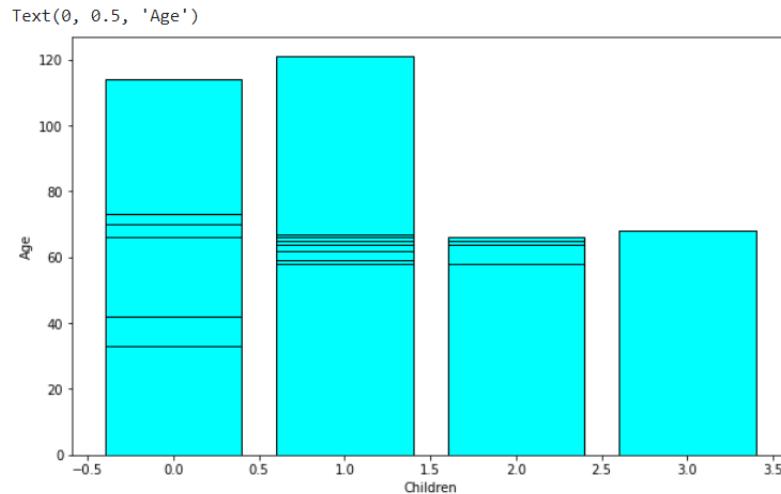
50. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .bar() แสดงแผนภูมิแท่งค่าระหว่าง Age และ Customer_for(จำนวนวันที่เป็นลูกค้า) ได้ผลลัพธ์แสดงดังภาพด้านล่าง

```
plt.figure(figsize=(30,6))
plt.bar(customer['Customer_for'], customer['Age'], color=(0.2, 0.4, 0.6, 0.6))
plt.title('Age & Customer for')
plt.xlabel('Customer for')
plt.ylabel('Age')
Text(0, 0.5, 'Age')
```



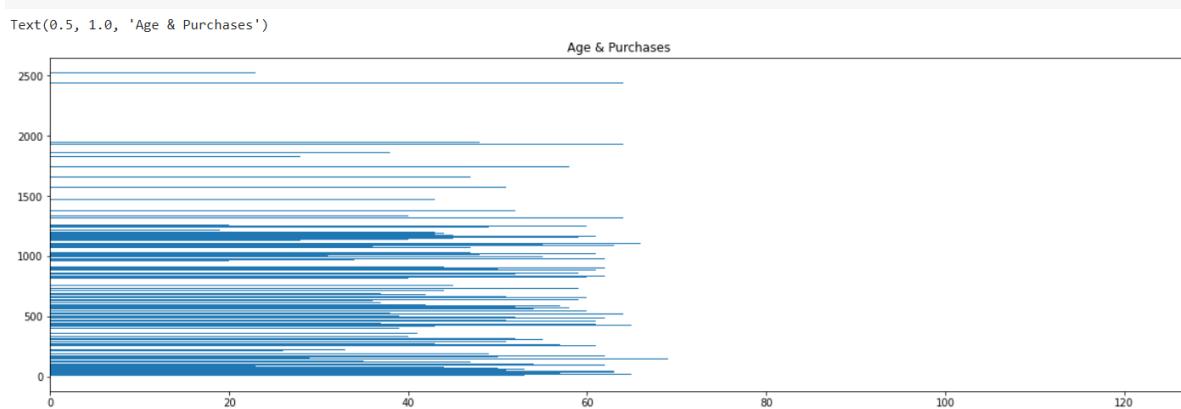
51. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .bar() เพื่อแสดงแผนภูมิแท่งระหว่าง Age และ Children ได้ผลลัพธ์ดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
plt.bar(customer['Children'], customer['Age'], color='cyan')
plt.xlabel('Children')
plt.ylabel('Age')
```



52. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Age และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

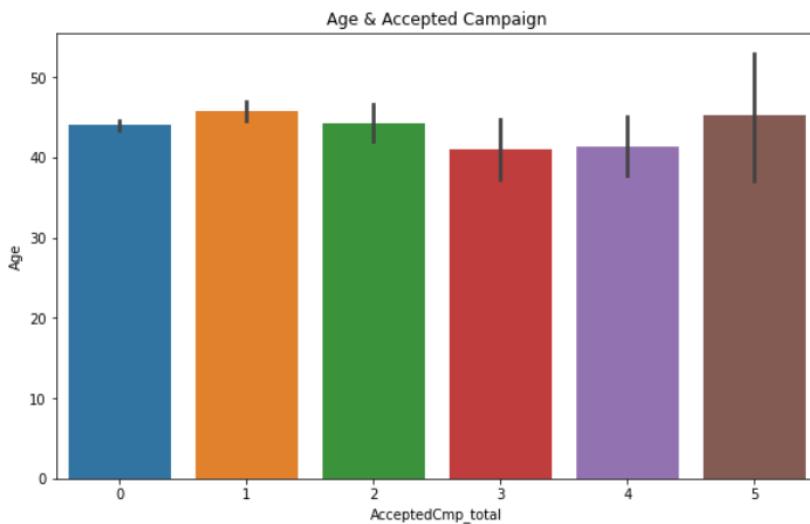
```
plt.figure(figsize=(20, 6))
plt.barh(customer['Purchases'], customer['Age'])
plt.title('Age & Purchases')
```



53. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .barplot() แสดงแผนภูมิแท่งค่าระหว่าง Age และ Accepted campaign ได้ผลลัพธ์แสดงดังภาพด้านล่าง

```
plt.figure(figsize=(10,6))
sns.barplot(x='AcceptedCmp_total', y='Age', data=customer)
plt.title('Age & Accepted Campaign')
```

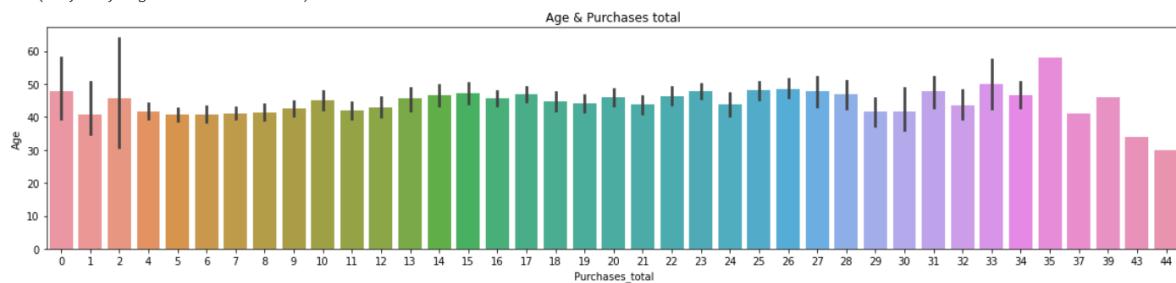
Text(0.5, 1.0, 'Age & Accepted Campaign')



54. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .barplot() แสดงแผนภูมิแท่งค่าระหว่าง Age และ Purchases total ได้ผลลัพธ์แสดงดังภาพด้านล่าง

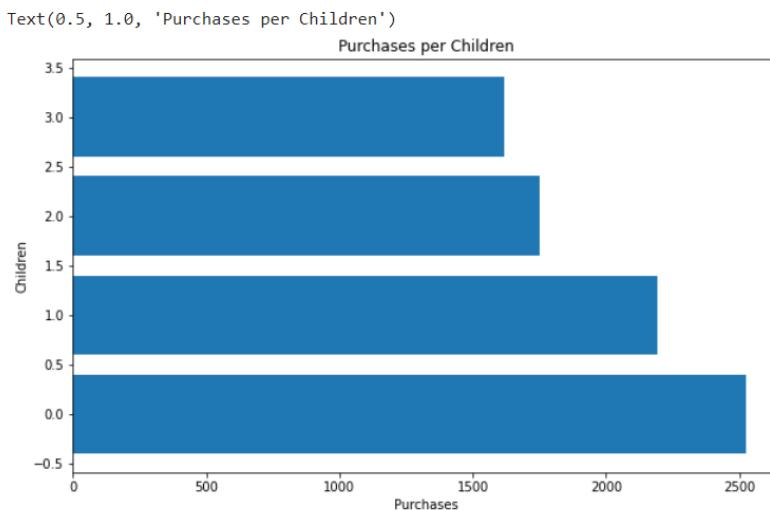
```
plt.figure(figsize=(20,4))
sns.barplot(x='Purchases_total', y='Age', data=customer)
plt.xticks()
plt.title('Age & Purchases total')
```

Text(0.5, 1.0, 'Age & Purchases total')



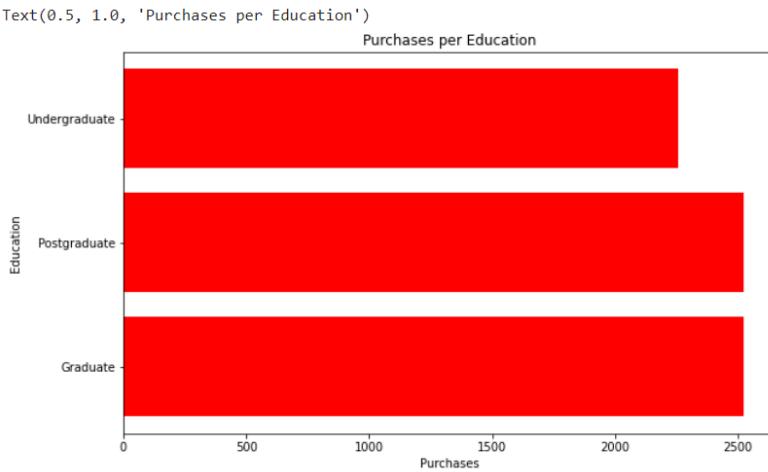
55. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Children และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases VS Children
plt.figure(figsize=(10,6))
plt.barh(customer["Children"],customer["Purchases"])
plt.xlabel("Purchases")
plt.ylabel("Children")
plt.title("Purchases per Children")
```



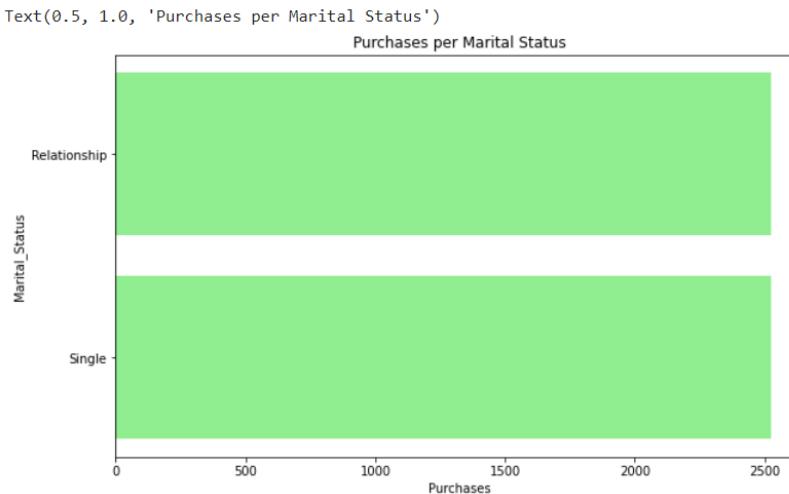
56. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Education และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases VS Education
plt.figure(figsize=(10,6))
plt.barh(customer["Education"],customer["Purchases"],color = 'red')
plt.xlabel("Purchases")
plt.ylabel("Education")
plt.title("Purchases per Education")
```



57. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง Marital Status และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases VS Marital_Status
plt.figure(figsize=(10,6))
plt.barh(customer["Marital_Status"],customer["Purchases"],color ='lightgreen')
plt.xlabel("Purchases")
plt.ylabel("Marital_Status")
plt.title("Purchases per Marital Status")
```



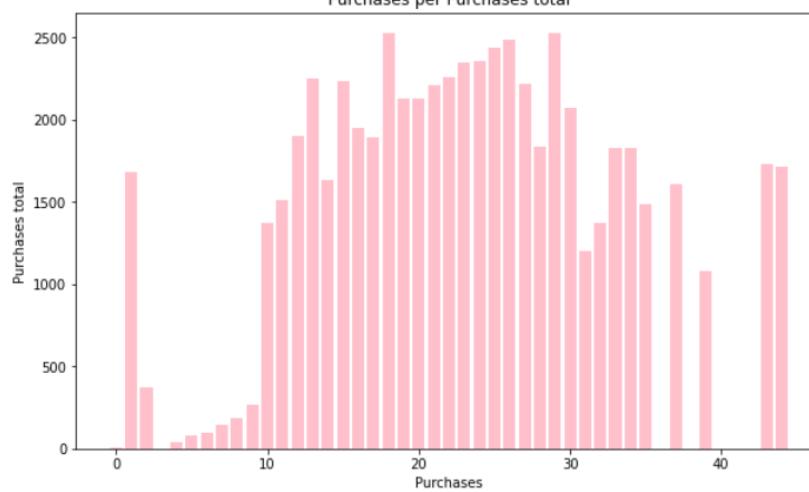
58. ทำการนำข้อมูลมา visualize โดยใช้ ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .barh() เพื่อแสดงแผนภูมิแท่งระหว่าง AcceptedCmp_total (การตอบรับ campaign ทั้งหมด) และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases VS AcceptedCmp_total
plt.figure(figsize=(10,6))
plt.barh(customer["AcceptedCmp_total"],customer["Purchases"],color ='yellow')
plt.xlabel("Purchases")
plt.ylabel("AcceptedCmp_total")
plt.title("Purchases per AcceptedCmp total")
```



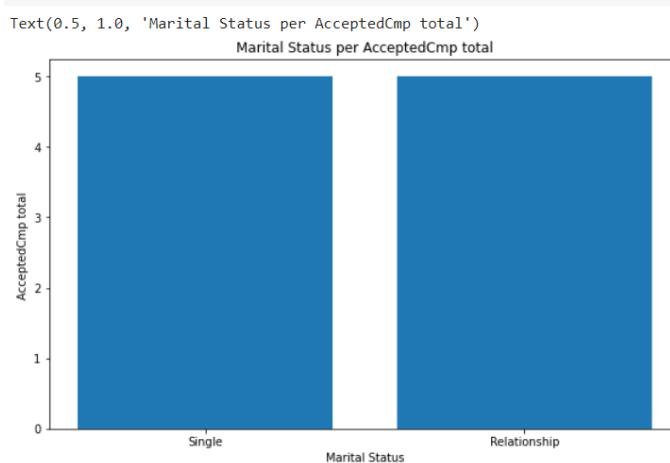
59. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .bar() เพื่อแสดงแผนภูมิแท่งระหว่าง Purchases total และ Purchases ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases VS Purchases_total
plt.figure(figsize=(10,6))
plt.bar(customer["Purchases_total"],customer["Purchases"],color = 'pink')
plt.xlabel("Purchases")
plt.ylabel("Purchases total")
plt.title("Purchases per Purchases total")
```



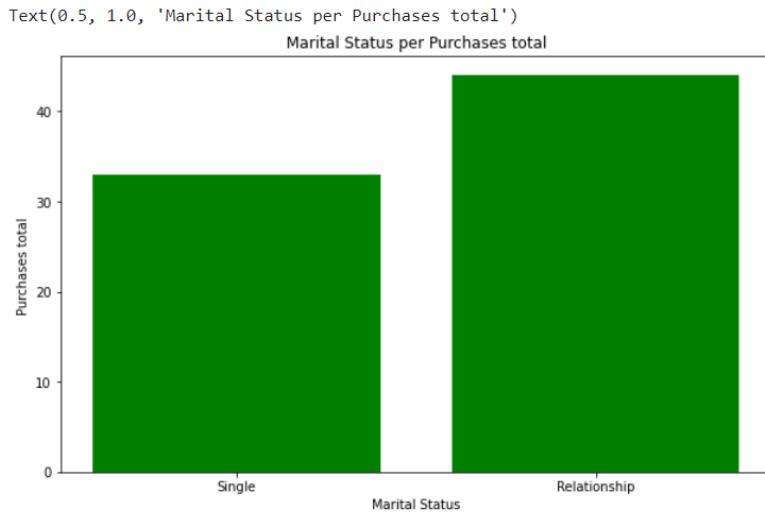
60. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .bar() เพื่อแสดงแผนภูมิแท่งระหว่าง Marital Status และ AcceptedCmp_total ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot AcceptedCmp_total VS Marital_Status
plt.figure(figsize=(10,6))
plt.bar(customer["Marital_Status"],customer["AcceptedCmp_total"])
plt.xlabel("Marital Status")
plt.ylabel("AcceptedCmp total ")
plt.title("Marital Status per AcceptedCmp total")
```



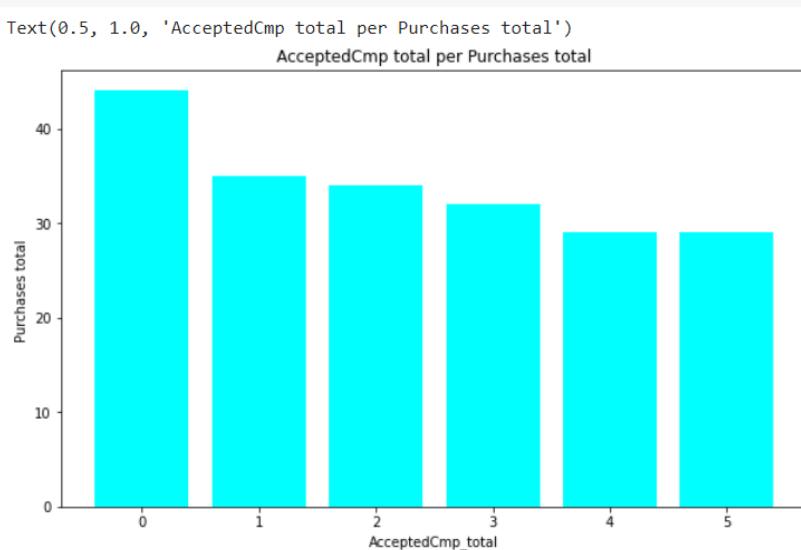
61. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .bar() เพื่อแสดงแผนภูมิแท่งระหว่าง Marital Status และ Purchases total ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot Purchases_total VS Marital_Status
plt.figure(figsize=(10,6))
plt.bar(customer["Marital_Status"],customer["Purchases_total"],color='green')
plt.xlabel("Marital Status")
plt.ylabel("Purchases total ")
plt.title("Marital Status per Purchases total")
```



62. ทำการนำข้อมูลมา visualize โดยใช้ไลบรารี matplotlib.pyplot ใช้ฟังก์ชัน .bar() เพื่อแสดงแผนภูมิแท่งระหว่าง Purchases total และ AcceptedCmp total ได้ผลลัพธ์ดังภาพด้านล่าง

```
## plot AcceptedCmp_total VS Purchases_total
plt.figure(figsize=(10,6))
plt.bar(customer["AcceptedCmp_total"],customer["Purchases_total"],color='cyan')
plt.xlabel("AcceptedCmp_total")
plt.ylabel("Purchases total ")
plt.title("AcceptedCmp total per Purchases total")
```



63. ใช้ฟังก์ชัน `.groupby()` จัดกลุ่ม feature Education กับ feature ต่างๆ แล้วนำฟังก์ชัน `.describe()` เพื่อดูค่าสถิติต่างๆ ใน customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

```
customer.groupby('Education').describe()
```

Education	ID									Year_Birth											
	count			mean			std			min			25%			50%			75%		
	Graduate	1127.0	5652.523514	3302.806089	0.0	2820.50	5544.0	8627.0	11191.0	1127.0	1969.635315	11.501761	1944.0	1960.0	1970.0	1978.0	1995.0				
Postgraduate	856.0	5525.970794	3179.153962	9.0	2903.25	5343.0	8146.5	11181.0	856.0	1966.404206	11.735394	1899.0	1957.0	1968.0	1975.0	1992.0					
Undergraduate	257.0	5547.910506	3228.305746	20.0	2625.00	5332.0	8370.0	11187.0	257.0	1973.167315	13.147274	1893.0	1965.0	1975.0	1981.0	1996.0					

64. ใช้ฟังก์ชัน `.groupby()` จัดกลุ่ม feature Age กับ feature ต่างๆ แล้วนำฟังก์ชัน `.describe()` เพื่อดูค่าสถิติต่างๆ ใน customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

```
customer.groupby('Age').describe()
```

Age	ID									Year_Birth											
	count			mean			std			min			25%			50%			75%		
	16	1.0	9909.000000	NaN	9909.0	9909.0	9909.0	9909.0	9909.0	1.0	1996.000000	NaN	1996.0	1996.0	1996.0	1996.0	1996.0	1996.0			
17	2.0	4044.000000	541.643794	3661.0	3852.5	4044.0	4235.50	4427.0	2.0	1995.000000	0.000000	1995.0	1995.0	1995.0	1995.0	1995.0	1995.0				
18	3.0	5905.666667	5284.351364	193.0	3549.0	6905.0	8762.00	10619.0	3.0	1994.666667	1.154701	1994.0	1994.0	1994.0	1995.0	1996.0	1996.0				
19	4.0	7945.250000	2204.552165	5184.0	7096.5	8024.5	8873.25	10548.0	4.0	1994.500000	1.000000	1993.0	1994.5	1995.0	1995.0	1995.0	1995.0				
20	7.0	4421.285714	2368.161996	821.0	3469.0	4483.0	5073.50	8560.0	7.0	1992.571429	0.786796	1992.0	1992.0	1992.0	1993.0	1994.0	1994.0				
...				
72	1.0	6932.000000	NaN	6932.0	6932.0	6932.0	6932.0	6932.0	1.0	1941.000000	NaN	1941.0	1941.0	1941.0	1941.0	1941.0	1941.0				
73	1.0	6663.000000	NaN	6663.0	6663.0	6663.0	6663.0	6663.0	1.0	1940.000000	NaN	1940.0	1940.0	1940.0	1940.0	1940.0	1940.0				
113	1.0	7829.000000	NaN	7829.0	7829.0	7829.0	7829.0	7829.0	1.0	1900.000000	NaN	1900.0	1900.0	1900.0	1900.0	1900.0	1900.0				
114	1.0	1150.000000	NaN	1150.0	1150.0	1150.0	1150.0	1150.0	1.0	1899.000000	NaN	1899.0	1899.0	1899.0	1899.0	1899.0	1899.0				
121	1.0	11004.000000	NaN	11004.0	11004.0	11004.0	11004.0	11004.0	1.0	1893.000000	NaN	1893.0	1893.0	1893.0	1893.0	1893.0	1893.0				

65. ใช้ฟังก์ชัน `.groupby()` จัดกลุ่ม feature Marital Status กับ feature ต่างๆ แล้วนำฟังก์ชัน `.describe()` เพื่อดูค่าสถิติต่างๆ ใน customer เช่น ค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน ค่าที่น้อยที่สุด และมากที่สุดในคอมลัมນ์

```
customer.groupby('Marital_Status').describe()
```

Marital_Status	ID									Year_Birth											
	count			mean			std			min			25%			50%			75%		
	Relationship	1444.0	5637.780471	3243.258885	0.0	2834.75	5528.0	8465.25	11188.0	1444.0	1968.843490	11.621682	1899.0	1959.0	1970.0	1977.0	1996.0				
Single	796.0	5509.400754	3253.233279	1.0	2801.75	5312.0	8358.25	11191.0	796.0	1968.737437	12.622139	1893.0	1959.0	1970.0	1977.0	1995.0					

ขั้นตอนที่ 2: Data Preprocessing & Data Representation

1. Feature Selection (การเลือกฟีเจอร์ที่ใช้)

ทำการ drop ที่ไม่จำเป็นต้องใช้ออก โดยใช้ฟังก์ชัน .drop() และเก็บ feature ที่จะนำไปใช้ทั้งหมด ใส่ในตัวแปร customer_final จากนั้นแสดงค่าออกมาเพื่อคุณลักษณะ โดยจะเหลือ feature ที่ใช้ทั้งหมด 9 features

```
drop_col = ['ID', 'Year_Birth', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue', 'MntWines',
           'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
           'MntGoldProducts', 'NumDealsPurchases', 'NumWebPurchases',
           'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
           'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Recency', 'Avg_Purchase', 'Response', 'Complain']
customer_final = customer.drop(drop_col, axis = 1)
customer_final
```

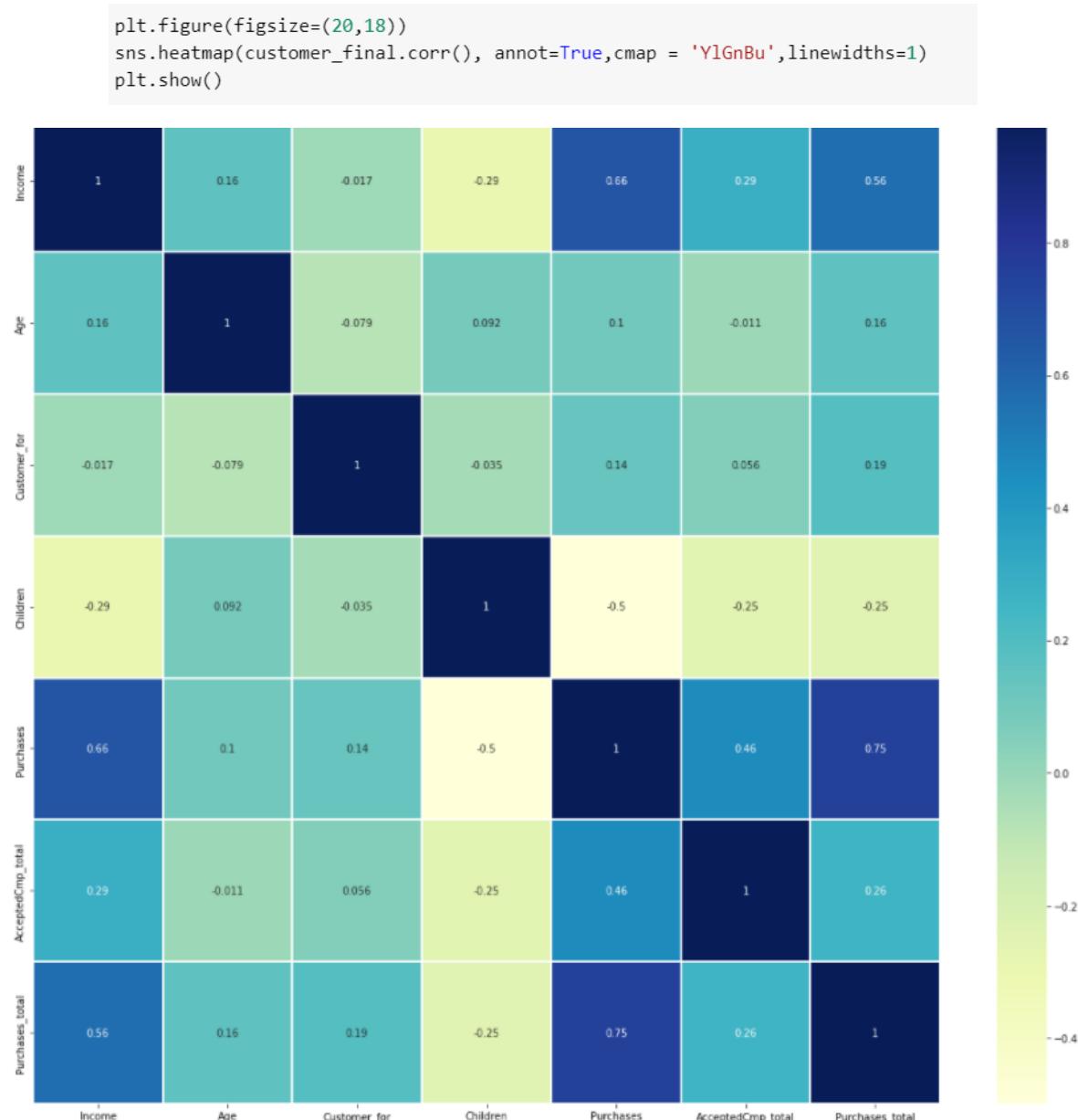
	Education	Marital_Status	Income	Age	Customer_for	Children	Purchases	AcceptedCmp_total	Purchases_total
0	Graduate	Single	58138.0	55	971	0	1617	1	25
1	Graduate	Single	46344.0	60	125	2	27	0	6
2	Graduate	Relationship	71613.0	48	472	0	776	0	21
3	Graduate	Relationship	26646.0	30	65	1	53	0	8
4	Postgraduate	Relationship	58293.0	33	321	1	422	0	19
...
2235	Graduate	Relationship	61223.0	46	541	1	1341	0	18
2236	Postgraduate	Relationship	64014.0	68	61	3	444	1	22
2237	Graduate	Single	56981.0	33	315	0	1241	1	19
2238	Postgraduate	Relationship	69245.0	58	316	1	843	0	23
2239	Postgraduate	Relationship	52869.0	58	782	2	172	1	11

ดังนั้นชุดข้อมูลที่ใช้ในการทำ Modeling จึงถูกแปลง成 2240 instances ประกอบไปด้วย 9 features ดังนี้

1. Education คือ ระดับการศึกษาของลูกค้าประกอบด้วย Undergraduate, Graduate, Postgraduate
2. Marital_Status คือ สถานภาพการสมรสประกอบด้วย Single, Relationship
3. Income คือ รายได้ของลูกค้า
4. Age คือ อายุของลูกค้า
5. Customer_for คือ ระยะเวลาที่ลูกค้าเป็นสมาชิก (วัน)
6. Children คือ จำนวนบุตรหลาน (คน)

7. Purchases คือ จำนวนการซื้อสินค้าต่อเดือนประกอบไปด้วย ไวน์, ผลไม้, ผลิตภัณฑ์จากเนื้อสัตว์, ผลิตภัณฑ์จากปลา, ผลิตภัณฑ์จากน้ำตาล, ห้องคำ
8. AcceptedCmp_total คือ จำนวนครั้งในการเข้าร่วมกิจกรรม แคมเปญหรือโปรโมชันจากทั้งหมด 5 ครั้ง
9. Purchases_total คือ จำนวนครั้งในการซื้อสินค้าผ่านช่องทางต่างๆได้แก่ ข้อเสนอ เว็บไซต์ แคตตาล็อก ร้านค้า

นำชุดข้อมูลมา visualize เพื่อแสดงความสัมพันธ์ของแต่ละฟีเจอร์โดยใช้ ไลบรารี seaborn ใช้ฟังก์ชัน .heatmap() ใช้นำเสนอข้อมูลแนวโน้มหรือความสัมพันธ์แบบง่าย ๆ โดยใช้ คู่สี และความเข้มข้นของสีแทนปริมาณหรือความถี่ ของ customer_final



ใช้ฟังก์ชัน .info() เพื่อดูข้อมูล information อาย่างคร่าวๆ ของ customer_final และทำการเช็คว่า customer_final มี missing value หรือไม่ โดยใช้ ฟังก์ชัน .isnull() โดยทุก feature ไม่มีค่า Missing value

customer_final.info()			customer_final.isnull().any()		
#	Column	Non-Null Count	Dtype	Education	False
0	Education	2240 non-null	object	Marital_Status	False
1	Marital_Status	2240 non-null	object	Income	False
2	Income	2240 non-null	float64	Age	False
3	Age	2240 non-null	int64	Customer_for	False
4	Customer_for	2240 non-null	int64	Children	False
5	Children	2240 non-null	int64	Purchases	False
6	Purchases	2240 non-null	int64	AcceptedCmp_total	False
7	AcceptedCmp_total	2240 non-null	int64	Purchases_total	False
8	Purchases_total	2240 non-null	int64	dtype: bool	

dtypes: float64(1), int64(6), object(2)
memory usage: 157.6+ KB

2. Outlier Detecting & Removal (การตรวจหาค่าผิดปกติและกำจัดออก)

```
customer_final['Income'] = np.where(customer_final['Income'] > 600000, 600000, customer_final['Income'])
customer_final['Age'] = np.where(customer_final['Age'] > 90, 90, customer_final['Age'])
```

3. Engineering & Transformation Features (การจัดการและการแปลงฟีเจอร์)

จากรูปจะเห็นได้ว่าฟีเจอร์ Education, Marital_Status เป็น categorical features ทำการแปลงให้เป็น continuous features ด้วย LabelEncoder() และทำการแปลงฟีเจอร์ Income เช่นกัน

```
from sklearn.preprocessing import LabelEncoder

customer_encoded = customer_final.copy()

encoder = LabelEncoder()
customer_encoded['Education'] = customer_encoded[['Education']].apply(encoder.fit_transform)
customer_encoded['Marital_Status'] = customer_encoded[['Marital_Status']].apply(encoder.fit_transform)

customer_encoded['Income'] = customer_encoded['Income'].astype('int')
```

customer_final.dtypes		customer_encoded.dtypes	
Education	object	Education	int64
Marital_Status	object	Marital_Status	int64
Income	float64	Income	int64
Age	int64	Age	int64
Customer_for	int64	Customer_for	int64
Children	int64	Children	int64
Purchases	int64	Purchases	int64
AcceptedCmp_total	int64	AcceptedCmp_total	int64
Purchases_total	int64	Purchases_total	int64
dtype: object		dtype: object	

4. Feature Scaling (การปรับขนาดฟีเจอร์)

ทำการปรับสเกลของข้อมูลด้วย StandardScaler() เพื่อทำให้ชุดข้อมูลในแต่ละฟีเจอร์มีความใกล้เคียงกัน ดังรูปและสร้างตัวแปรใหม่มารับค่าซึ่งว่า customer_scaled

```
from sklearn.preprocessing import StandardScaler

customer_scaling = customer_encoded.copy()

scaler = StandardScaler()
customer_scaled = scaler.fit_transform(customer_scaling)
customer_scaled = pd.DataFrame(customer_scaled, columns = customer_scaling.columns)
customer_scaled
```

	Education	Marital_Status	Income	Age	Customer_for	Children	Purchases	AcceptedCmp_total	Purchases_total
0	-0.894974	1.346874	0.243707	0.912609	1.976745	-1.264505	1.679417	0.621248	1.320826
1	-0.894974	1.346874	-0.241011	1.334624	-1.667011	1.396361	-0.961275	-0.501912	-1.154596
2	-0.894974	-0.742460	0.797512	0.321787	-0.172468	-1.264505	0.282673	-0.501912	0.799685
3	-0.894974	-0.742460	-1.050574	-1.197469	-1.925433	0.065928	-0.918094	-0.501912	-0.894025
4	0.568341	-0.742460	0.250077	-0.944260	-0.822831	0.065928	-0.305254	-0.501912	0.539114
...
2235	-0.894974	-0.742460	0.370496	0.152981	0.124718	0.065928	1.221032	-0.501912	0.408829
2236	0.568341	-0.742460	0.485203	2.009849	-1.942661	2.726794	-0.268717	0.621248	0.929970
2237	-0.894974	1.346874	0.196156	-0.944260	-0.848673	-1.264505	1.054951	0.621248	0.539114
2238	0.568341	-0.742460	0.700190	1.165818	-0.844366	0.065928	0.393948	-0.501912	1.060255
2239	0.568341	-0.742460	0.027158	1.165818	1.162714	1.396361	-0.720458	0.621248	-0.503169

2240 rows × 9 columns

5. Dimensionally Reduction by using ‘PCA’ (การลดมิติของข้อมูล)

เนื่องจากจำนวนฟีเจอร์ของ customer_scaled มี 9 ฟีเจอร์ซึ่งจำนวนมิติของข้อมูลในการทำโมเดล ไม่ได้เยอะจังไม่จำเป็นต้องทำการลดมิติของข้อมูลแต่ต้องการทำ PCA เพื่อดู principle component ของชุดข้อมูลเพื่อนำไปใช้ประกอบกับการแบ่งคลัสเตอร์เพื่อให้เห็นมุมมองต่างๆ

```
pca = PCA(n_components = 9, random_state=RANDOM_STATE)
customer_pca = pca.fit_transform(customer_scaled)

print('n_components =', pca.n_components_)
print('Percentage of explained variance with 9 components is {:.1f}'.format(pca.explained_variance_ratio_.sum()*100), '%')

n_components = 9
Percentage of explained variance with 9 components is 100.0 %
```

หลังจากการทำ pca โดยใช้ n_components = 9 แสดงแต่ละ components "ได้ดังนี้"

	0	1	2	3	4	5	6	7	8
0	2.664568	-0.477647	-0.039835	-1.598867	1.482155	-0.265189	-1.255402	-0.398818	-0.276639
1	-1.783606	2.128529	1.824187	-0.365162	0.545231	0.289770	0.131352	0.268713	-0.220387
2	1.224657	0.531644	-0.199092	-0.252802	-1.062176	-0.992718	-0.646463	0.027615	0.522945
3	-1.881068	0.024295	1.030333	0.124439	-2.094670	-0.167483	0.307713	-0.607415	0.013539
4	-0.196643	-0.145438	-0.283857	0.801368	-0.859356	-0.574736	1.010353	-0.267538	0.449190
...
2235	0.928402	0.607056	-0.594133	-0.660637	-0.752491	-0.433796	0.053023	-0.133540	-0.804875
2236	-0.221852	3.251590	-0.320320	1.159293	0.338690	1.660182	1.537765	-0.577481	0.209215
2237	1.469107	-0.792400	1.978378	-0.451682	-0.410326	-0.461616	0.107076	-0.571499	-0.173733
2238	0.849738	1.543468	-0.586053	1.036081	-0.077398	-0.488708	0.209327	-0.342397	0.212272
2239	-0.714416	0.652684	-1.474622	0.026130	0.872313	1.498843	-0.186607	0.722625	0.134139

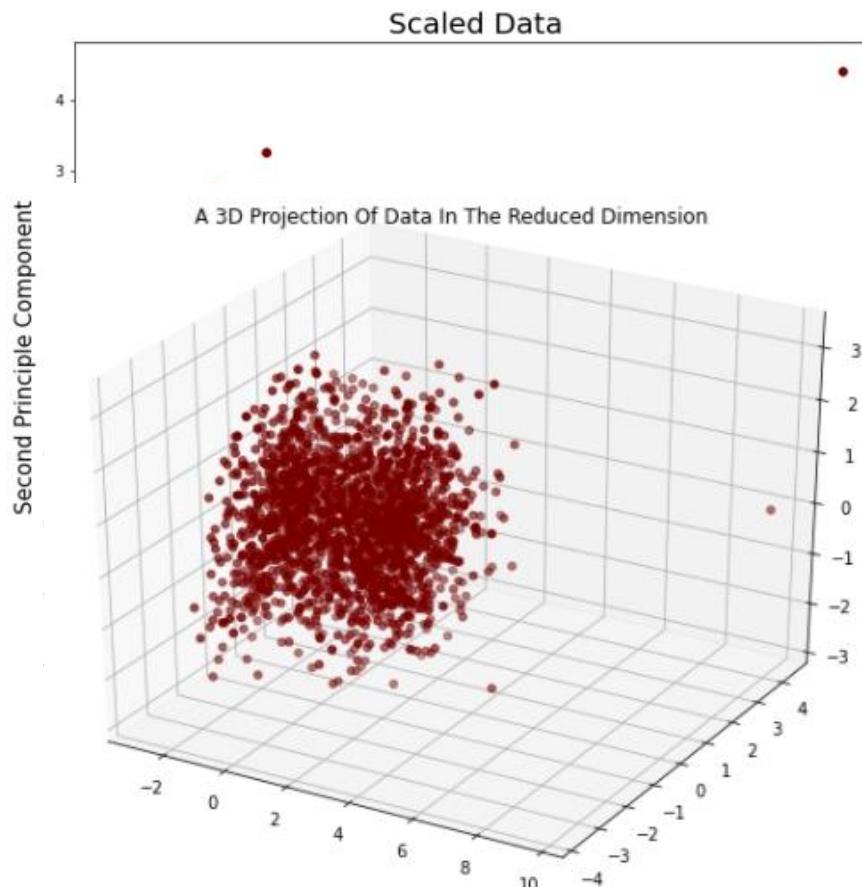
แสดง first, second, third principle components ของชุดข้อมูล ดังนี้

```

0      2.664568
1     -1.783606
2      1.224657
3     -1.881068
4     -0.196643
...
2235    0.928402
2236   -0.221852
2237    1.469107
2238    0.849738
2239   -0.714416
Name: 0, Length: 2240, dtype: float64 0      -0.477647
1      2.128529
2      0.531644
3      0.024295
4     -0.145438
...
2235    0.607056
2236   3.251590
2237   -0.792400
2238   1.543468
2239   0.652684
Name: 1, Length: 2240, dtype: float64 0      -0.039835
1      1.824187
2     -0.199092
3     1.030333
4     -0.283857
...
2235   -0.594133
2236   -0.320320
2237   1.978378
2238   -0.586053
2239   -1.474622
Name: 2, Length: 2240, dtype: float64

```

ทำการ plot เพื่อแสดงรูปร่างของชุดข้อมูลโดยแสดงเฉพาะ first, second principle components และ first, second, third principle components ซึ่งจากรูปทั้ง 2 เห็นได้ว่าชุดข้อมูลมีการเกาะกลุ่มเป็นรูปทรงกลม ซึ่งหมายความว่าการนำไปทำ clustering ด้วยวิธี K-Means โดยใช้ชุดข้อมูลชื่อ customer_pca



ขั้นตอนที่ 3: Modeling

เทคนิคการทำ Unsupervised Learning Model ในการทำ clustering ที่เลือกใช้ได้แก่

1. K-Means Clustering
2. Agglomerative Clustering
3. DBSCAN Clustering

โดยจะนำผลลัพธ์การทำ clustering ในแต่ละโมเดลมาเปรียบเทียบกันเพื่อหาโมเดลที่ดีที่สุดและเหมาะสมในการวิเคราะห์กลุ่มลูกค้าจาก dataset ชุดนี้

K-Means Clustering

1. คำนวณหาค่า The sum of residual square (RSS) เพื่อหาจำนวน k clusters ที่เหมาะสมในการทำ K-Means clustering ซึ่งจะทำการสร้างลูปเพื่อวนหาค่า RSS ของ n_clusters ตั้งแต่ 1-10 ที่ fit กับ K-Means ด้วยฟังก์ชัน .inertia_() เก็บใส่ไว้ในอาเรย์ wcss เพื่อนำไปใช้กับ Elbow method เพื่อหาค่า RSS ที่ดีที่สุด

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init='k-means++', max_iter=300, n_init=100,
                     random_state=RANDOM_STATE)
    kmeans.fit(customer_pca)
    wcss.append(kmeans.inertia_)
wcss
[20159.99999999993,
 15307.931026262926,
 13884.530332364322,
 12697.964875713757,
 11844.292100553228,
 11090.089941605842,
 10552.006563561554,
 10065.696555222083,
 9590.95246541001,
 9217.240497039806]
```

2. นำค่า RSS ในอาเรย์ wcss ที่ได้จาก K-Means model ที่มีพารามิเตอร์ n_clusters ตั้งแต่ 1-10 มาคำนวณหา elbow point โดยใช้ฟังก์ชัน Elbow method จะได้ว่า elbow point คือ n_clusters = 3 เนื่องจากที่ n_clusters = 3 ให้ค่า RSS ที่ drop มากที่สุด

```
#Elbow method หา k cluster ที่มีการ drop ค่า RSS มากที่สุด
from kneed import KneeLocator

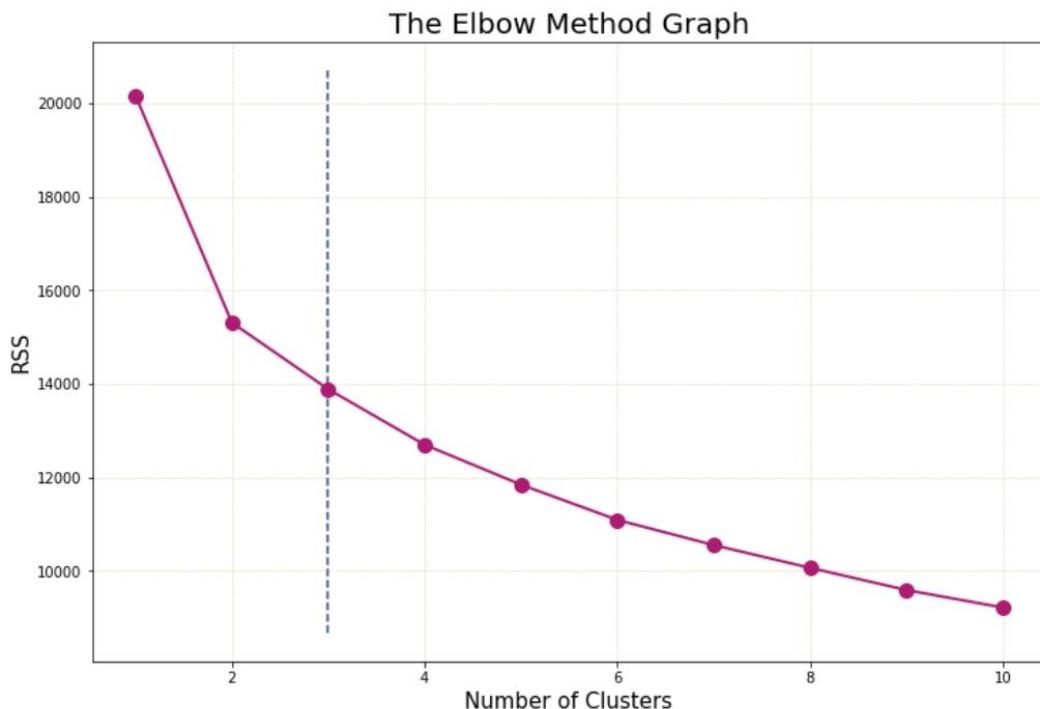
kn = KneeLocator(np.arange(1,11), wcss, curve='convex', direction='decreasing')
elbow_point = kn.elbow

#แสดง elbow point
print('The elbow point is ', elbow_point)
```

The elbow point is 3

```
plt.figure(figsize=(12,8))
plt.grid(True, color = "#D4D5B0", linewidth = "0.5", linestyle = "--")
#Plot RSS ของแต่ละ k clusters
plt.plot(np.arange(1,11), wcss, linewidth=2, marker='o', ms=10, color='#B81B73')
#Plot elbow point = 3
plt.vlines(elbow_point, plt.ylim()[0], plt.ylim()[1], linestyles='dashed', color='#2D4A8F')

plt.title('The Elbow Method Graph', fontsize=20)
plt.xlabel('Number of Clusters', fontsize=15)
plt.ylabel('RSS', fontsize=15)
plt.show()
```



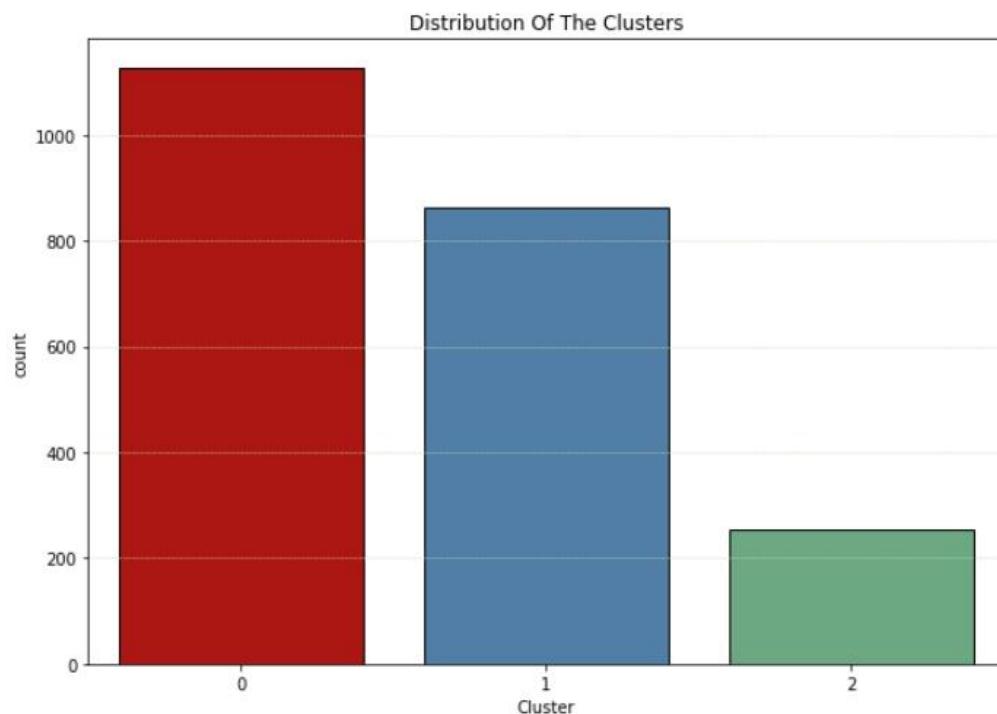
3. จะได้ว่า `n_clusters = 3` เป็นจำนวนคลัสเตอร์ที่เหมาะสมที่จะนำไปใช้ในการสร้างโมเดล K-Means Clustering โดยระบุพารามิเตอร์ `n_clusters = 3`, `n_init = 100`, `random_state = 20` จากนั้นทำการ `fit model` และ `predict` คลัสเตอร์กับข้อมูล `customer_pca` ทำให้สามารถแบ่งคลัสเตอร์ได้ทั้งหมด 3 คลัสเตอร์ ดังนี้ คลัสเตอร์ 1: 1125 instances คลัสเตอร์ 2: 862 instances คลัสเตอร์ 3: 253 instances

```
kmeans = KMeans(n_clusters = 3, n_init=100, random_state=RANDOM_STATE)
kmeans.fit(customer_pca)
```

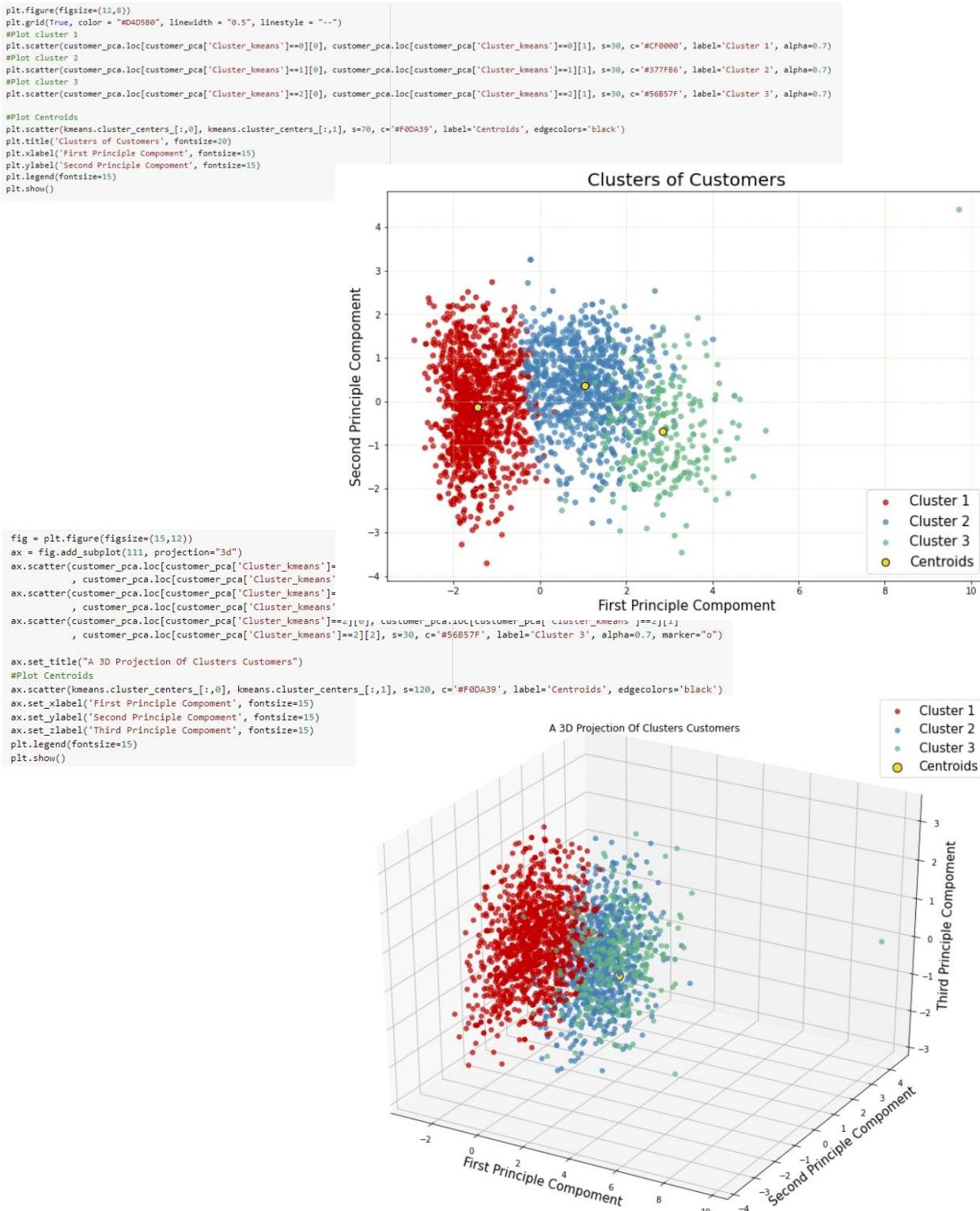
```
KMeans(n_clusters=3, n_init=100, random_state=20)
```

```
print("Clusters present: {}".format(np.unique(labels_kmeans)))
print("Cluster sizes KMeans clustering: {}".format(np.bincount(labels_kmeans)))
```

```
Clusters present: [0 1 2]
Cluster sizes KMeans clustering: [1125  862  253]
```



4. ทำการ Visualizing เพื่อแสดงคลัสเตอร์ที่แบ่งได้ทั้ง 3 คลัสเตอร์โดยแสดงเฉพาะ first, second, third principle component เพื่อให้เห็นรูปร่างของคลัสเตอร์ที่แบ่งได้จากโน้มเดล และแสดง cluster centroid ที่ใช้ในการแบ่งของแต่ละคลัสเตอร์บนกราฟ จากรูปจะเห็นได้ว่ามีการรวม outlier เข้าไปในการแบ่งคลัสเตอร์ด้วย



5. Visualize การแบ่งคลัสเตอร์เพื่อแสดงความสัมพันธ์กับรายได้และจำนวนการซื้อสินค้าต่อเดือนของลูกค้าแต่ละคลัสเตอร์

```
customer_kmeans = pd.concat([customer_final, pd.DataFrame({'Cluster' : labels_kmeans})], axis=1)
customer_kmeans
```

	Education	Marital_Status	Income	Age	Customer_for	Children	Purchases	AcceptedCmp_total	Purchases_total	Cluster
0	Graduate	Single	58138.0	55	971	0	1617	1	25	1
1	Graduate	Single	46344.0	60	125	2	27	0	6	0
2	Graduate	Relationship	71613.0	48	472	0	776	0	21	1
3	Graduate	Relationship	26646.0	30	65	1	53	0	8	0
4	Postgraduate	Relationship	58293.0	33	321	1	422	0	19	1
...
2235	Graduate	Relationship	61223.0	46	541	1	1341	0	18	1
2236	Postgraduate	Relationship	64014.0	68	61	3	444	1	22	1
2237	Graduate	Single	56981.0	33	315	0	1241	1	19	1
2238	Postgraduate	Relationship	69245.0	58	316	1	843	0	23	1
2239	Postgraduate	Relationship	52869.0	58	782	2	172	1	11	0

2240 rows × 10 columns

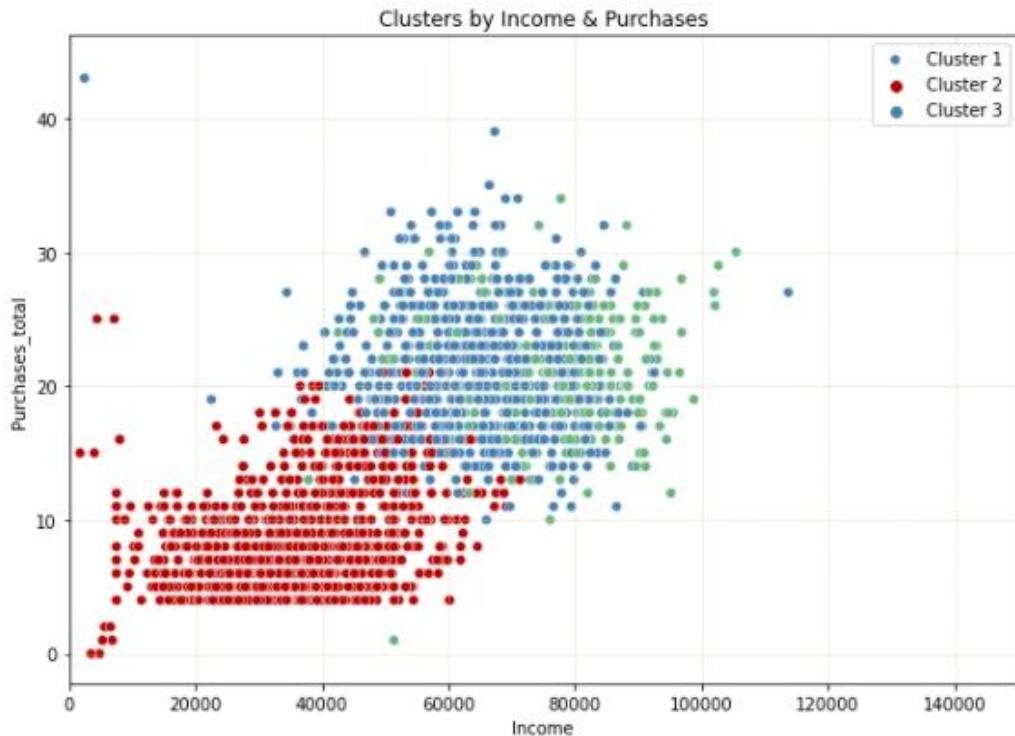
```
plt.figure(figsize=(10,7))
plt.grid(True, color = "#D4D5B0", linewidth = "0.5", linestyle = "--")
pl = sns.scatterplot(customer_kmeans["Income"], customer_kmeans["Purchases"], hue = customer_kmeans["Cluster"], palette= cl, data=customer_kmeans)
pl.set_title("Clusters by Income & Purchases")
pl.set_xlim(xmin=0, xmax=150000)
plt.legend(['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
plt.show()
```



จากราฟจะเห็นได้ว่าคลัสเตอร์ที่ 1 เป็นกลุ่มลูกค้าที่มีรายได้น้อยถึงปานกลาง มีปริมาณการซื้อสินค้าต่อเดือนจำนวนน้อยที่สุด คลัสเตอร์ที่ 2 เป็นกลุ่มลูกค้าที่มีรายรายได้ปานกลางถึงมาก มีปริมาณการซื้อสินค้าต่อเดือนค่อนข้างเยอะ และคลัสเตอร์ที่ 3 เป็นกลุ่มลูกค้าที่มีรายได้สูงและมีจำนวนคนในกลุ่มน้อยที่สุด มีปริมาณการซื้อสินค้าเยอะมาก

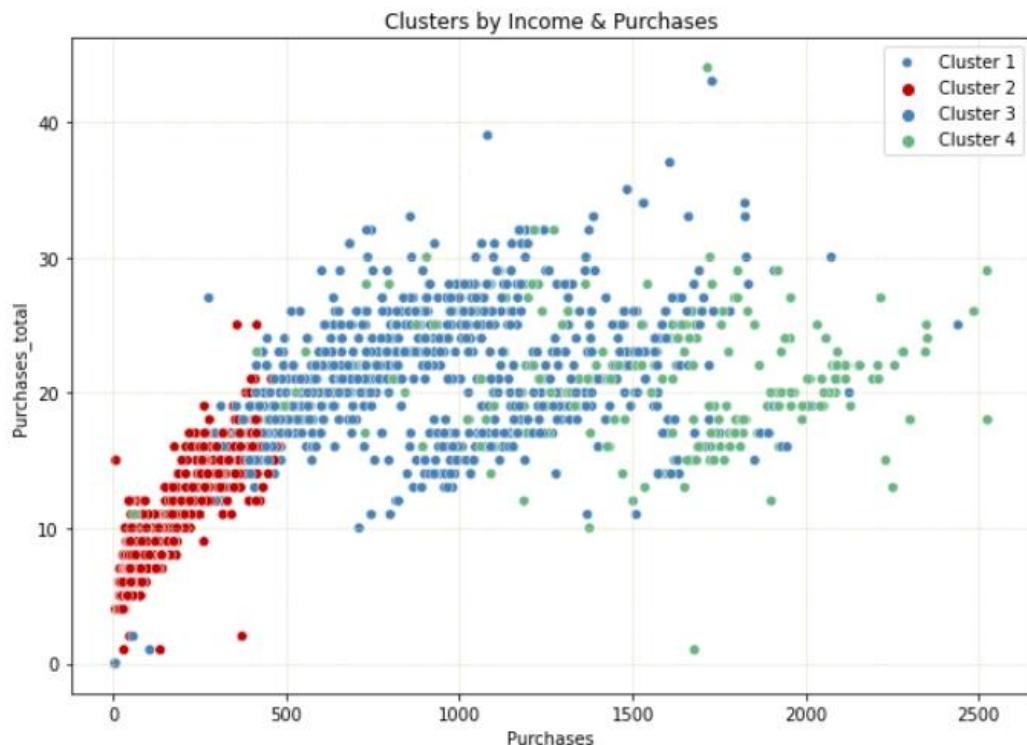
6. Visualize การแบ่งคลัสเตอร์เพื่อแสดงความสัมพันธ์กับรายได้และจำนวนการผ่านช่องทางต่างๆ ต่อเดือนของลูกค้าแต่ละคลัสเตอร์

```
plt.figure(figsize=(10,7))
plt.grid(True, color = "#D4D5B0", linewidth = "0.5", linestyle = "--")
pl = sns.scatterplot(customer_kmeans["Income"], customer_kmeans["Purchases_total"], hue = customer_kmeans["Cluster"], palette= cl, data=customer_kmeans)
pl.set_title("Clusters by Income & Purchases")
pl.set_xlim(xmin=0, xmax=150000)
plt.legend(['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
plt.show()
```



จากราฟจะเห็นได้ว่าคลัสเตอร์ที่ 1 เป็นกลุ่มลูกค้าที่มีรายได้น้อยถึงปานกลาง มีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆ ต่อเดือนจำนวนน้อยที่สุด คลัสเตอร์ที่ 2 เป็นกลุ่มลูกค้าที่มีรายรายได้ปานกลางถึงมาก มีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆ ต่อเดือนเยอะที่สุด และคลัสเตอร์ที่ 3 เป็นกลุ่มลูกค้าที่มีรายได้สูงและมีจำนวนคนในกลุ่มน้อยที่สุด มีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆ ต่อเดือนค่อนข้างเยอะ

7. Visualize การแบ่งคลัสเตอร์เพื่อแสดงความสัมพันธ์กับจำนวนการซื้อสินค้าต่อเดือนและจำนวนการผ่านช่องทางต่างๆต่อเดือนของลูกค้าแต่ละคลัสเตอร์

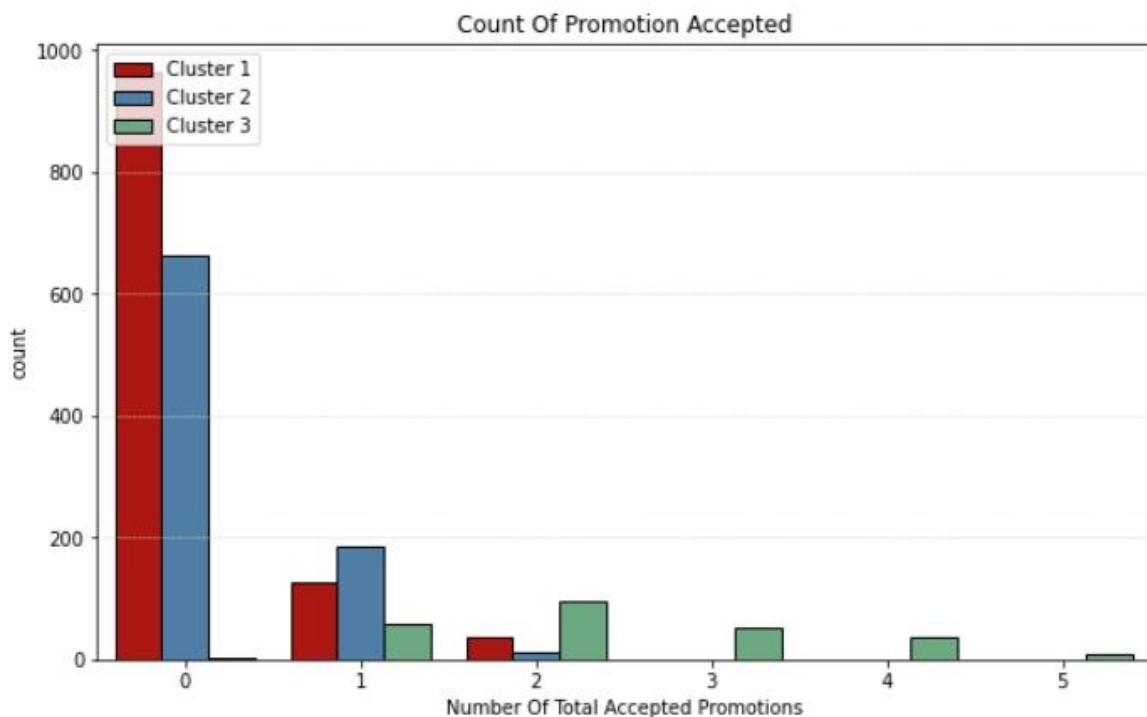


```
plt.figure(figsize=(10,7))
plt.grid(True, color = "#D4D5B0", linewidth = "0.5", linestyle = "--")
pl = sns.scatterplot(customer_kmeans["Purchases"], customer_kmeans["Purchases_total"], hue = customer_kmeans["Cluster"], palette= cl, data=customer_kmeans)
pl.set_title("Clusters by Income & Purchases")
plt.legend(['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
plt.show()
```

จากราฟจะเห็นได้ว่าคลัสเตอร์ที่ 1 เป็นกลุ่มลูกค้าที่มีปริมาณการซื้อสินค้าต่อเดือนและมีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆต่อเดือนจำนวนน้อยที่สุด คลัสเตอร์ที่ 2 เป็นกลุ่มลูกค้าที่มีปริมาณการซื้อสินค้าต่อเดือนและมีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆต่อเดือนปานกลางถึงเยอะ และคลัสเตอร์ที่ 3 เป็นกลุ่มลูกค้าที่มีจำนวนคนในกลุ่มน้อยที่สุด มีปริมาณการซื้อสินค้าต่อเดือนจำนวนมากและมีปริมาณการซื้อสินค้าผ่านช่องทางต่างๆต่อเดือนปานกลาง

8. Visualize การแบ่งคลัสเตอร์เพื่อแสดงความสัมพันธ์กับจำนวนการเข้าร่วมแคมเปญและโปรโมชันของลูกค้าแต่ละคลัสเตอร์

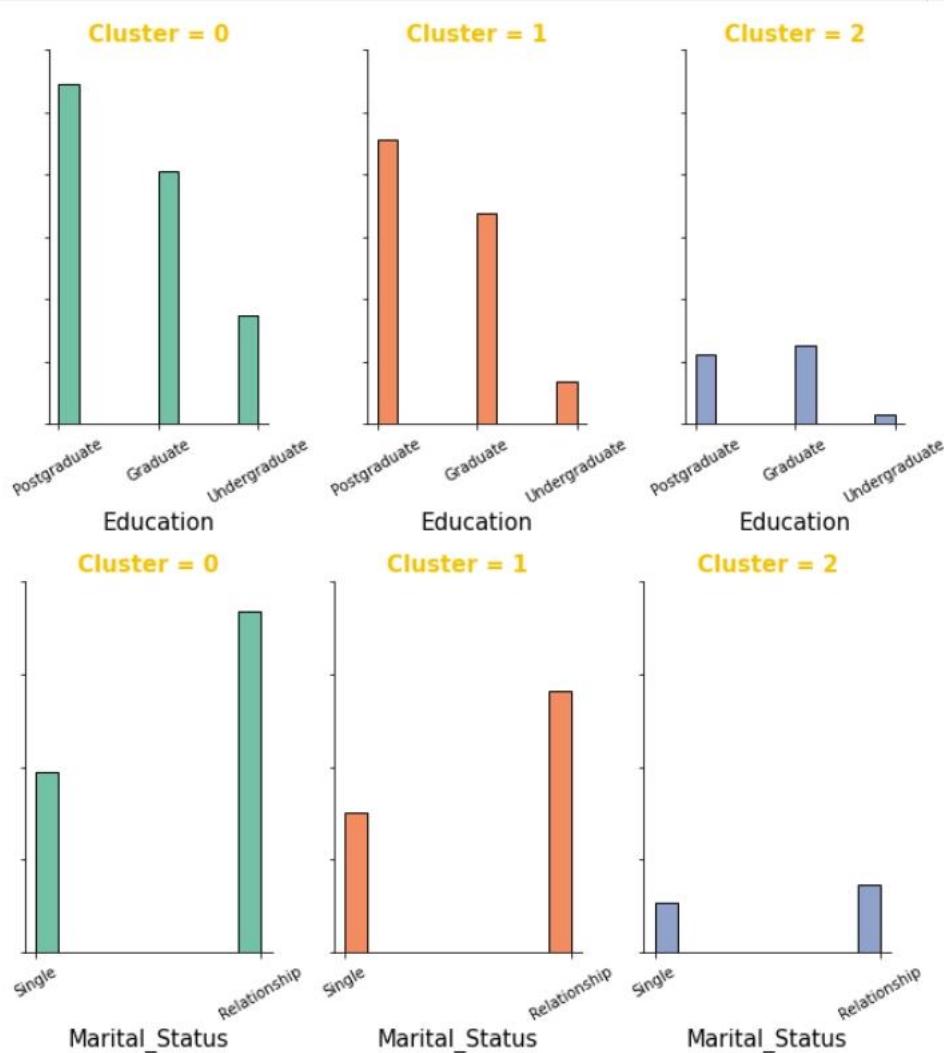
```
plt.figure(figsize=(10,6))
plt.grid(True, color = "#D4D5B0", linewidth = "0.5", linestyle = "--")
pl = sns.countplot(x=customer_kmeans["AcceptedCmp_total"],hue=customer_kmeans["Cluster"], palette= cl, ec='k')
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.legend(['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
plt.show()
```

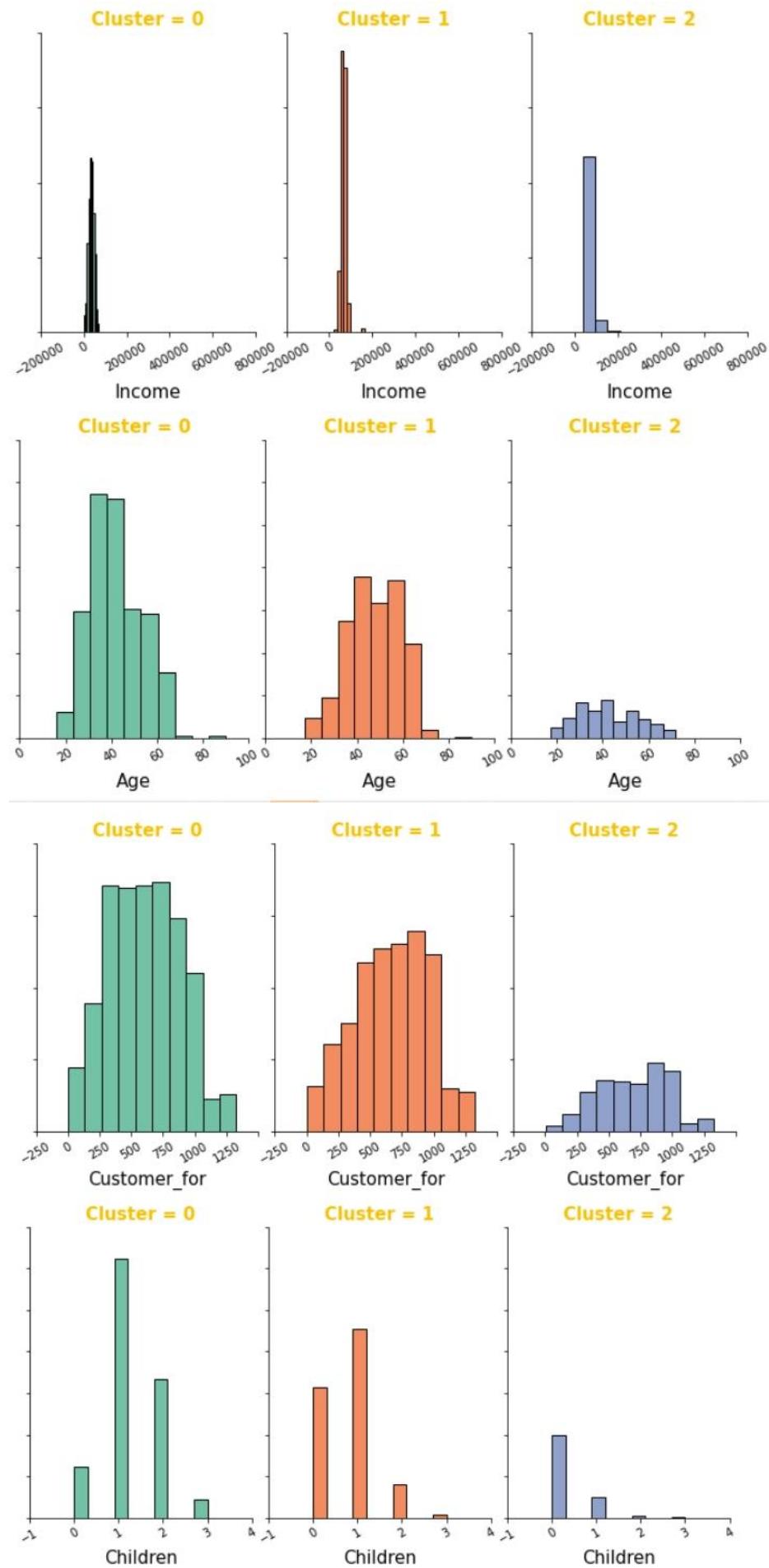


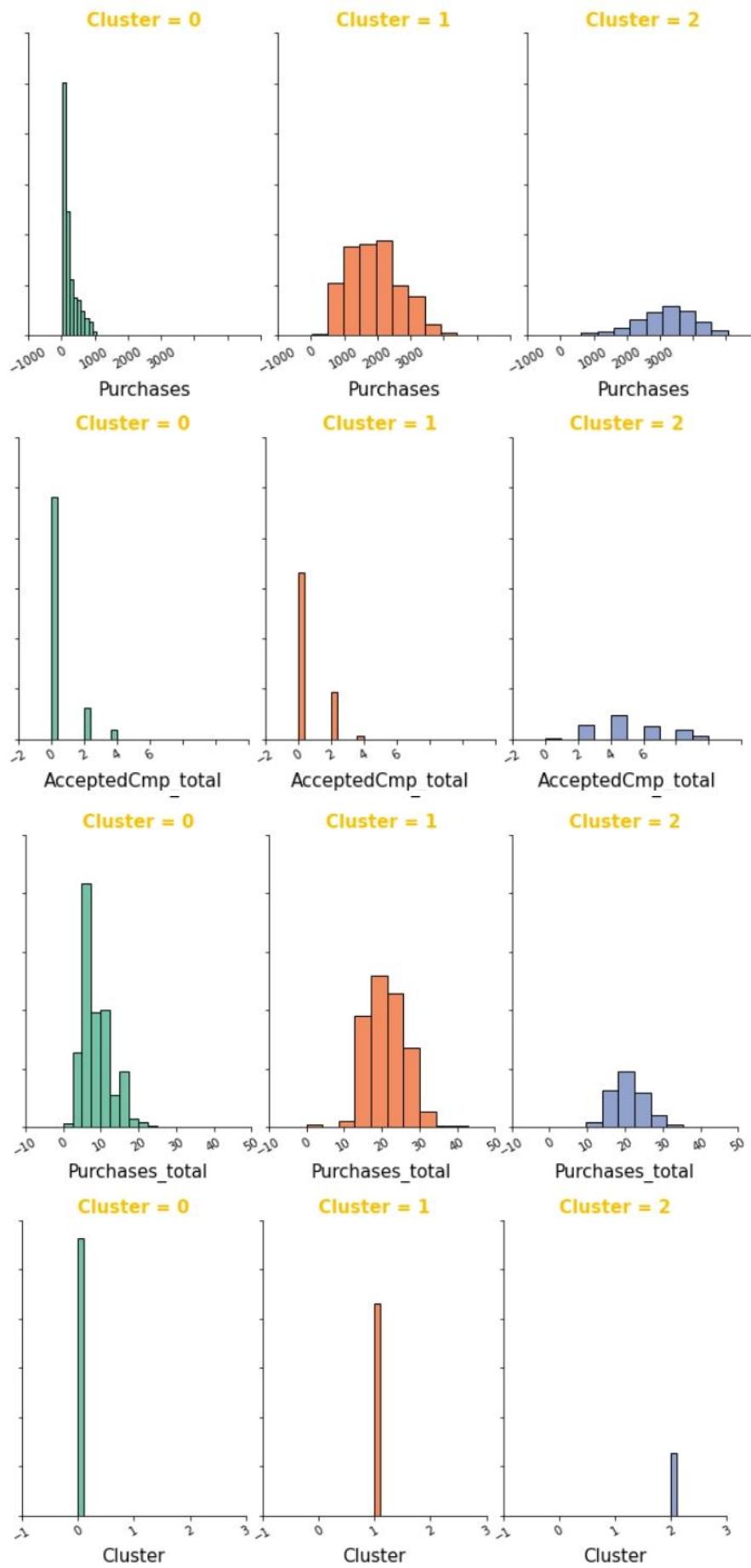
จากราฟเห็นได้ว่าคลัสเตอร์ที่ 1 และ 2 มีจำนวนลูกค้าที่ไม่ได้เข้าร่วมแคมเปญหรือรับโปรโมชันเลยมากที่สุด และคลัสเตอร์ที่ 3 มีการรับโปรโมชันและเข้าร่วมแคมเปญตลอด

9. ทำการ visualize เพื่อแสดงความสัมพันธ์ของการแบ่งคลัสเตอร์กับฟีเจอร์แต่ละอัน ได้ดังนี้

```
for i in customer_kmeans:
    x = sns.FacetGrid(customer_kmeans, col='Cluster', hue='Cluster', palette='Set2')
    x.map(plt.hist, i, bins=10, ec='k')
    x.set_xticklabels(rotation=30, color='black')
    x.set_yticklabels(color='black')
    x.set_xlabels(size=15, color='black')
    x.set_titles(size=15, color='#FFC300', fontweight='bold')
    x.fig.set_figheight(5)
```







Agglomerative Clustering

- ทำการ slicing ข้อมูลจาก customer_pca ด้วยคำสั่ง customer_pca.iloc[:, 0:9]

```
[120] customer_pca = customer_pca.iloc[:, 0:9]
customer_pca
```

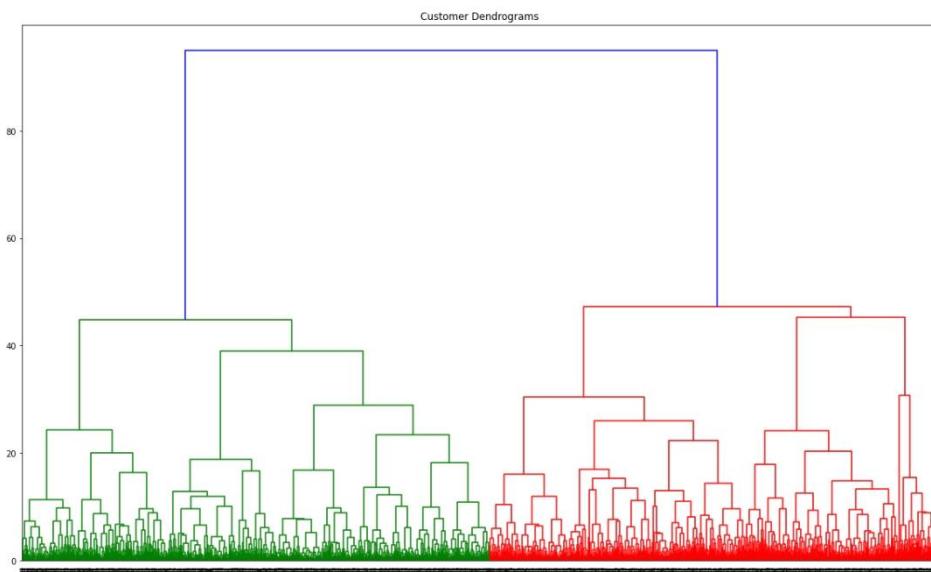
	0	1	2	3	4	5	6	7	8
0	2.664568	-0.477647	-0.039835	-1.598867	1.482155	-0.265189	-1.255402	-0.398818	-0.276639
1	-1.783606	2.128529	1.824187	-0.365162	0.545231	0.289770	0.131352	0.268713	-0.220387
2	1.224657	0.531644	-0.199092	-0.252802	-1.062176	-0.992718	-0.646463	0.027615	0.522945
3	-1.881068	0.024295	1.030333	0.124439	-2.094670	-0.167483	0.307713	-0.607415	0.013539
4	-0.196643	-0.145438	-0.283857	0.801368	-0.859356	-0.574736	1.010353	-0.267538	0.449190
...
2235	0.928402	0.607056	-0.594133	-0.660637	-0.752491	-0.433796	0.053023	-0.133540	-0.804875
2236	-0.221852	3.251590	-0.320320	1.159293	0.338690	1.660182	1.537765	-0.577481	0.209215
2237	1.469107	-0.792400	1.978378	-0.451682	-0.410326	-0.461616	0.107076	-0.571499	-0.173733
2238	0.849738	1.543468	-0.586053	1.036081	-0.077398	-0.488708	0.209327	-0.342397	0.212272
2239	-0.714416	0.652684	-1.474622	0.026130	0.872313	1.498843	-0.186607	0.722625	0.134139

2240 rows × 9 columns

- Import ไลบรารี scipy.cluster.hierarchy เพื่อนำมาสร้าง dendrogram จาก customer_pca ด้วยคำสั่ง shc.dendrogram(shc.linkage(customer_pca, method='ward'))

```
[121] import scipy.cluster.hierarchy as shc

plt.figure(figsize=(20, 12))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(customer_pca, method='ward'))
plt.xticks(rotation = 90)
```



3. Import AgglomerativeClustering จากไลบรารี sklearn.cluster เพื่อทำการ clustering ด้วยคำสั่ง
`agg = AgglomerativeClustering(n_clusters = 3, affinity='euclidean', linkage='ward')`
 และนำโมเดลไป fit และ predict กับข้อมูล customer_pca ด้วยคำสั่ง `labels_agg = agg.fit_predict(customer_pca)` และสร้างคอลัม์เพิ่มเข้าไปใน customer_pca โดยชื่อคอลัมน์คือ Cluster_Agg ด้วยคำสั่ง `customer_pca['Cluster_Agg'] = labels_agg` เรากำหนด n_clusters = 3 เพื่อทำให้สามารถเปรียบเทียบกับ KMeans ได้ และพิมพ์ข้อมูลออกมา จะได้ผลลัพธ์ดังนี้

```
[122] from sklearn.cluster import AgglomerativeClustering
agg = AgglomerativeClustering(n_clusters = 3, affinity='euclidean', linkage='ward')
labels_agg = agg.fit_predict(customer_pca)

customer_pca['Cluster_Agg'] = labels_agg
customer_pca
```

	0	1	2	3	4	5	6	7	8	Cluster_Agg
0	2.664568	-0.477647	-0.039835	-1.598867	1.482155	-0.265189	-1.255402	-0.398818	-0.276639	0
1	-1.783606	2.128529	1.824187	-0.365162	0.545231	0.289770	0.131352	0.268713	-0.220387	1
2	1.224657	0.531644	-0.199092	-0.252802	-1.062176	-0.992718	-0.646463	0.027615	0.522945	2
3	-1.881068	0.024295	1.030333	0.124439	-2.094670	-0.167483	0.307713	-0.607415	0.013539	1
4	-0.196643	-0.145438	-0.283857	0.801368	-0.859356	-0.574736	1.010353	-0.267538	0.449190	1
...
2235	0.928402	0.607056	-0.594133	-0.660637	-0.752491	-0.433796	0.053023	-0.133540	-0.804875	2
2236	-0.221852	3.251590	-0.320320	1.159293	0.338690	1.660182	1.537765	-0.577481	0.209215	2
2237	1.469107	-0.792400	1.978378	-0.451682	-0.410326	-0.461616	0.107076	-0.571499	-0.173733	0
2238	0.849738	1.543468	-0.586053	1.036081	-0.077398	-0.488708	0.209327	-0.342397	0.212272	2
2239	-0.714416	0.652684	-1.474622	0.026130	0.872313	1.498843	-0.186607	0.722625	0.134139	1

4. พิมพ์ค่าที่ cluster ท่านายอกกว่ามีอะไรบ้าง และพิมพ์จำนวน instance ของแต่ละ cluster ออกมากด้วยคำสั่ง `print('Cluster present: {}'.format(np.unique(labels_agg)))` และ `print('Cluster sizes Agglomerative clustering: {}'.format(np.bincount(labels_agg)))` ตามลำดับ และได้ผลลัพธ์ดังภาพ

```
[123] print("Clusters present: {}".format(np.unique(labels_agg)))
print("Cluster sizes Agglomerative clustering: {}".format(np.bincount(labels_agg)))

Clusters present: [0 1 2]
Cluster sizes Agglomerative clustering: [ 469 1145  626]
```

จะได้ 3 cluster ประกอบไปด้วย cluster 1:469 , 2: 1145, 3: 626 แม้ว่าจะมีการแบ่ง cluster ได้คละ label กับ Kmeans แต่ทั้งสองไม่เดลเมื่อการแบ่ง cluster จำนวนใกล้เคียงกัน

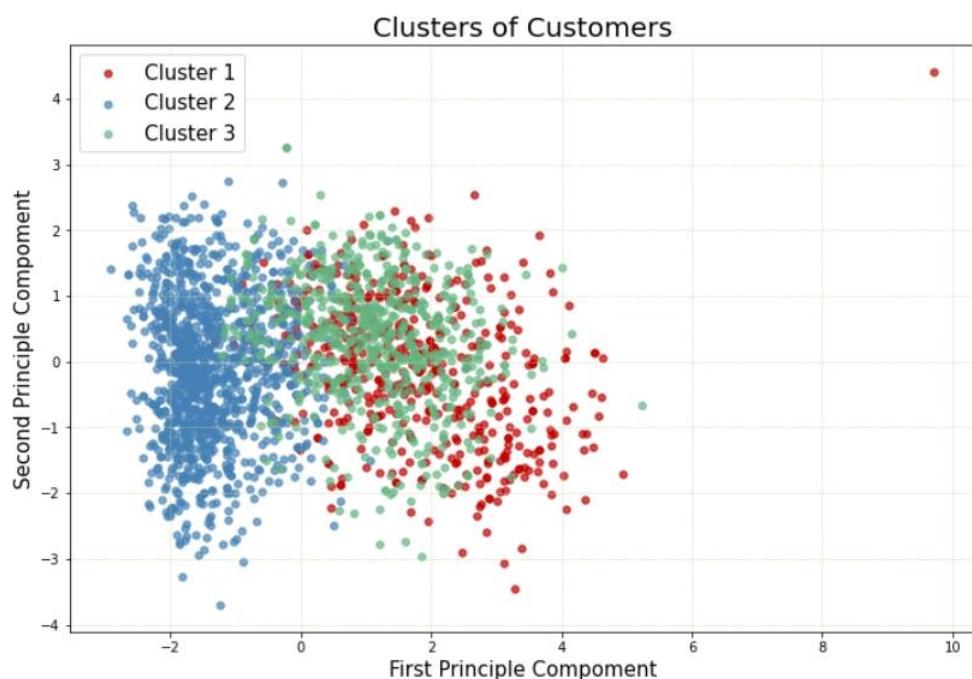
5. สร้าง dataframe ชื่อ customer_agg โดยนำ customer_final มา concatenate กับ dataframe ที่เก็บค่าที่ไม่เดลทำนายว่าเป็น cluster ในในแนวคอลัมน์ด้วยคำสั่ง customer_agg = pd.concat([customer_ofinal, pd.DataFrame({'Cluster' : labels_agg})], axis=1) จะได้ผลลัพธ์ดังภาพ

```
customer_agg = pd.concat([customer_final, pd.DataFrame({'Cluster' : labels_agg})], axis=1)
customer_agg
```

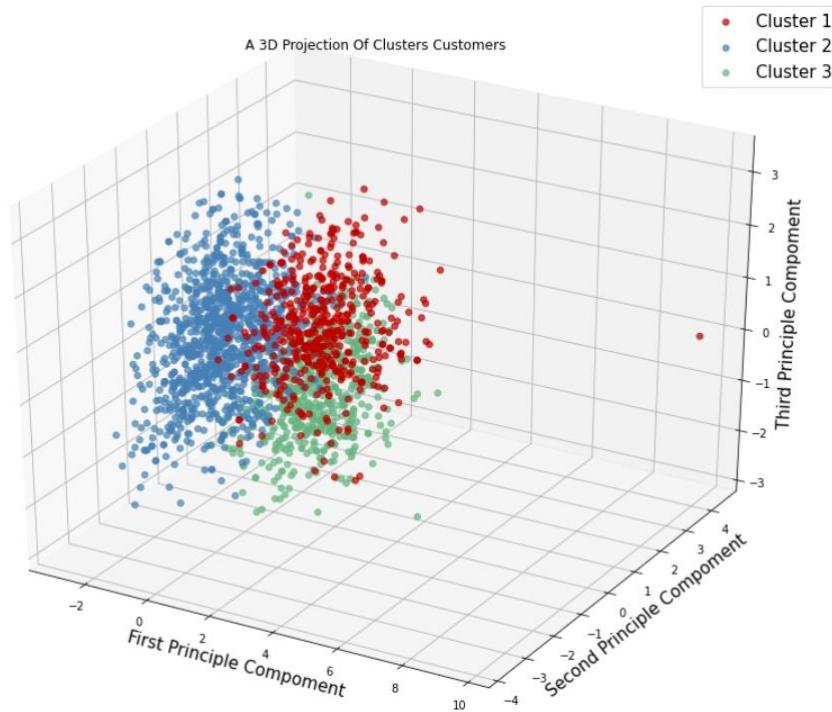
	Education	Marital_Status	Income	Age	Customer_for	Children	Purchases	AcceptedCmp_total	Purchases_total	Cluster
0	Graduate	Single	58138.0	55	971	0	1617	1	25	0
1	Graduate	Single	46344.0	60	125	2	27	0	6	1
2	Graduate	Relationship	71613.0	48	472	0	776	0	21	2
3	Graduate	Relationship	26646.0	30	65	1	53	0	8	1
4	Postgraduate	Relationship	58293.0	33	321	1	422	0	19	1
...
2235	Graduate	Relationship	61223.0	46	541	1	1341	0	18	2
2236	Postgraduate	Relationship	64014.0	68	61	3	444	1	22	2
2237	Graduate	Single	56981.0	33	315	0	1241	1	19	0
2238	Postgraduate	Relationship	69245.0	58	316	1	843	0	23	2
2239	Postgraduate	Relationship	52869.0	58	782	2	172	1	11	1

2240 rows × 10 columns

6. ทำการ plot กราฟ scatter plot ระหว่าง First Principle Component และ Second Principle Component ของทั้ง 3 clusters

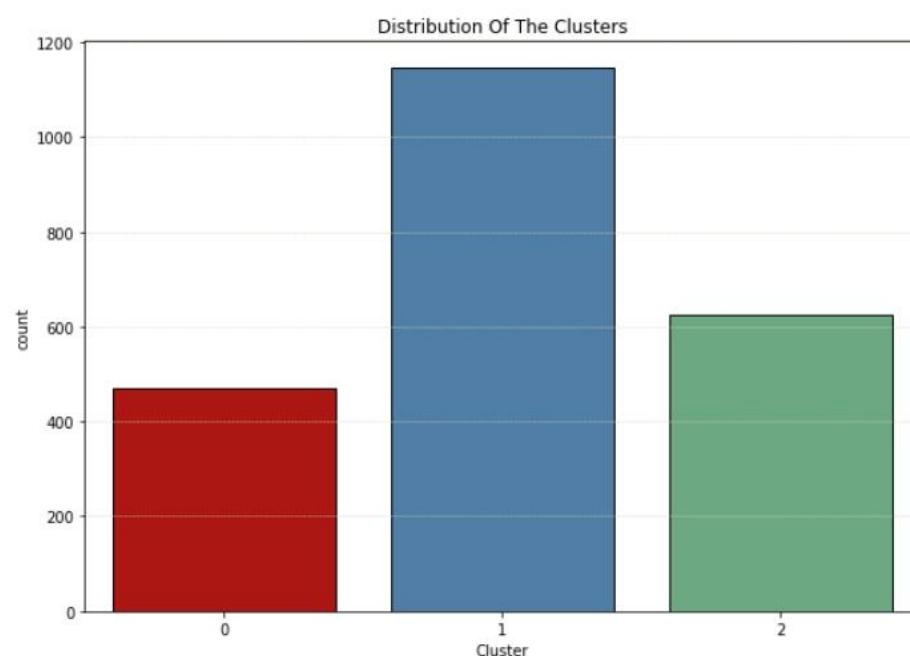


7. ทำการ plot กราฟสามมิติ ระหว่าง First Principle Component, Second Principle Component และ Third Principle Component ด้วยคำสั่งดังภาพและได้ผลลัพธ์ดังภาพจากกราฟทั้ง 2 จะเห็นว่ามีการรวม outlier เข้าไปใน cluster ด้วย

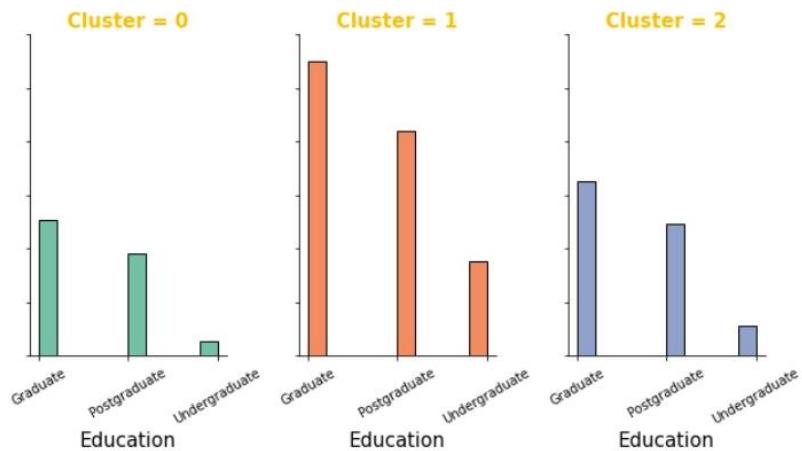


- ทำการ plot กราฟแท่งเพื่อเปรียบเทียบจำนวน instance ระหว่าง 3 clusters ด้วยคำสั่ง sns.countplot(x = customer_agg["Cluster"], palette= cl, ec='k') และได้ผลลัพธ์ดังภาพ

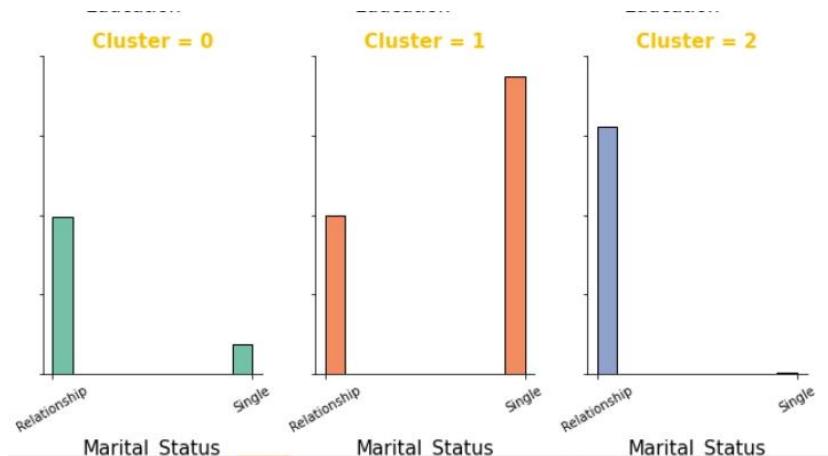
```
print("Clusters present: {}".format(np.unique(labels_agg)))
print("Cluster sizes Agglomerative clustering: {}".format(np.bincount(labels_agg)))
```



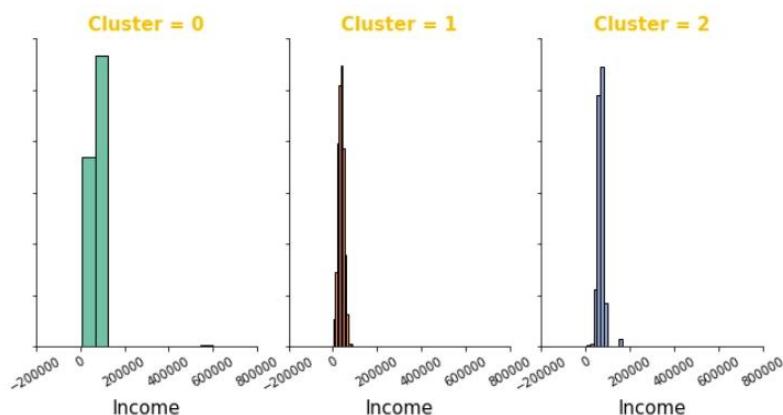
2. ทำการ plot กราฟแท่งเพื่อเปรียบเทียบจำนวน instance ในแต่ละ cluster ตาม features ต่างๆ ด้วยคำสั่ง `x = sns.FacetGrid(customer_agg, col='Cluster', hue='Cluster', palette='Set2')`
`x.map(plt.hist, i, bins=10, ec='k')` และได้ผลลัพธ์ดังนี้



จากราฟจะเห็นได้ว่า Cluster 1 มีจำนวน instance มากที่สุด และในแต่ละ cluster มีจำนวน instance ที่เป็น Graduate มากที่สุด



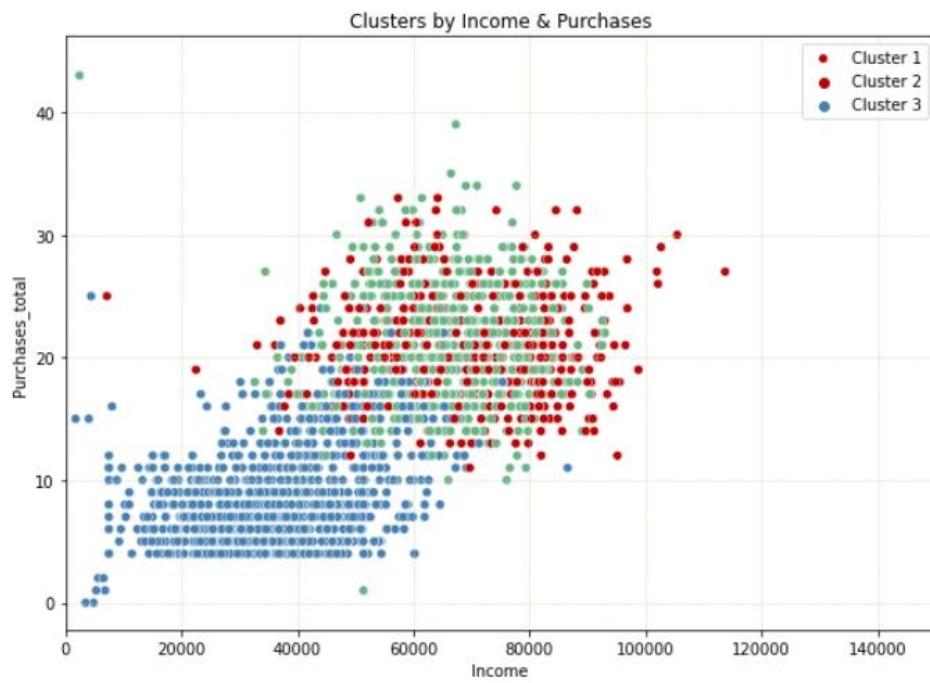
จากราฟจะเห็นได้ว่า Cluster 1 จะมี instance ที่เป็น Single มากกว่า Relationship ซึ่งต่างจาก Cluster 0 และ Cluster 2



7. ทำการ plot scatter plot ระหว่าง Income และ Purchases ของแต่ละ cluster จะได้ผลลัพธ์ดังภาพ



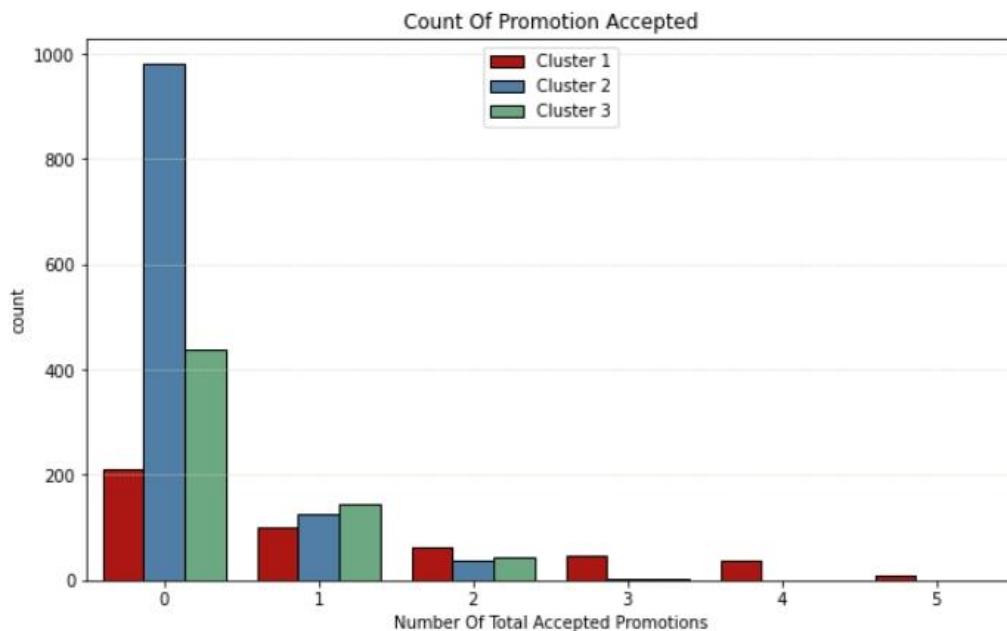
8. ทำการ plot scatter plot ระหว่าง Income และ Purchase_total ของแต่ละ cluster จะได้ผลลัพธ์ดังภาพ



9. ทำการ plot scatter plot ระหว่าง Purchases และ Purchase_total ของแต่ละ cluster จะได้ผลลัพธ์ดังภาพ



3. ทำการ plot กราฟแท่งที่แบ่งกลุ่มโดยจำนวนครั้งของการตอบรับแคมเปญเป็นตัวแบ่ง และจับกลุ่ม 3 cluster เข้าด้วยกัน และได้ผลลัพธ์ดังภาพ



DBSCAN

- ทำการสร้างโมเดล หมายคัลเลอต์ โดยใช้ default parameter โดยใช้ข้อมูล customer_scaled และพิมพ์ค่าอกมาว่ามี cluster อะไรบ้าง แต่ละ cluster มีกี่ instance

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN()
y_dbscan = dbscan.fit_predict(customer_scaled)
print("Clusters present: {}".format(np.unique(y_dbscan)))
print("Cluster sizes: {}".format(np.bincount(y_dbscan + 1)))

Clusters present: [-1  0   1   2   3   4   5   6   7   8   9   10  11  12  13  14]
Cluster sizes: [2029    17    12    75    40     5     6     5     12     7     7     5     5     5]
```

จะเห็นได้ว่า cluster -1 หรือ Noise มีจำนวนเยอะมากถึง 2029 คือไม่ถูกจัดอยู่ในกลุ่มใดๆเลย

- ปรับ parameter โดย parameter หลักที่ใช้คือ eps, min_samples ซึ่งเรากำหนดไว้ที่ 6
เราจะทำการหา eps ที่ทำให้แบ่ง cluster ได้ balanced มากที่สุด

```
[ ] for eps in[1,2,3,4,5]:
    print("\n eps = {}".format(eps))
    dbscan = DBSCAN(eps=eps, min_samples = 6)
    labels = dbscan.fit_predict(customer_pca)
    print("Clusters present: {}".format(np.unique(labels)))
    print("Cluster sizes: {}".format(np.bincount(labels+1)))
```

เราได้ eps=2 ซึ่งทำให้แบ่งคลัสเตอร์ได้ 2 cluster และมี noise เล็กน้อย

```
eps = 1
Clusters present: [-1  0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21  22
23 24 25 26]
Cluster sizes: [1065    50    293    200    13    10    90    36    136    49    30    19    17    50
22    22    28    10    13    14      6    11    25      6      6      6      7      6]

eps = 2
Clusters present: [-1  0   1]
Cluster sizes: [ 51  769 1420]

eps = 3
Clusters present: [-1  0]
Cluster sizes: [  8 2232]

eps = 4
Clusters present: [-1  0]
Cluster sizes: [  1 2239]

eps = 5
Clusters present: [-1  0]
Cluster sizes: [  1 2239]
```

3. สร้าง model ด้วย eps=2, min_samples=6

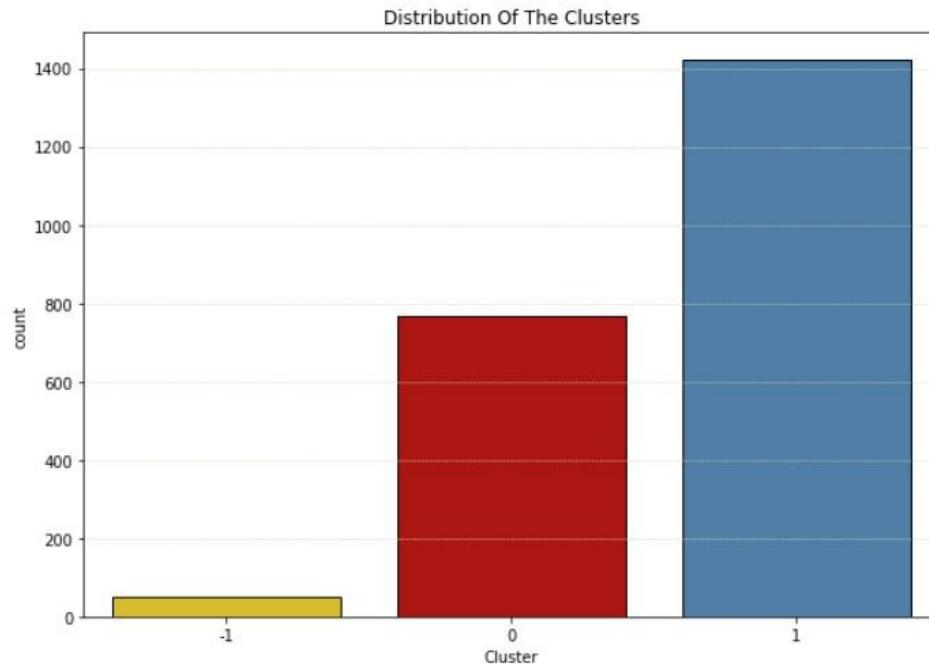
```

dbSCAN = DBSCAN(eps=2, min_samples = 6)
labels_dbSCAN = dbSCAN.fit_predict(customer_pca)

print("Clusters present: {}".format(np.unique(labels_dbSCAN)))
print("Cluster sizes DBSCAN clustering: {}".format(np.bincount(labels_dbSCAN + 1)))

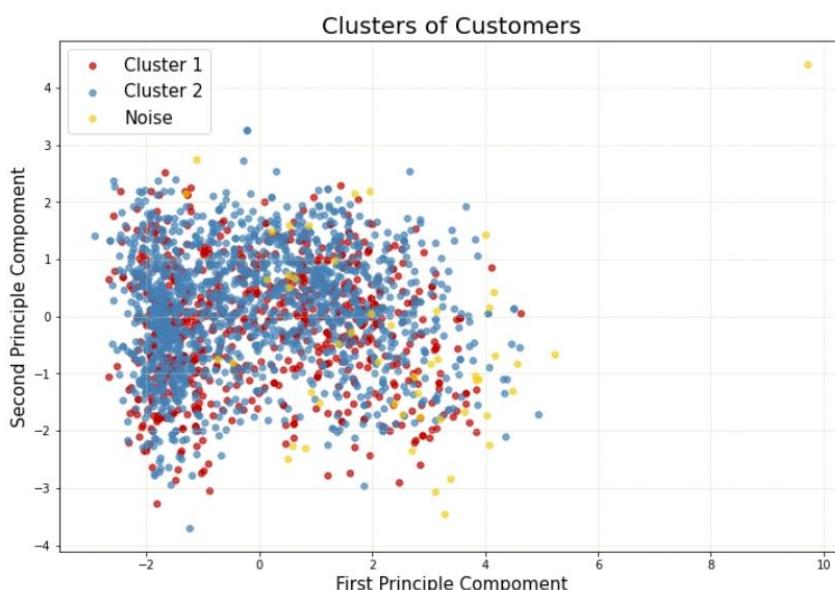
```

Clusters present: [-1 0 1]
Cluster sizes DBSCAN clustering: [51 769 1420]

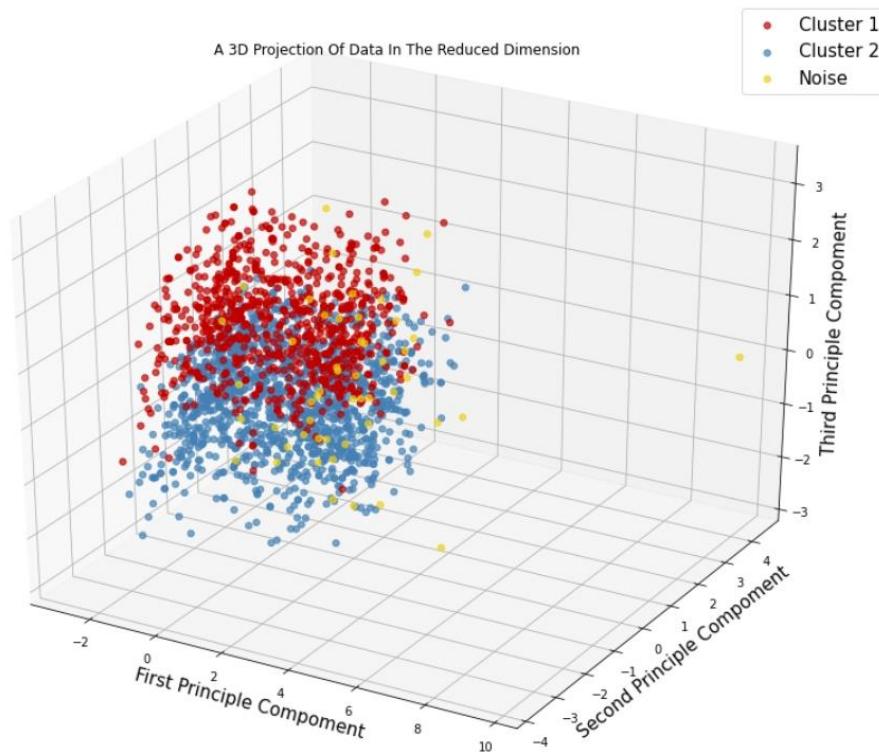


จะแบ่งคลัสเตอร์ได้ดังนี้ Cluster1: 769 instances, cluster2: 1420 instances และ Noise จำนวน 51 instances

4. visualize แสดง cluster กับ principle component (First และ Second)



5. visualize แสดง cluster กับ principle component แบบ 3 มิติ (First, Second และ Third)



ขั้นตอนที่ 4: Evaluating Model

หลังจากทำ clustering ทั้ง 3 model และขั้นตอนต่อไปคือการประเมินผลของโมเดลแต่ละอันเพื่อเปรียบเทียบและหาโมเดลที่สุดในการทำ clustering

1. เปรียบเทียบ model ด้วยวิธี ARI

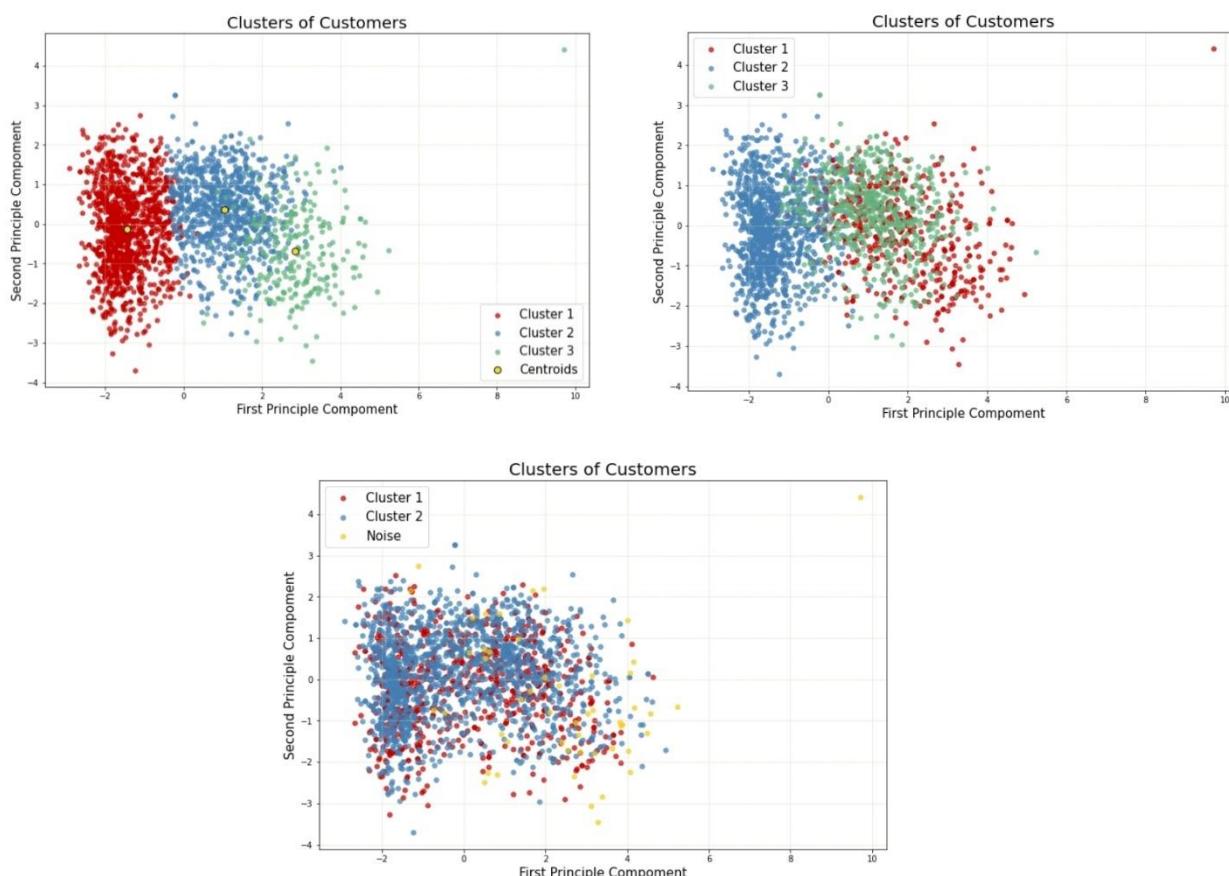
Import adjusted_rand_score() จาก sklearn.metrics.cluster เพื่อทำการเปรียบเทียบโมเดล

```
from sklearn.metrics.cluster import adjusted_rand_score
print("ARI - KMeans VS Agglomerative: {:.2f}".format(adjusted_rand_score(labels_kmeans, labels_agg)))
print("ARI - KMeans VS DBSCAN: {:.2f}".format(adjusted_rand_score(labels_kmeans, labels_dbscan)))
print("ARI - Agglomerative VS DBSCAN: {:.2f}".format(adjusted_rand_score(labels_agg, labels_dbscan)))
```

```
ARI - KMeans VS Agglomerative: 0.58
ARI - KMeans VS DBSCAN: 0.02
ARI - Agglomerative VS DBSCAN: 0.18
```

ผลลัพธ์ที่ได้คือการเปรียบเทียบแต่ละโมเดลว่ามีการจัดกลุ่มคลัสเตอร์คล้ายคลึงกันมากน้อยแค่ไหน สรุปได้ว่า

- K-Means และ Agglomerative clusterings มีความคล้ายคลึงในการจัดกลุ่มถึง 58 เปอร์เซ็นต์
- K-Means และ DBSCAN clusterings แทบไม่มีความคล้ายคลึงในการจัดกลุ่มเลย
- Agglomerative และ DBSCAN มีความคล้ายคลึงในการจัดกลุ่มแค่ 18 เปอร์เซ็นต์



2. เปรียบเทียบ model ด้วยวิธี Silhouette Score

import silhouette_score() จาก sklearn.metrics.cluster เพื่อประเมินการจัดกลุ่มคลัสเตอร์ของแต่ละโมเดลว่ามีการจัดรูปร่างกลุ่มของแต่ละคลัสเตอร์มีความชัดเจนและเป็นรูปร่างที่สมบูรณ์มากน้อยแค่ไหนซึ่งโมเดลที่ได้ silhouette score เข้าใกล้ 1 แสดงว่ามีการจัดกลุ่มได้อย่างสมบูรณ์

```
from sklearn.metrics.cluster import silhouette_score

customer_pca = customer_pca.iloc[:, 0:9]
algorithms = [KMeans(n_clusters = 3, n_init=100, random_state=RANDOM_STATE),
              AgglomerativeClustering(n_clusters = 3, affinity='euclidean', linkage='ward'),
              DBSCAN(eps=2, min_samples = 6)]
random_state = np.random.RandomState(seed=0)
random_clusters = random_state.randint(low=0, high=2, size=len(customer_final))
```

```
#random assignment
print("Random assignment - silhouette score: {:.2f}".format(silhouette_score(customer_pca, random_clusters)))
Random assignment - silhouette score: 0.00

for algorithm in algorithms:
    clusters = algorithm.fit_predict(customer_pca)
    print("{} : {:.2f}".format(algorithm.__class__.__name__, silhouette_score(customer_pca, clusters)))
KMeans : 0.19
AgglomerativeClustering : 0.17
DBSCAN : 0.14
```

หลังจากการเปรียบเทียบแต่ละโมเดลด้วย silhouette score พบร่วมกันว่า Kmeans มีการจัดกลุ่มคลัสเตอร์ได้เป็นรูปร่างและสัดส่วนชัดเจนมากสุดคือ 0.19 เปอร์เซ็นต์ รองลงมาคือ Agglomerative clustering และ DBSCAN ตามลำดับ

บทที่ 4

สรุปผลและข้อเสนอแนะ

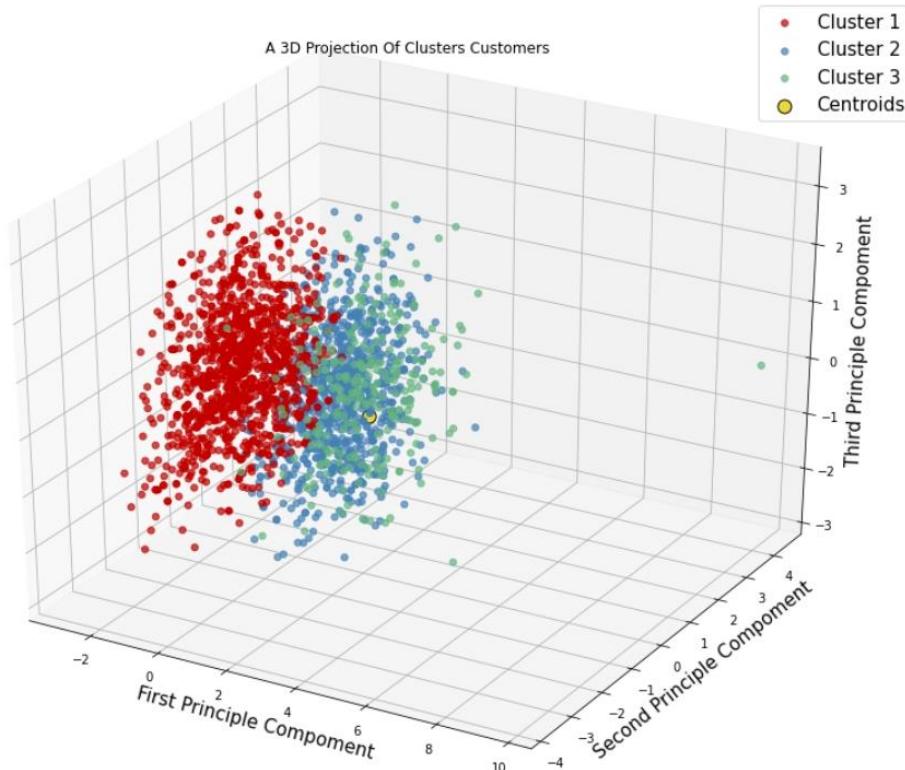
1. สรุปผลและอภิปรายผล

จากการทำ Data Exploration และ Visualize ชุดข้อมูล customer_pca พบว่าชุดข้อมูลนี้รูป่างจัดกลุ่มมีความเป็นวงกลมและเมื่อทำการประเมินผลของโมเดลในการทำ clustering ทั้ง 3 วิธี สรุปได้ว่า โมเดลที่เหมาะสมและเลือกใช้ในการจัดกลุ่มลูกค้าตามชุดข้อมูล คือการทำ K-Means Clustering ที่มีพารามิเตอร์ ดังนี้

```
kmeans = KMeans(n_clusters = 3, n_init=100, random_state=RANDOM_STATE)
kmeans.fit(customer_pca)

KMeans(n_clusters=3, n_init=100, random_state=20)
```

ซึ่งสามารถจัดกลุ่มคลัสเตอร์ได้ทั้งหมด 3 คลัสเตอร์ ดังรูป

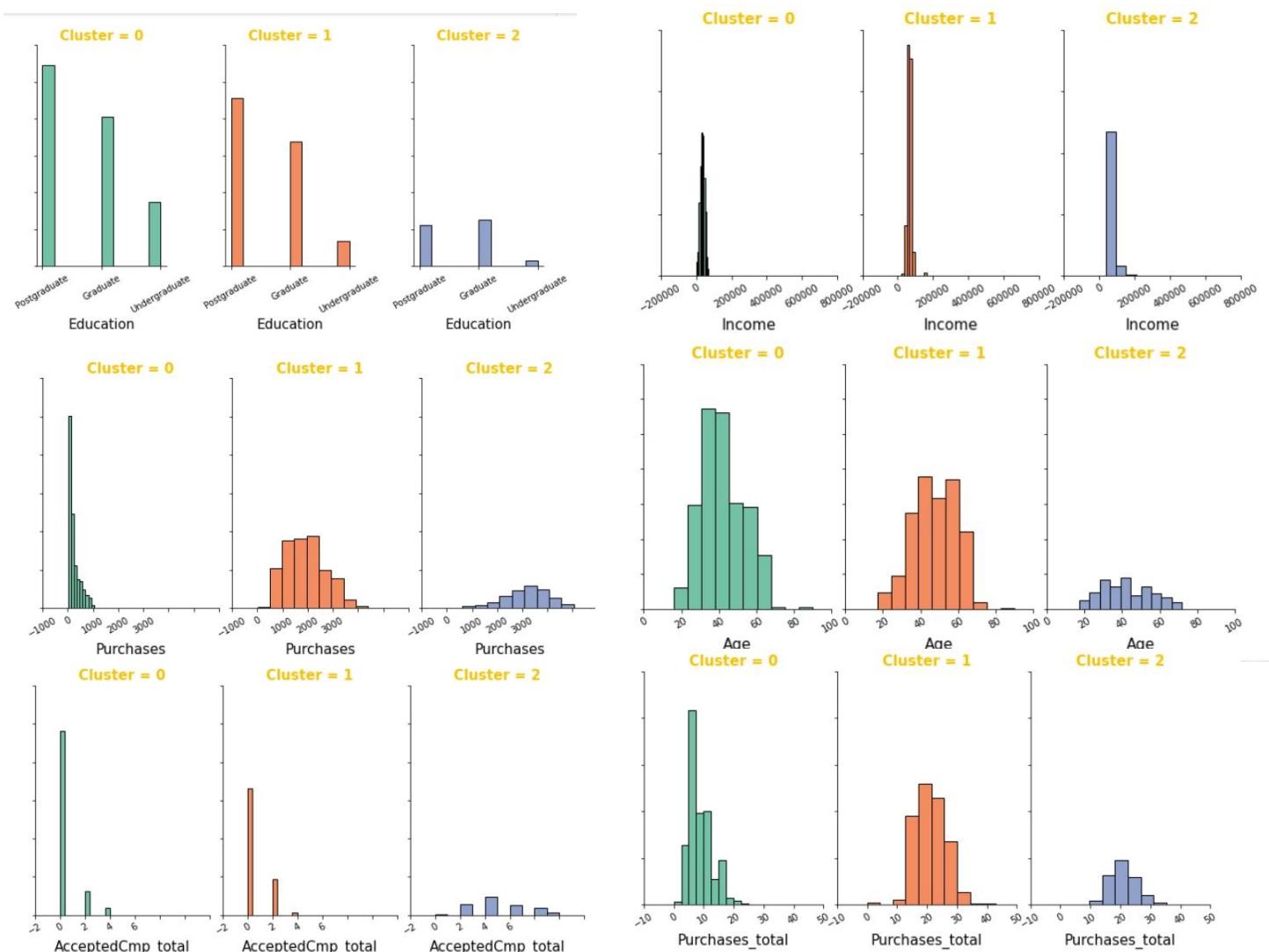


เนื่องจากทำการ visualize เพื่อศึกษาความสัมพันธ์ของการจัดกลุ่มคลัสเตอร์แต่ละคลัสเตอร์กับฟีเจอร์แต่ละอันได้ดังนี้ สรุปได้ว่า

- คลัสเตอร์ที่ 1 มีลักษณะเด่นของกลุ่มคือ กลุ่มรายได้น้อยถึงปานกลาง อยู่ในช่วงวัยทำงาน มีการจบการศึกษาและไม่จบการศึกษาปะปนกัน มีจำนวนการซื้อสินค้าน้อยและมีการเข้าร่วมแคมเปญและรับโปรโมชันน้อย จึงควรมีการสร้างแรงจูงใจในการซื้อเพิ่มมากขึ้น โดยการให้ส่วนลด หรือ แคมเปญที่คุ้มค่าแก่กลุ่มลูกค้านี้

- คลัสเตอร์ที่ 2 มีลักษณะเด่นของกลุ่มคือ กลุ่มรายได้ปานกลางถึงมาก อยู่ในช่วงวัยกลางคน ส่วนมากเป็นผู้ที่มีการศึกษาระดับสูง มีจำนวนการซื้อสินค้าที่เยอะ แต่ไม่ค่อยเข้าร่วมแคมเปญหรือสิทธิพิเศษที่จัดให้ จึงควรมีสิทธิพิเศษสำหรับลูกค้ากลุ่มนี้ เพราะถือเป็นลูกค้ากลุ่มสำคัญ ควรมีโปรโมชันหรือแคมเปญสำหรับตอบแทนลูกค้าที่ซื้อจำนวนเยอะๆ ในอนาคตอาจจะเป็น loyal customer นั่นเอง

- คลัสเตอร์ที่ 3 มีลักษณะเด่นของกลุ่มคือ กลุ่มรายได้สูง อยู่ในช่วงวัยกลางคน เป็นผู้มีการจบการศึกษาระดับสูงเกือบทั้งหมด มีจำนวนการซื้อเยอะมากและมีการเข้าร่วมแคมเปญตลอด จึงควรมีบัตรกำนัลสิทธิพิเศษหรือของสมนาคุณสำหรับลูกค้าที่เข้าร่วมแคมเปญ หรือซื้อสินค้าจำนวนมาก เพื่อกระตุ้นให้ลูกค้ามีการซื้อมากขึ้นและมีความ loyal customer หากว่ามากขึ้น



2. ประโยชน์ที่ได้รับ

1. สามารถนำไปใช้ในการแบ่งกลุ่มลูกค้าเพื่อไปใช้ในการวิเคราะห์การตลาดและตอบสนองลูกค้าได้ตามกลุ่มเป้าหมาย
2. สามารถเพิ่มยอดขายและดึงดูดความสนใจจากลูกค้าได้เมื่อรู้ว่ากลุ่มลูกค้าแต่ละกลุ่มมีจุดเด่นอะไรบ้าง
3. สามารถนำโมเดลไปปรับใช้ในการตลาดและธุรกิจต่างๆ รวมถึงอาจนำไปใช้กับชุดข้อมูลอื่นๆ
4. ได้ฝึกฝนและเพิ่มทักษะในการทำ Supervised Learning model มากยิ่งขึ้น
5. พัฒนาการทำงานและรู้จักสังเกตชุดข้อมูล วิเคราะห์ชุดข้อมูลได้ลึกซึ้งมากยิ่งขึ้น

3. ข้อเสนอแนะเพิ่มเติม

1. ควรกำจัด outlier ให้อย่างครอบคลุมเพื่อนำไปสร้างโมเดลแบ่งคลัสเตอร์ได้อย่างถูกต้องสมบูรณ์ขึ้น
2. ถ้าใช้ข้อมูลที่มีขนาดใหญ่มากขึ้น มีพีเจอร์ที่นำเสนอจะเพิ่มขึ้นอาจทำให้แบ่งคลัสเตอร์ได้อย่างสมเหตุสมผลมากยิ่งขึ้น