Walmart Case Study

By Gautam Naik (gautamnaik1994@gmail.com)
Github: https://github.com/gautamnaik1994/Walmart-Data-Analysis-case-Study

About Walmart

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

Metric

- We will use visualization of all variable with purchase amount to visually confirm our findings.
- Central Limit Theorem with bootstrapping will be used to estimate the confidence interval.
- Hypothesis testing will be done on Gender, Marital Status and Age using z test and t test to answer question about spending patterns

Dataset

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features:

- User_ID:User ID
- Product_ID:Product ID
- Gender:Sex of User
- Age:Age in bins
- Occupation:Occupation(Masked)
- City_Category:Category of the City (A,B,C)
- StayInCurrentCityYears:Number of years stay in current city
- Marital_Status: Marital Status
- ProductCategory: Product Category (Masked)
- Purchase:Purchase Amount

Table of contents

Data Cleaning and Manipulations

In [ ]:
```
import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import pickle
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm,ttest_ind,ttest_1samp,ttest_rel,f_oneway
import concurrent.futures
sns.set_style("whitegrid")
import duckdb as db
```

In [ ]:
```
df=pd.read_csv('./walmart_data.csv')
df.shape
df.head()
df.info()
```

Out[ ]:
```
(550068, 10)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

Observations

- No null values are present in the dataset.

In [ ]:
```
df.describe()
```

Out[ ]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [ ]:
```
columns=["User_ID","Product_ID","Gender","Age","Occupation","City_Category",\
         "Stay_In_Current_City_Years","Marital_Status","Product_Category","Purchase"]
category_columns=["Gender","City_Category","Marital_Status","Product_Category", "Age",\
                  "Occupation", "User_ID","Product_ID"]
```

In [ ]:
```
print("Unique Values in dataset")
for col in columns:
    print(f"{col} : {df[col].nunique()}")
```

```
Unique Values in dataset
User_ID : 5891
Product_ID : 3631
Gender : 2
Age : 7
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category : 20
Purchase : 18105
```

In [ ]:
```
df["Age"].value_counts()
```

Out[ ]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

In [ ]:
```
df["Occupation"].value_counts()
```

Out[ ]:
```
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
```

In [ ]:
```
df["Stay_In_Current_City_Years"].value_counts()
```

Out[ ]:
```
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: count, dtype: int64
```

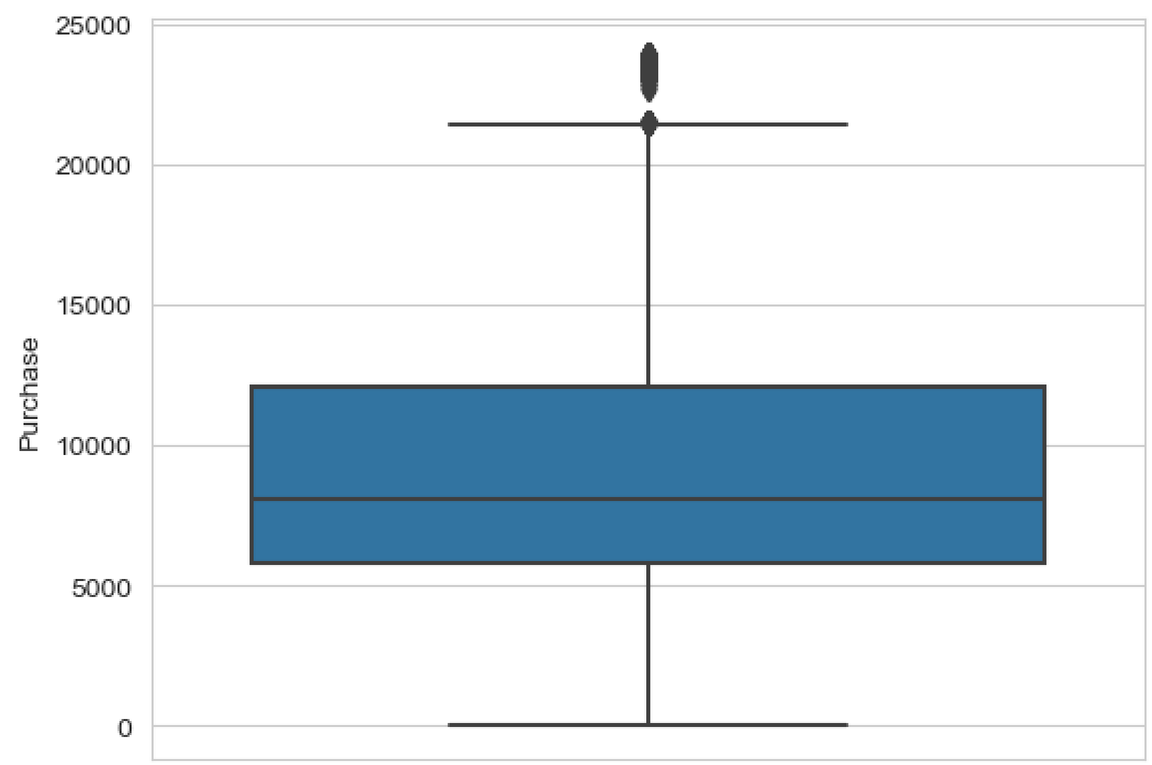
In [ ]:
```
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].replace(to_replace = "4+", value = "4")
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].astype(np.int8)
```

In [ ]:
```
df["Purchase"] = df["Purchase"].astype("int32")
# df["User_ID"] = df["User_ID"].astype("int32")
```

In [ ]:
```
for col in category_columns:
    df[col] = df[col].astype('category')
    # df[col] = df[col].cat.codes
```

In [ ]:
```
sns.boxplot( y="Purchase", data=df)
```

Out[ ]:
```
<Axes: ylabel='Purchase'>
```



Observations

- There is some outlier data in the dataset.
- We will remove the outlier data by dropping the bottom 5 percentile and above 95 percentile values of Purchase data

In [ ]:
```
five_percentile = np.percentile(df["Purchase"],[5])[0]
ninetyfive_percentile = np.percentile(df["Purchase"],[95])[0]
five_percentile, ninetyfive_percentile
```
Out[ ]:
```
(1984.0, 19336.0)
```

In [ ]:
```
df=df.loc[(df["Purchase"] > five_percentile) & (df["Purchase"] < ninetyfive_percentile)]
```

In [ ]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Index: 495033 entries, 0 to 545914
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     495033 non-null  category
 1   Product_ID                  495033 non-null  category
 2   Gender                      495033 non-null  category
 3   Age                         495033 non-null  category
 4   Occupation                  495033 non-null  category
 5   City_Category               495033 non-null  category
 6   Stay_In_Current_City_Years  495033 non-null  int8
 7   Marital_Status              495033 non-null  category
 8   Product_Category            495033 non-null  category
 9   Purchase                    495033 non-null  int32
dtypes: category(8), int32(1), int8(1)
memory usage: 11.2 MB
```

In [ ]:
```
df.describe(include='category')
```
Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Marital_Status	Product_Category
count	495033	495033	495033	495033	495033	495033	495033	495033
unique	5891	3546	2	7	21	3	2	16
top	1001680	P00265242	M	26-35	4	B	0	5
freq	937	1843	372284	198375	64892	209348	292453	143987

In [ ]:
```
df.describe()
```
Out[ ]:

	Stay_In_Current_City_Years	Purchase
count	495033.000000	495033.000000
mean	1.859179	9100.472771
std	1.289395	4193.962676
min	0.000000	1985.000000
25%	1.000000	5987.000000
50%	2.000000	8047.000000
75%	3.000000	11796.000000
max	4.000000	19335.000000

In [ ]:
```
unique_users=df.drop_duplicates(subset='User_ID')
```

In [ ]:
```
unique_users.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Index: 5891 entries, 0 to 243533
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   User_ID                     5891 non-null   category
 1   Product_ID                  5891 non-null   category
 2   Gender                      5891 non-null   category
 3   Age                         5891 non-null   category
 4   Occupation                  5891 non-null   category
 5   City_Category               5891 non-null   category
 6   Stay_In_Current_City_Years  5891 non-null   int8
 7   Marital_Status              5891 non-null   category
 8   Product_Category            5891 non-null   category
 9   Purchase                    5891 non-null   int32
dtypes: category(8), int32(1), int8(1)
memory usage: 466.9 KB
```

In [ ]:
```
with open("data.pickle", "wb") as f:
    pickle.dump(df, f)

with open("unique_users.pickle", "wb") as f:
    df = pickle.dump(unique_users, f)
```

Exploratory Data Analysis

In [ ]:
```
with open("./data.pickle", "rb") as f:
    df = pickle.load(f)

with open("./unique_users.pickle", "rb") as f:
    unique_users = pickle.load(f)
```

Univariate Analysis

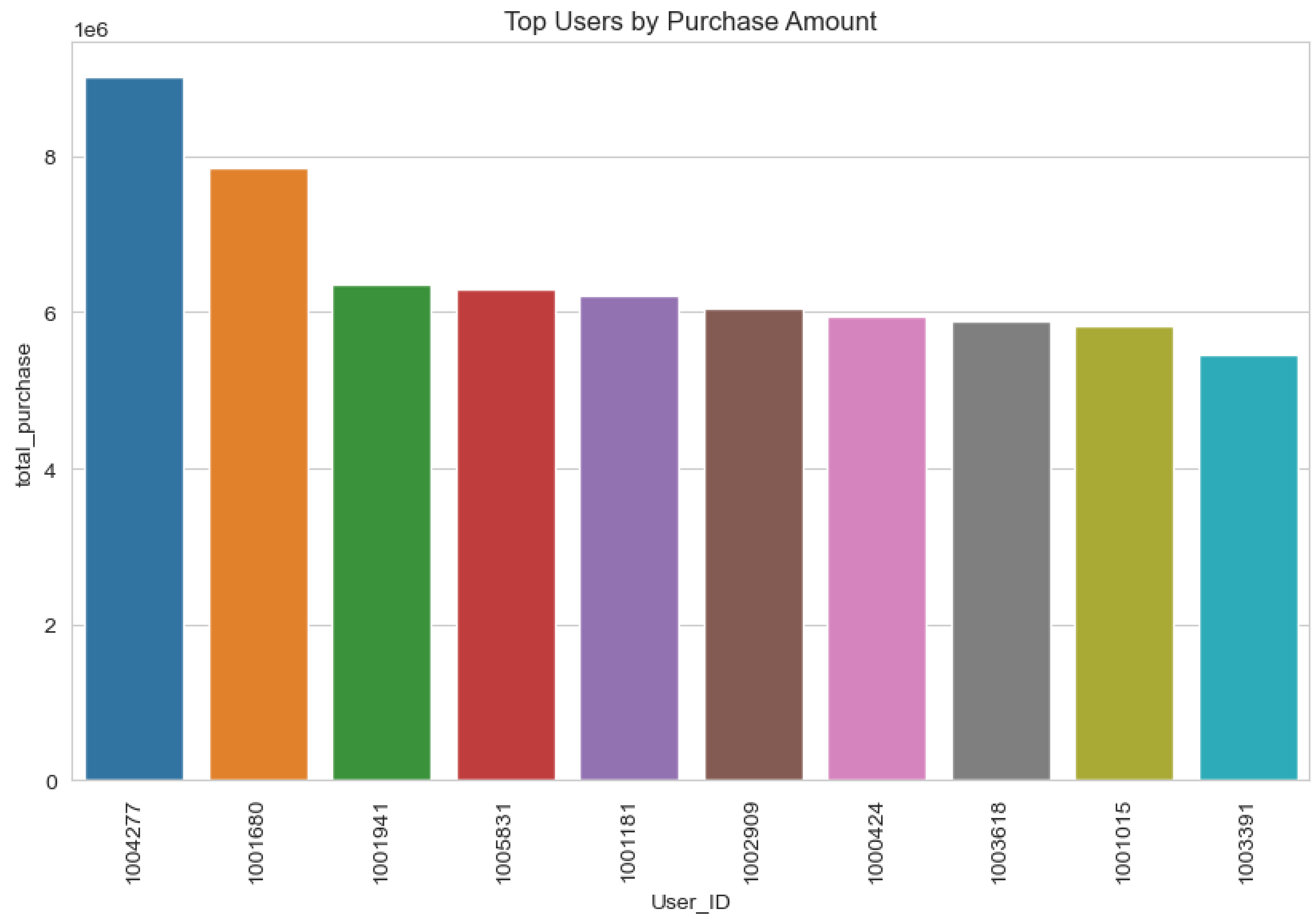
In [ ]:
```
top_users=db.sql("""
    select User_ID, sum(Purchase) as total_purchase from df group by
        User_ID order by total_purchase desc limit 10
""").to_df()
```

In [ ]:
```
fig, axes = plt.subplots(3, 3, figsize=(20, 18))

sns.countplot(x="City_Category", data=unique_users, ax=axes[0, 0])
axes[0, 0].set_title("Top City categories");

sns.countplot(x="Occupation", data=unique_users , ax=axes[0, 1])
axes[0, 1].set_title("Top Occupation categories");

sns.countplot(x="Age", data=unique_users, ax=axes[0,2])
axes[0,2].set_title("Top Age categories");

unique_users["Marital_Status"].value_counts(normalize=True )[:10].plot(kind="pie", autopct='%1.1f%%', startangle=90, ax=axes[1,1])
axes[1,1].set_title("Marital_Status Distribution");

unique_users["Gender"].value_counts(normalize=True )[:10].plot(kind="pie", autopct='%1.1f%%', startangle=90, ax=axes[1,0])
axes[1,0].set_title("Gender Distribution");

sns.countplot(x="Stay_In_Current_City_Years", data=unique_users, ax=axes[1,2])
axes[1,2].set_title("Stay_In_Current_City_Years Distribution");

sns.countplot(x="Product_Category", data=df, order=df["Product_Category"].value_counts().iloc[:10].index, ax=axes[2,0])
axes[2,0].set_title("Top Product_Category By Count");

sns.countplot(x="Product_ID", data=df, order=df["Product_ID"].value_counts().iloc[:10].index, ax=axes[2,1])
axes[2,1].set_title("Top Product_ID By Count");
plt.setp(axes[2,1].get_xticklabels(), rotation=90)

sns.histplot(x="Purchase", data=df, kde=False, ax=axes[2 , 2])
axes[2,2].set_title("Purchase Ammount Distribution");

plt.show();
```

Observations

- City Category **C** has the highest count of users
- Most users belong to Occupation category **4, 7 and 0**
- Majority of the users belong to 26-35 age bracket
- There are 71.7 percent male users as compared 28.8 female users
- 42% of the users are married while 58% are unmarried
- Majority of the users stay in the current city for 1 year
- Product Category **5, 1, 8** are the most popular product category
- Majority of user pay in 5000 - 10000 dollars range

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x="User_ID", y="total_purchase" , data=top_users, order=top_users["User_ID"])
plt.title("Top Users by Purchase Amount");
# rotate x tick labels
plt.xticks(rotation=90);
```

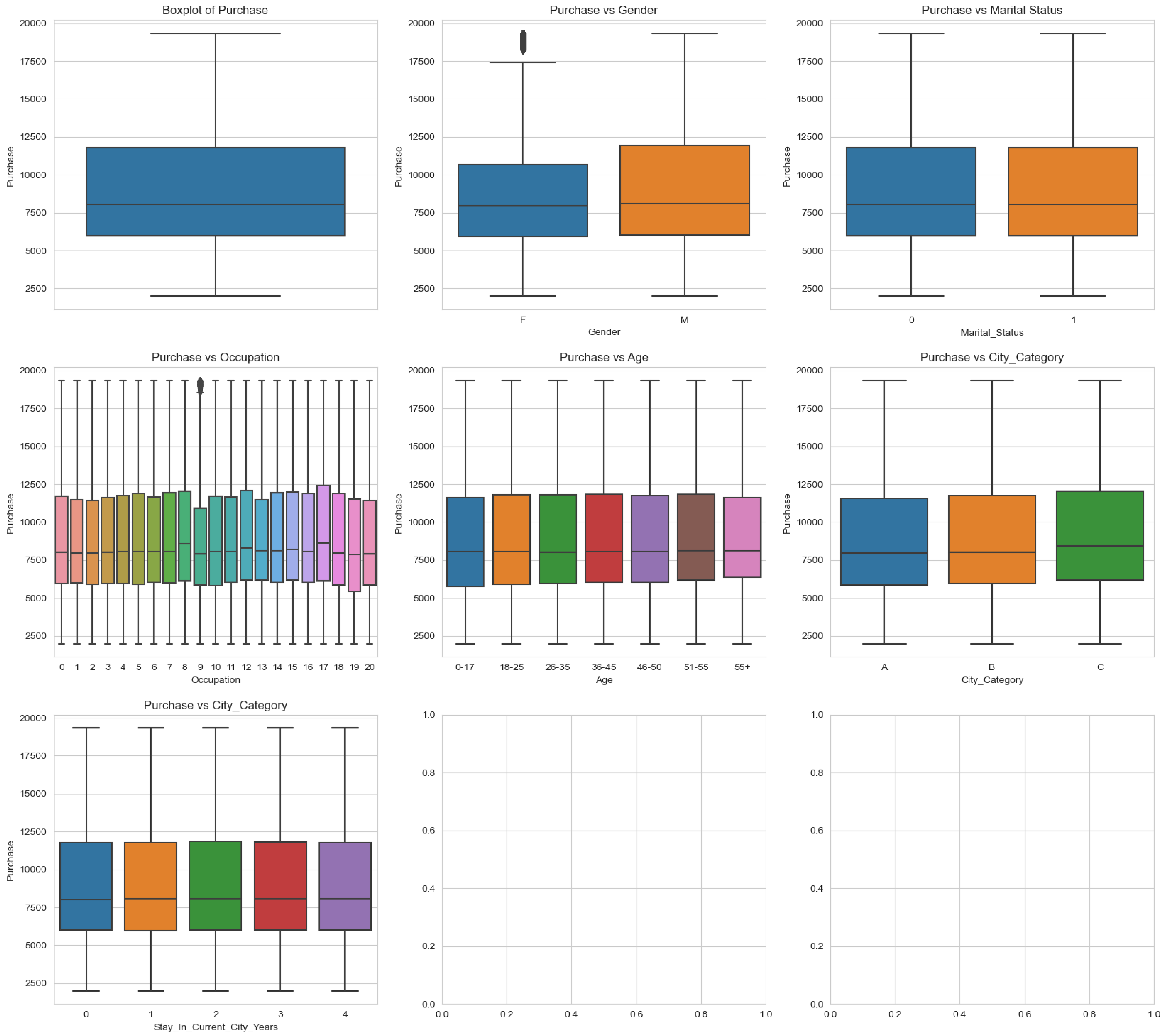


Observations

- Above table shows the top 10 users who purchased the most products

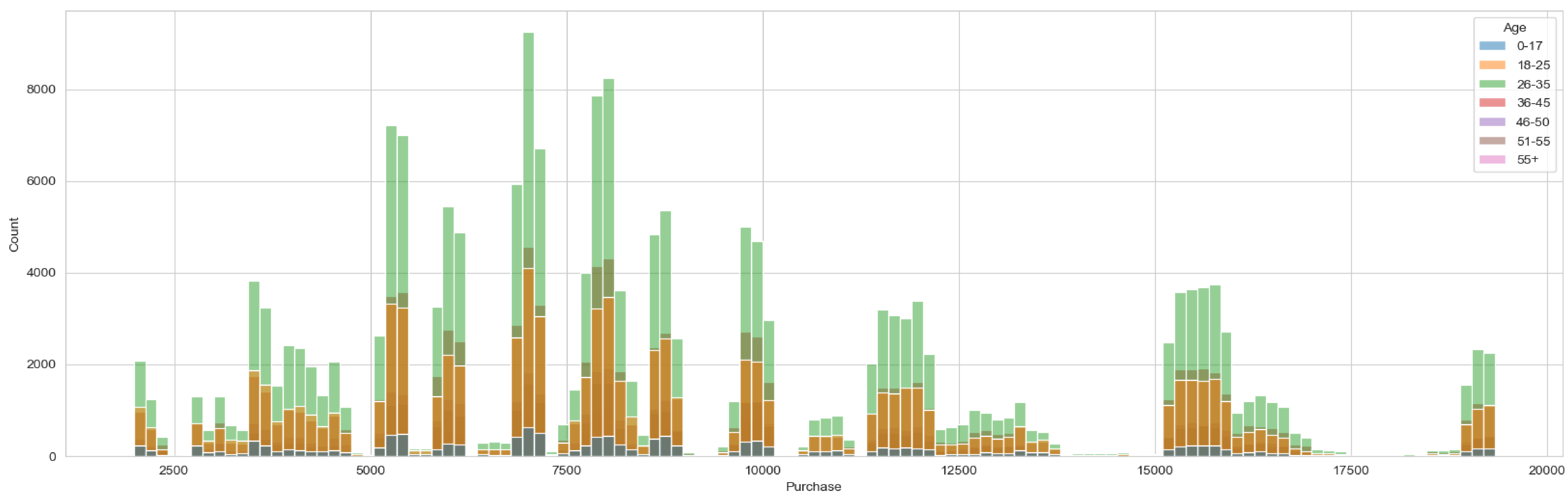
Bivariate Analysis

In [ ]:
```
fig, axes = plt.subplots(3, 3, figsize=(20, 18))
sns.boxplot(data=df, y="Purchase", ax=axes[0,0])
axes[0,0].set_title("Boxplot of Purchase");

sns.boxplot(data=df, y="Purchase", x="Gender", ax=axes[0,1])
axes[0,1].set_title("Purchase vs Gender");

sns.boxplot(data=df, y="Purchase", x="Marital_Status", ax=axes[0,2])
axes[0,2].set_title("Purchase vs Marital Status");

sns.boxplot(data=df, x="Occupation", y="Purchase", ax=axes[1,0]);
axes[1,0].set_title("Purchase vs Occupation");

sns.boxplot(data=df, x="Age", y="Purchase", ax=axes[1,1]);
axes[1,1].set_title("Purchase vs Age");

sns.boxplot(data=df, x="City_Category", y="Purchase", ax=axes[1,2]);
axes[1,2].set_title("Purchase vs City_Category");

sns.boxplot(data=df, x="Stay_In_Current_City_Years", y="Purchase", ax=axes[2,0]);
axes[2,0].set_title("Purchase vs City_Category");
```



- No particular pattern is present when purchase data is plotted with other categorical variables.
- Box plots indicate the presense of outlier data.

In [ ]:
```
plt.figure(figsize=(20, 6))
sns.histplot(data=df, x="Purchase", hue="Age");
```

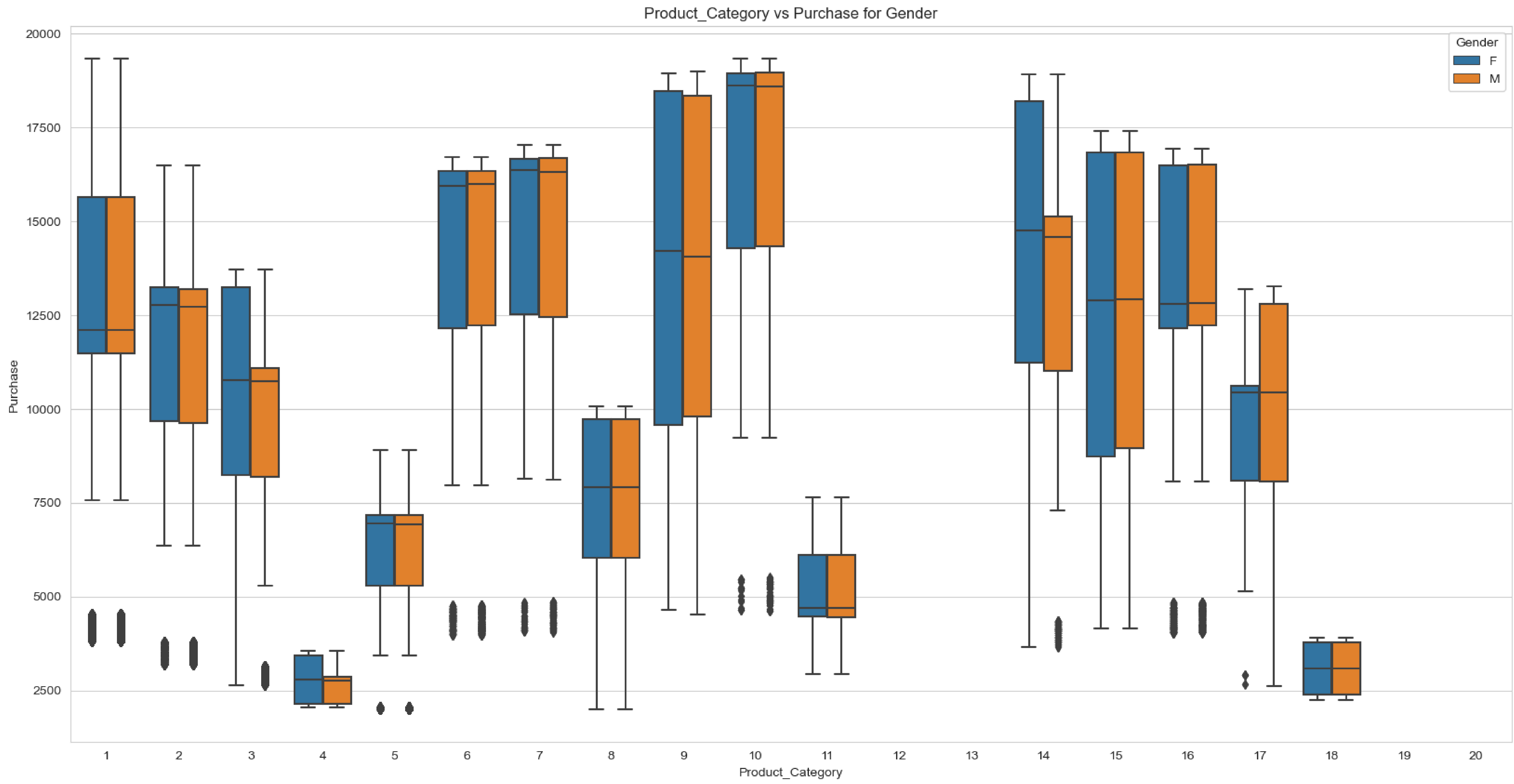


Observations

- Above plot shows purchase data distribution for all age groups.
- Majority of the spending is between 5000 and 10000 dollars.

Multi-variate Analysis

In [ ]:
```
plt.figure(figsize=(20, 10))
sns.boxplot(data=df, x="Product_Category", y="Purchase", hue="Gender");
plt.title("Product_Category vs Purchase for Gender");
```

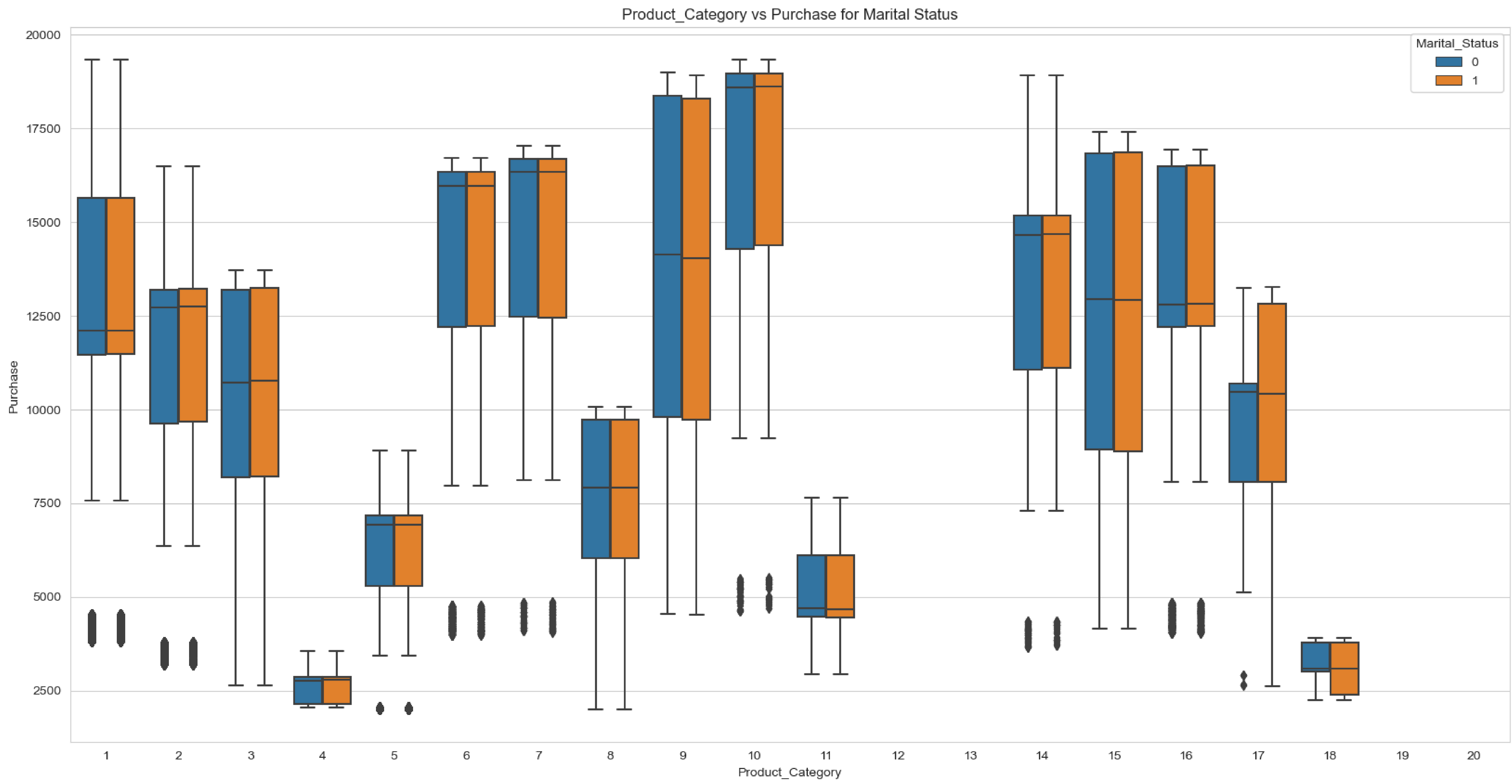


Insights

- Product Category 3,15 and 14 has higher female users
- Product Category 17 has higher male users

In [ ]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(data=df, x="Product_Category", y="Purchase", hue="Marital_Status");
plt.title("Product_Category vs Purchase for Marital Status");
```

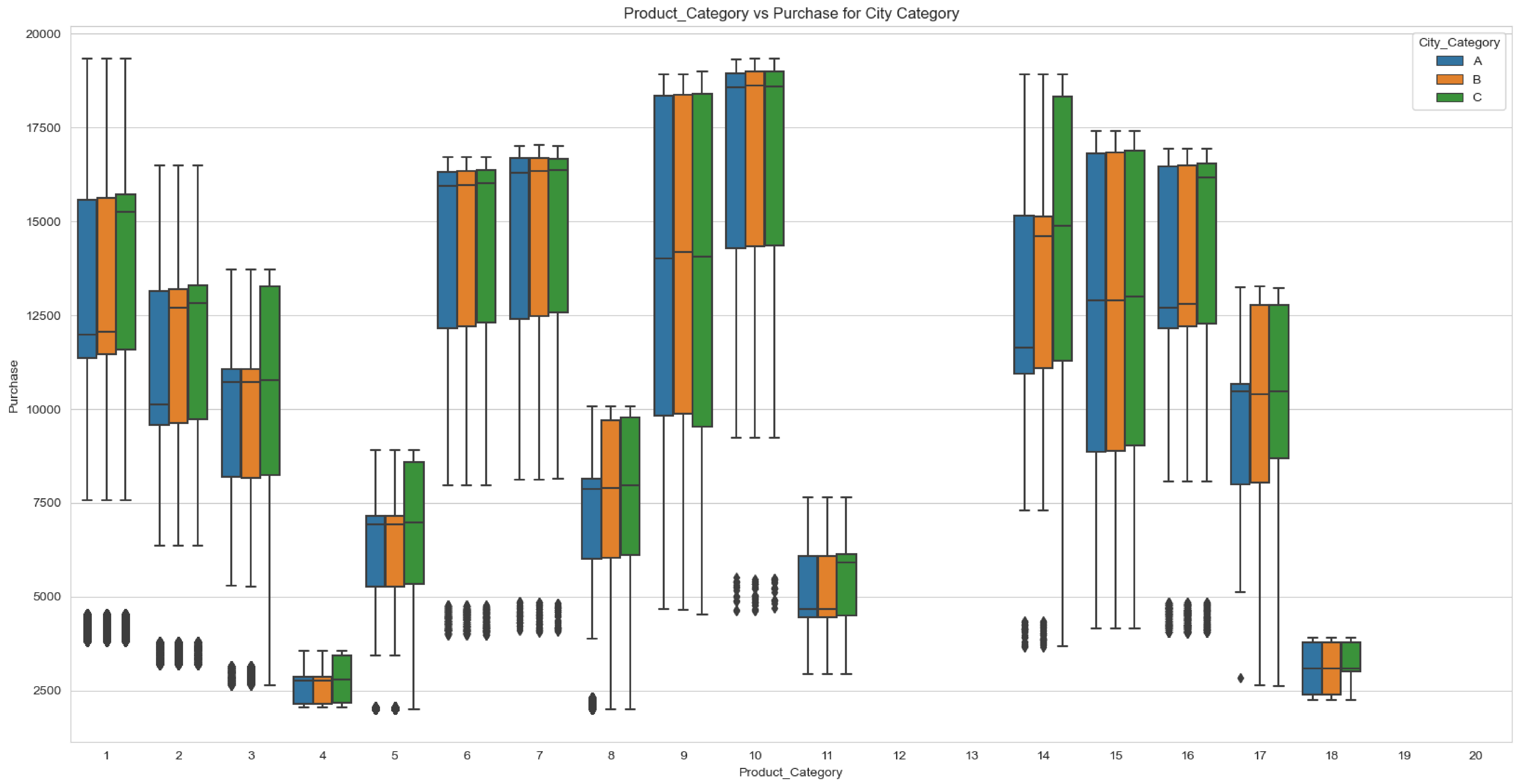


Insights

- Product Category 9 and 17 is more popular among Married users

In [ ]:

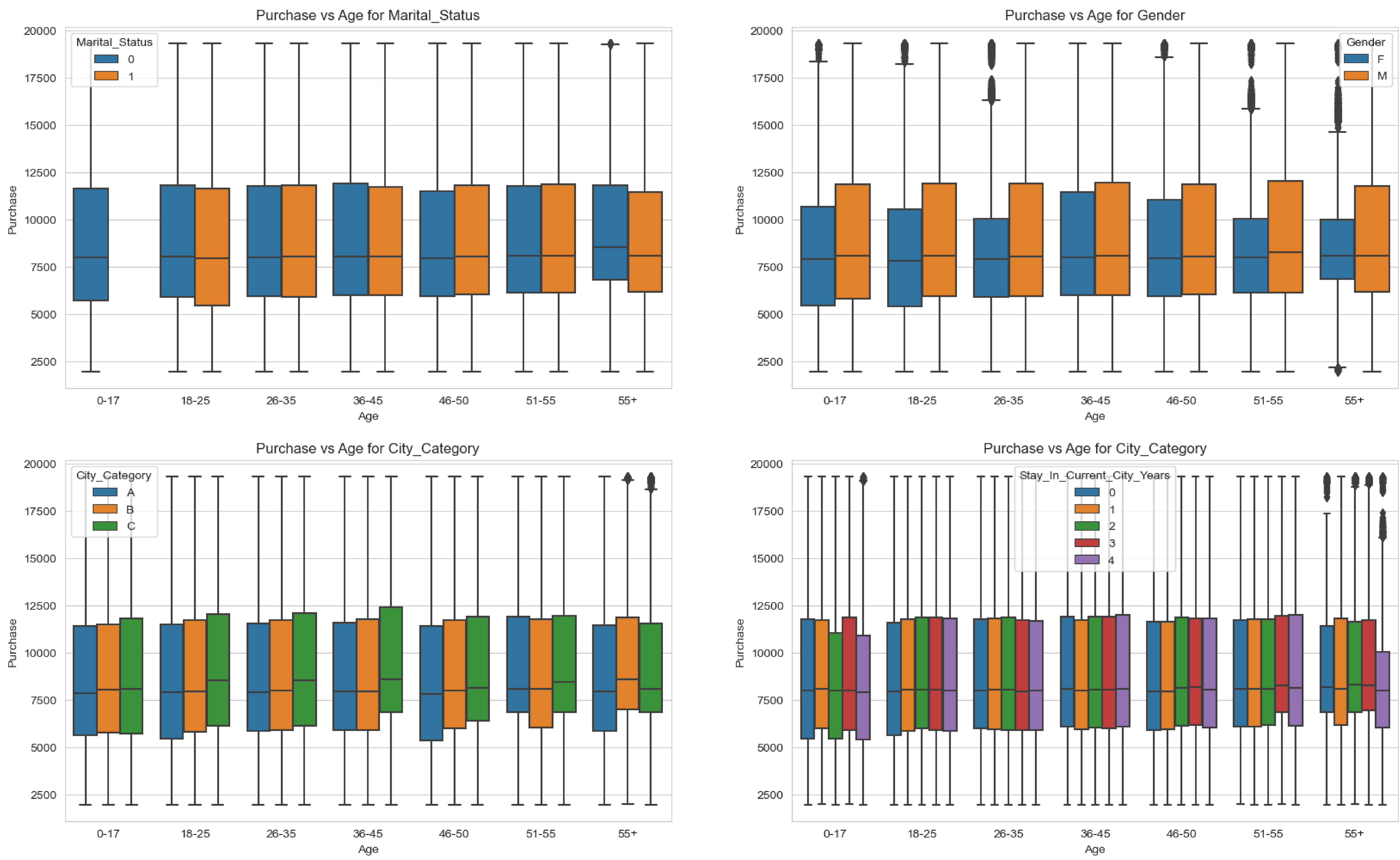
```
plt.figure(figsize=(20, 10))
sns.boxplot(data=df, x="Product_Category", y="Purchase", hue="City_Category");
plt.title("Product_Category vs Purchase for City Category");
```



Insights

- Product Category 14 is most popular in Category C city
- Demand for all product category in all cities seems to be equal

In [ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(20, 12))

sns.boxplot(data=df, x="Age", y="Purchase", hue="Marital_Status", ax=axes[0,0] );
axes[0,0].set_title("Purchase vs Age for Marital_Status");

sns.boxplot(data=df, x="Age", y="Purchase", hue="Gender", ax=axes[0,1]);
axes[0,1].set_title("Purchase vs Age for Gender");

sns.boxplot(data=df, x="Age", y="Purchase", hue="City_Category", ax=axes[1,0]);
axes[1,0].set_title("Purchase vs Age for City_Category");

sns.boxplot(data=df, x="Age", y="Purchase", hue="Stay_In_Current_City_Years", ax=axes[1,1]);
axes[1,1].set_title("Purchase vs Age for City_Category");
```



Insights

- Female user of age group 55+ appear to spend very less compared to males
- User in City Categoty 4 and 55+ age group have lower purchase amount

In [ ]:
```
count_wise_data = db.sql("""
SELECT Age, Product_Category, cnt, rnk_cnt
FROM (
    SELECT
        Age, Product_Category, COUNT(*) AS cnt,
        RANK() OVER (PARTITION BY Age ORDER BY cnt DESC) AS rnk_cnt,
    FROM df
```

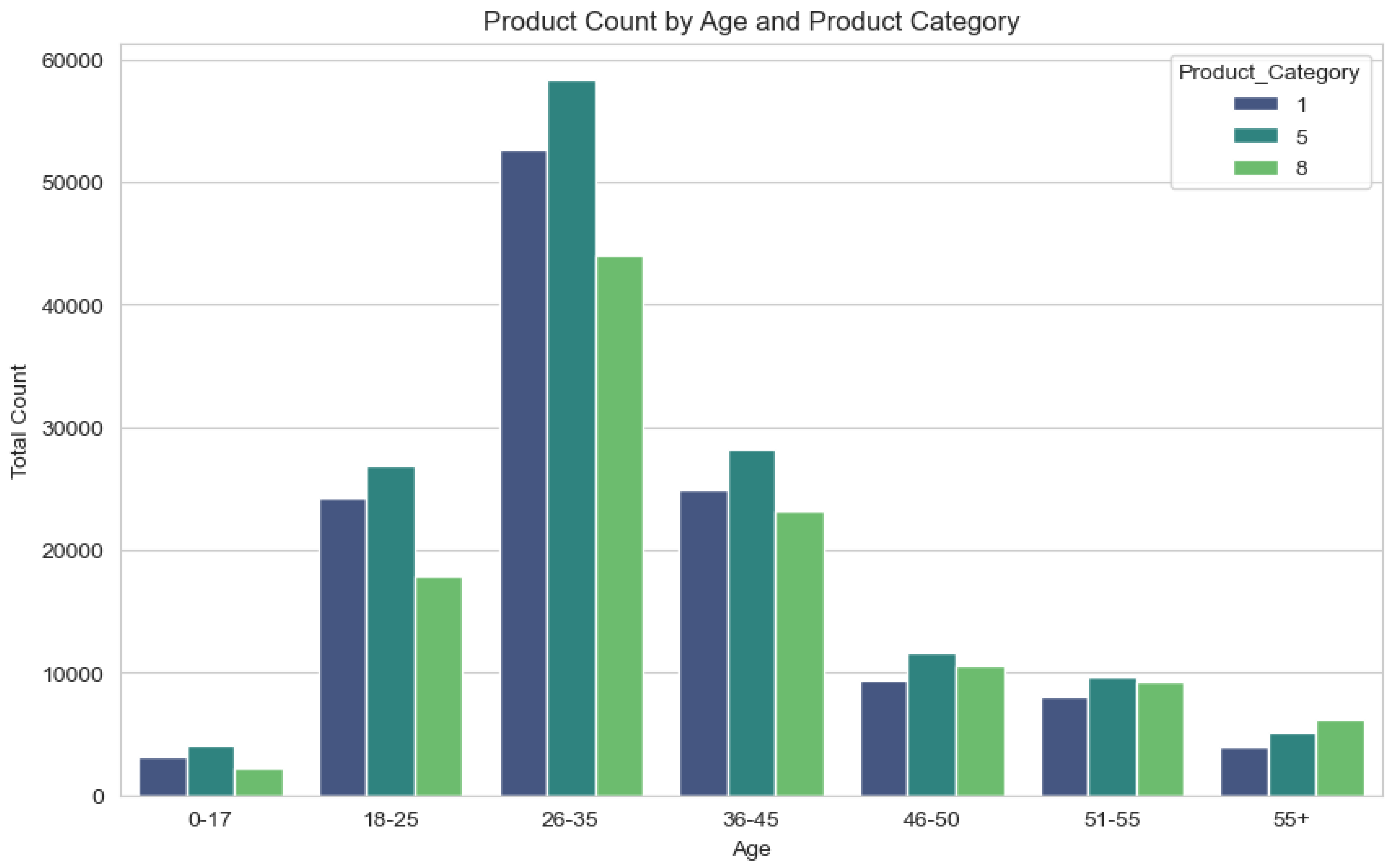
```
    GROUP BY Age, Product_Category
) AS ranked_data
WHERE rnk_cnt <=3 order by Age, rnk_cnt;
""").to_df()
count_wise_data
```

Out[ ]:

	Age	Product_Category	cnt	rnk_cnt
0	0-17	5	4067	1
1	0-17	1	3193	2
2	0-17	8	2252	3
3	18-25	5	26919	1
4	18-25	1	24199	2
5	18-25	8	17823	3
6	26-35	5	58411	1
7	26-35	1	52684	2
8	26-35	8	44067	3
9	36-45	5	28234	1
10	36-45	1	24964	2
11	36-45	8	23222	3
12	46-50	5	11579	1
13	46-50	8	10621	2
14	46-50	1	9411	3
15	51-55	5	9617	1
16	51-55	8	9319	2
17	51-55	1	8043	3
18	55+	8	6194	1
19	55+	5	5160	2
20	55+	1	3916	3

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Age', y='cnt', hue='Product_Category', data=count_wise_data, palette='viridis')
plt.xlabel('Age')
plt.ylabel('Total Count')
plt.title('Product Count by Age and Product Category');
```



Insights

- Above plot shows top products in terms of count for each age group
- Product category 5 is popular product category for all age groups except 55+
- Product category 8 is most popular product category for 55+ age group

In [ ]:

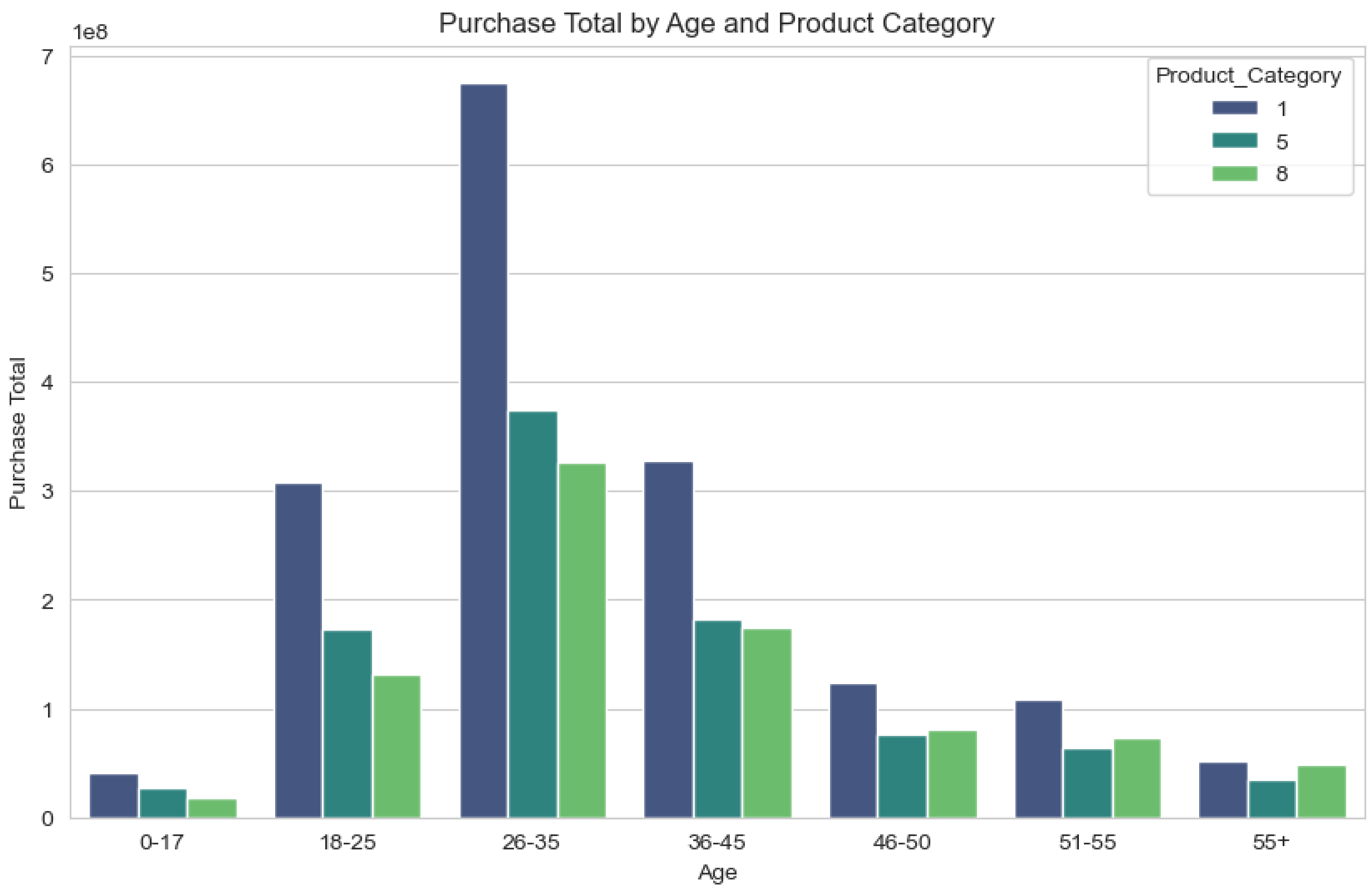
```
purchase_wise_data = db.sql("""
SELECT Age, Product_Category, purchase_total, rnk_total
FROM (
    SELECT
        Age, Product_Category, SUM(Purchase) AS purchase_total,
        RANK() OVER (PARTITION BY Age ORDER BY purchase_total DESC) AS rnk_total
    FROM df
    GROUP BY Age, Product_Category
) AS ranked_data
WHERE rnk_total <= 3  order by Age, rnk_total
""").to_df()
purchase_wise_data
```

Out[ ]:

	Age	Product_Category	purchase_total	rnk_total
0	0-17	1	41130457.0	1
1	0-17	5	26573050.0	2
2	0-17	8	17223031.0	3
3	18-25	1	308668603.0	1
4	18-25	5	172230397.0	2
5	18-25	8	132147662.0	3
6	26-35	1	675174790.0	1
7	26-35	5	374041907.0	2
8	26-35	8	327152697.0	3
9	36-45	1	328241666.0	1
10	36-45	5	182463537.0	2
11	36-45	8	175240903.0	3
12	46-50	1	123558710.0	1
13	46-50	8	80199149.0	2
14	46-50	5	75554504.0	3
15	51-55	1	108184920.0	1
16	51-55	8	72550223.0	2
17	51-55	5	63816994.0	3
18	55+	1	52380275.0	1
19	55+	8	48967858.0	2
20	55+	5	34305815.0	3

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Age', y='purchase_total', hue='Product_Category', data=purchase_wise_data, palette='viridis')
plt.xlabel('Age')
plt.ylabel('Purchase Total')
plt.title('Purchase Total by Age and Product Category');
```



Insights

- In terms of total purchase amount Product category 1 appears to be the most profitable category among all age groups
- This data can be used to manage the inventory for the stores

In [ ]:

```
purchase_data=db.sql("""
SELECT User_ID, Gender, count(*) cnt, avg(Purchase) avg_purchase
from df group by User_ID, Gender order by cnt desc;
""").to_df()
print("Male")
purchase_data.query("Gender == 'M'").head(10)
print("Female")
purchase_data.query("Gender == 'F'").head(10)
```

Male

Out[ ]:

	User_ID	Gender	cnt	avg_purchase
0	1001680	M	937	8386.797225
1	1004277	M	881	10224.961407
2	1001181	M	817	7609.392901
3	1001941	M	793	8019.283733
4	1000889	M	760	6765.482895
5	1003618	M	737	7998.316147
6	1005831	M	696	9051.349138
7	1001015	M	678	8576.731563
8	1002063	M	659	7503.135053
10	1000424	M	641	9271.140406

Female

Out[ ]:

	User_ID	Gender	cnt	avg_purchase
9	1001150	F	653	6809.712098
17	1001088	F	624	8465.769231
21	1003224	F	591	9069.399323
23	1003539	F	563	9558.509769
25	1001605	F	544	7561.470588
27	1001448	F	540	8810.890741
35	1005643	F	525	8216.163810
36	1000752	F	518	7032.204633
50	1006036	F	478	7862.179916
56	1002820	F	472	7531.603814

Insights

- Both data table shows the average purchase amount and count of sales for top 10 male and female users
- We can see that there is a big difference in count of sales between male and female users.

In [ ]:

```
top_products = db.sql("""
SELECT Product_ID, Product_Category, purchase_total
FROM (
    SELECT
        Product_ID, Product_Category, SUM(Purchase) AS purchase_total,
        RANK() OVER (PARTITION BY Product_Category ORDER BY purchase_total DESC) AS rnk_total
    FROM df
    GROUP BY Product_ID, Product_Category
) AS ranked_data
WHERE rnk_total =1  order by purchase_total desc, Product_Category
""").to_df()
top_products
```

Out[ ]:

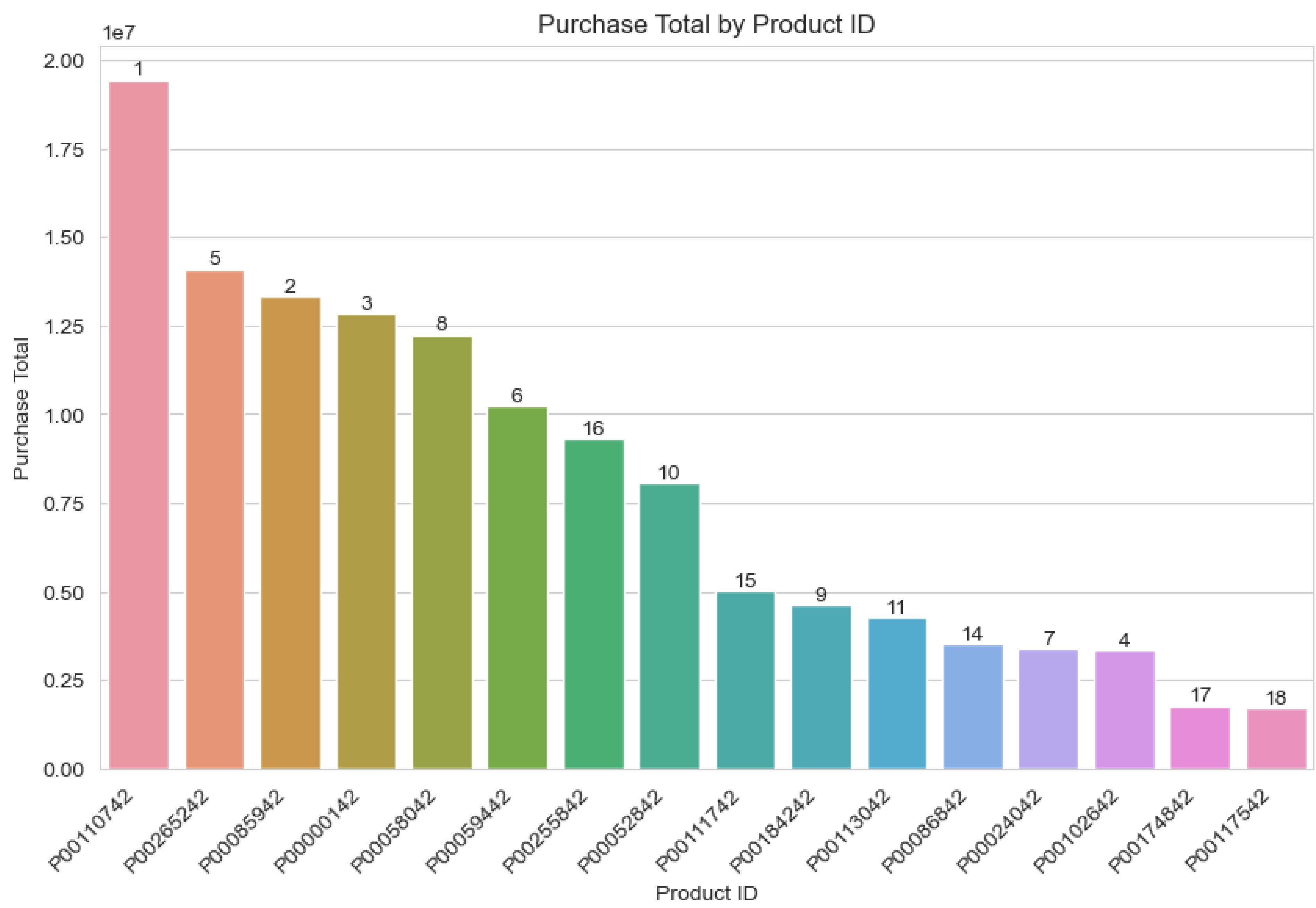
	Product_ID	Product_Category	purchase_total
0	P00110742	1	19441595.0
1	P00265242	5	14096689.0
2	P00085942	2	13325003.0
3	P00000142	3	12837476.0
4	P00058042	8	12248655.0
5	P00059442	6	10221399.0
6	P00255842	16	9315477.0
7	P00052842	10	8064470.0
8	P00111742	15	5016226.0
9	P00184242	9	4622073.0
10	P00113042	11	4267142.0
11	P00086842	14	3518562.0
12	P00024042	7	3393070.0
13	P00102642	4	3350394.0
14	P00174842	17	1775599.0
15	P00117542	18	1710647.0

In [ ]:

```
top_products["Product_ID"]= top_products["Product_ID"].astype(str)
```

In [ ]:

```
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Product_ID', y='purchase_total', data=top_products)

for i, row in top_products.iterrows():
    ax.text(i, row['purchase_total'], str(row['Product_Category']), ha='center', va='bottom')

plt.title("Purchase Total by Product ID")
plt.xlabel("Product ID")
plt.ylabel("Purchase Total")
plt.xticks(rotation=45, ha='right')
plt.show();
```



Insights

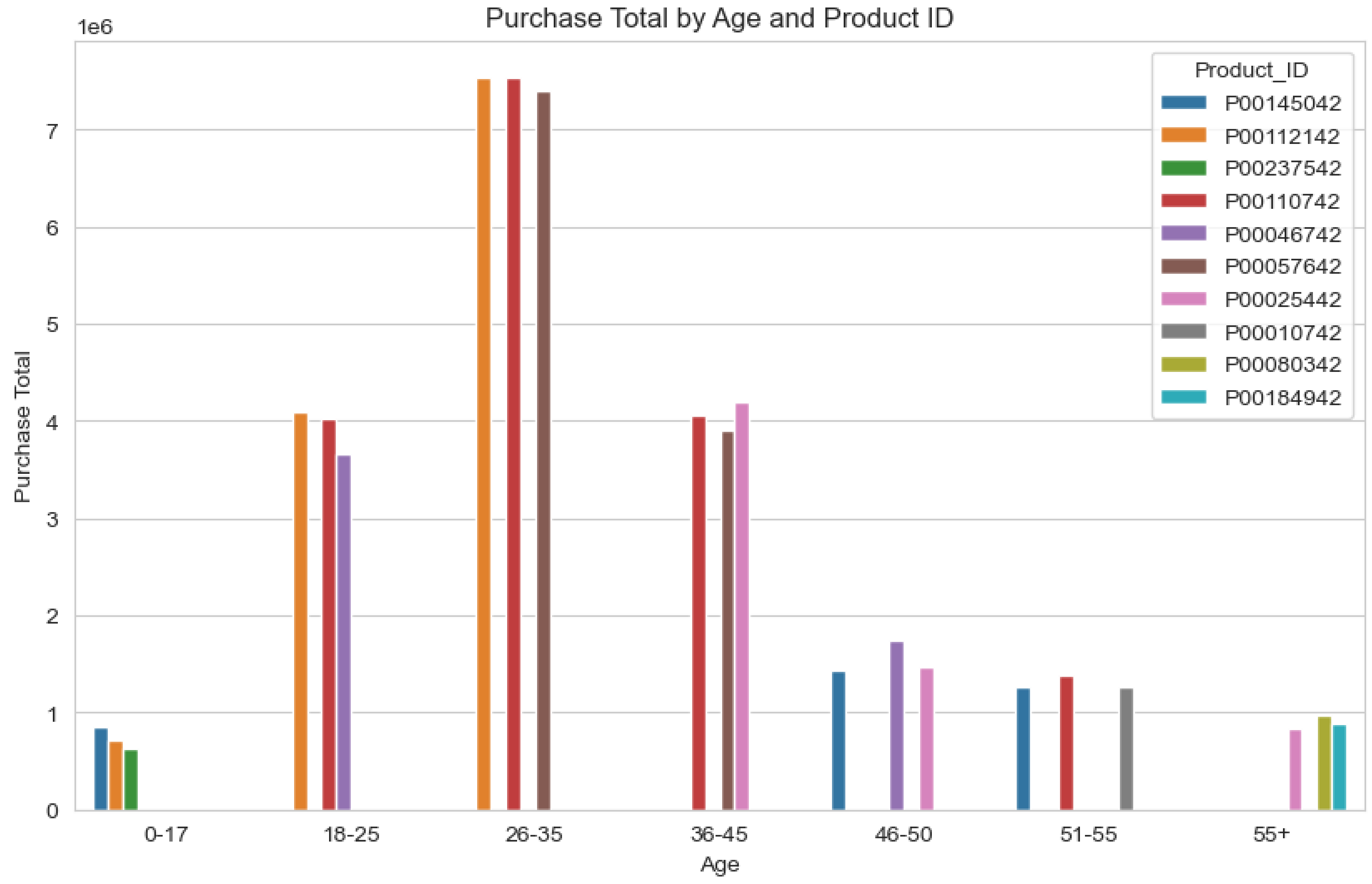
- The category name is printed on top of each bar
- Above plot shows top product for each category in terms of purchase value
- This data can be used to manage the inventory for the stores

In [ ]:

```
top_products_age=db.sql("""
SELECT Age, Product_ID, purchase_total, rnk_total
FROM (
    SELECT
        Age, Product_ID, SUM(Purchase) AS purchase_total,
```

```
        RANK() OVER (PARTITION BY Age ORDER BY purchase_total DESC) AS rnk_total
    FROM df
    GROUP BY Age, Product_ID
) AS ranked_data
WHERE rnk_total <= 3  order by Age, rnk_total
""").to_df()
top_products_age["Product_ID"]= top_products_age["Product_ID"].astype(str)
```

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Age', y='purchase_total', hue='Product_ID', data=top_products_age)
plt.xlabel('Age')
plt.ylabel('Purchase Total')
plt.title('Purchase Total by Age and Product ID');
```



Insights

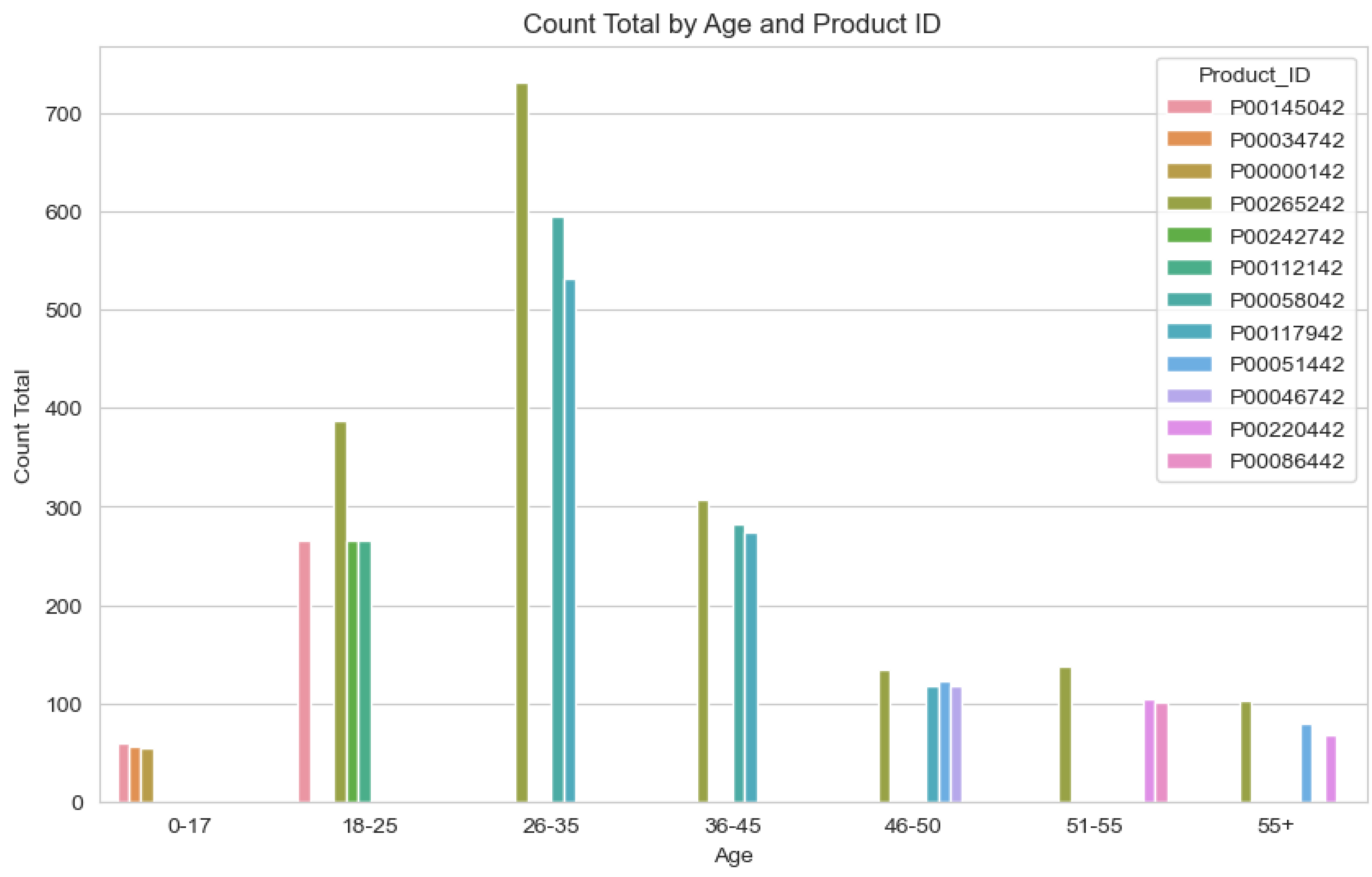
- Above plot shows top products in terms of sale for each age group
- This data can be used to manage the inventory for the stores

In [ ]:
```
count_wise_age_product_data = db.sql("""
SELECT Age, Product_ID, cnt, rnk_cnt
FROM (
    SELECT
        Age, Product_ID, COUNT(*) AS cnt,
        RANK() OVER (PARTITION BY Age ORDER BY cnt DESC) AS rnk_cnt,
    FROM df
    GROUP BY Age, Product_ID
) AS ranked_data
WHERE rnk_cnt <=3 order by Age, rnk_cnt;
""").to_df()
count_wise_age_product_data["Product_ID"]= count_wise_age_product_data["Product_ID"].astype(str)
count_wise_age_product_data
```

Out[ ]:

	Age	Product_ID	cnt	rnk_cnt
0	0-17	P00145042	59	1
1	0-17	P00034742	56	2
2	0-17	P00000142	55	3
3	18-25	P00265242	387	1
4	18-25	P00242742	265	2
5	18-25	P00112142	265	2
6	18-25	P00145042	265	2
7	26-35	P00265242	732	1
8	26-35	P00058042	595	2
9	26-35	P00117942	532	3
10	36-45	P00265242	308	1
11	36-45	P00058042	282	2
12	36-45	P00117942	274	3
13	46-50	P00265242	135	1
14	46-50	P00051442	122	2
15	46-50	P00117942	117	3
16	46-50	P00046742	117	3
17	51-55	P00265242	138	1
18	51-55	P00220442	104	2
19	51-55	P00086442	101	3
20	55+	P00265242	103	1
21	55+	P00051442	79	2
22	55+	P00220442	67	3

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Age', y='cnt', hue='Product_ID', data=count_wise_age_product_data)
plt.xlabel('Age')
plt.ylabel('Count Total')
plt.title('Count Total by Age and Product ID');
```



Insights

- Above plot shows top products in terms of popularity for each age group

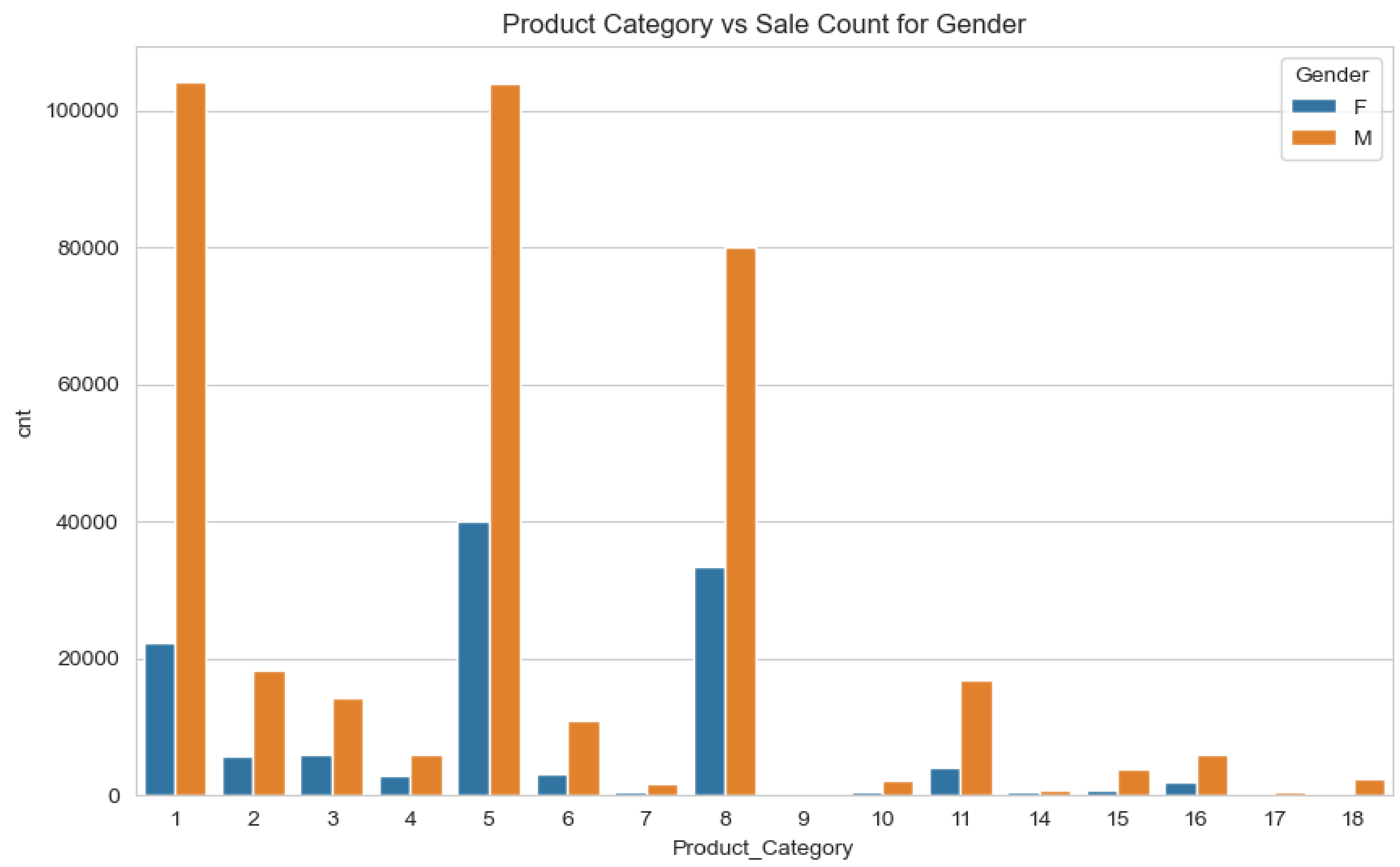
In [ ]:
```
db.sql("""
SELECT Stay_In_Current_City_Years, sum(Purchase) total_purchase
from df group by Stay_In_Current_City_Years order by total_purchase desc;
""").to_df()
```

Out[ ]:

	Stay_In_Current_City_Years	total_purchase
0	1	1.583421e+09
1	2	8.394286e+08
2	3	7.839477e+08
3	4	6.933037e+08
4	0	6.049331e+08

In [ ]:
```
gender_wise_sale = db.sql("""
SELECT Gender, Product_Category, count(*) cnt from df group by Product_Category, Gender order by cnt desc;
""").to_df()
```

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Product_Category', y='cnt', hue='Gender', data=gender_wise_sale)
plt.title('Product Category vs Sale Count for Gender');
```



Insights

- Above plot shows the difference in count of sales between male and female users for different categories
- There is a big difference in count of sales between male and female users for all categories.

In [ ]:
```
db.sql("""
SELECT Product_Category, COUNT(DISTINCT Product_ID) AS cnt
FROM df
GROUP BY Product_Category
ORDER BY cnt DESC;
""").to_df()
```

Out[ ]:

	Product_Category	cnt
0	8	1046
1	5	956
2	1	493
3	11	248
4	2	152
5	6	118
6	7	101
7	16	98
8	3	90
9	4	88
10	15	44
11	14	44
12	18	30
13	10	25
14	17	11
15	9	2

In [ ]:

```
female_products = db.sql("""
SELECT Product_Category, COUNT(DISTINCT Product_ID) AS cnt
FROM df
WHERE Gender = 'F' AND Product_ID NOT IN (SELECT DISTINCT Product_ID FROM df WHERE Gender = 'M')
GROUP BY Product_Category
ORDER BY cnt DESC;

""").to_df()
female_products["Product_Category"]= female_products["Product_Category"].astype(str)
```

In [ ]:

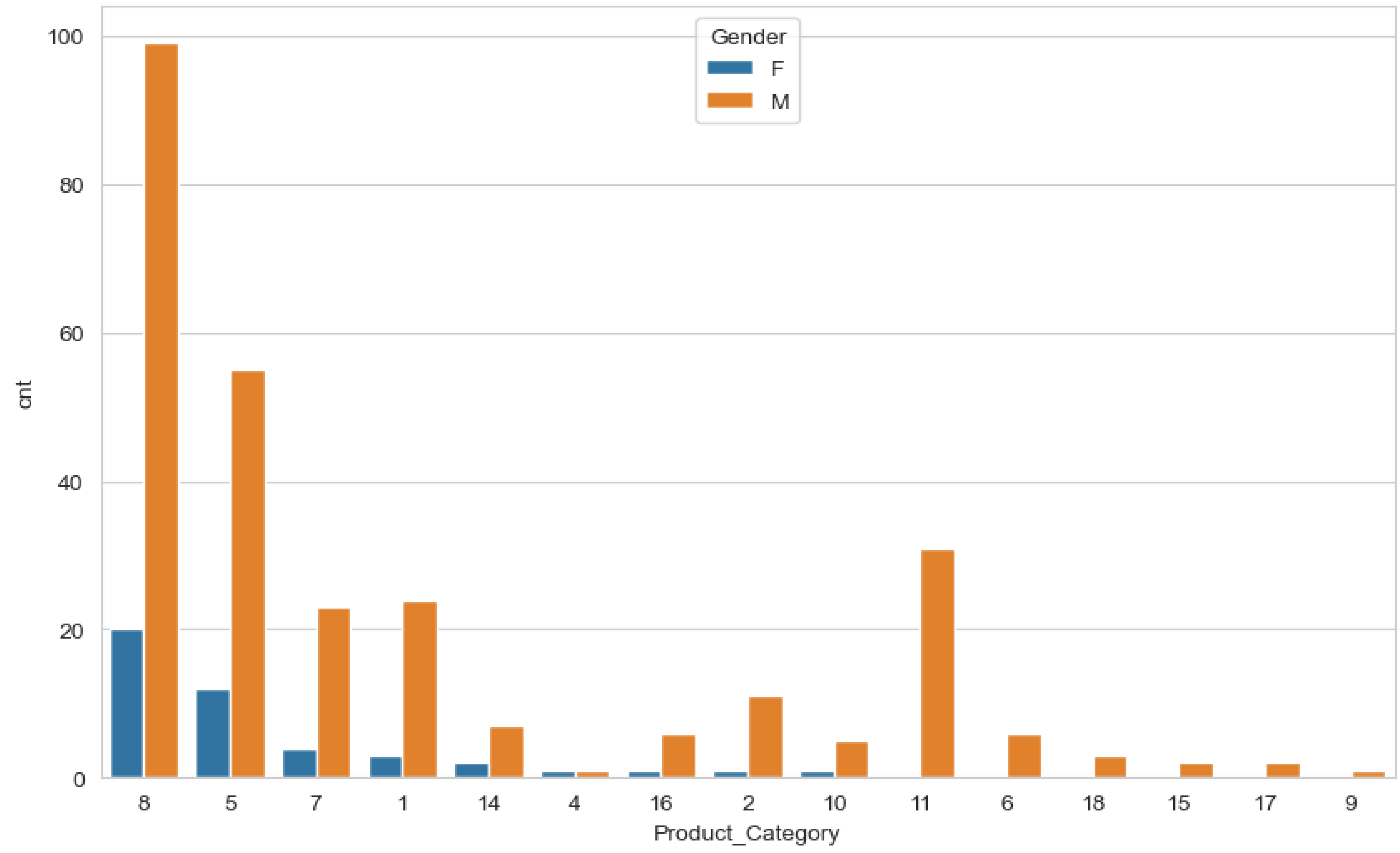
```
male_products = db.sql("""
SELECT Product_Category, COUNT(DISTINCT Product_ID) AS cnt
FROM df
WHERE Gender = 'M' AND Product_ID NOT IN (SELECT DISTINCT Product_ID FROM df WHERE Gender = 'F')
GROUP BY Product_Category
ORDER BY cnt DESC;

""").to_df()
male_products["Product_Category"] = male_products["Product_Category"].astype(str)
```

In [ ]:

```
female_products["Gender"]="F"
male_products["Gender"]="M"
result = pd.concat([female_products, male_products], axis=0, ignore_index=True)
# result
```

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Product_Category', y='cnt', hue="Gender" ,data=result);
```



Insights

- Above plot show the count of unique products that is used by only 1 gender in a given category.
- We can see that there are lot less female only products than male products.
- This might be one of the reasons why there are more male users

In [ ]:

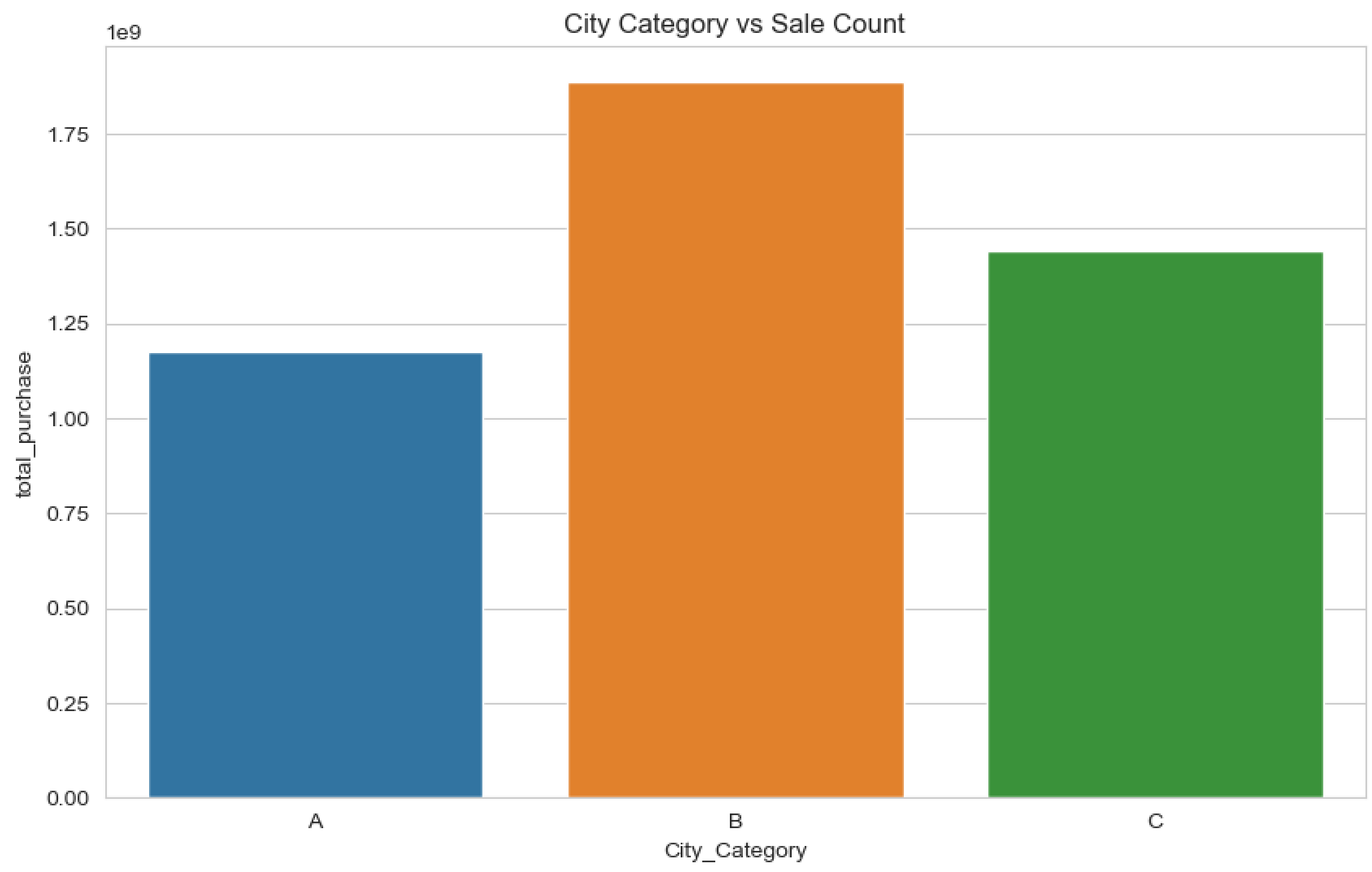
```
city_sales = db.sql("""
SELECT City_Category, sum(Purchase) total_purchase, count(*) from df group by City_Category order by total_purchase desc;
""").to_df()
city_sales
```

Out[ ]:

	City_Category	total_purchase	count_star()
0	B	1.886158e+09	209348
1	C	1.443221e+09	152037
2	A	1.175656e+09	133648

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='City_Category', y='total_purchase', data=city_sales)
plt.title('City Category vs Sale Count');
```



Insights

- Above plot shows the difference in sales between different cities
- Even though City C has more users, City B had higher sales

Hypothesis Testing

In [ ]:
```
critical_value = norm.ppf(0.95 + (1-0.95)/2)

confidence_intervals= ["90","95","99"]
sample_sizes=[300,3000,30000]
def calc_sample_mean(data, no_of_sample_means,number_of_samples):
    mean = []
    for i in range(no_of_sample_means):
        sample = data.sample(n=number_of_samples, replace=True)
        sample_mean=sample.mean()
        mean.append(sample_mean)
    return mean

def generate_confidence_intervals(data, standard_deviation_population):
    # standard_error_sample=standard_deviation_population/np.sqrt(5000)
    # margin_of_error = critical_value * standard_error_sample
    # data_mean = np.mean(data)
    # confidence_interval = [data_mean-margin_of_error, data_mean+margin_of_error]
    confidence_interval = {
        "90": np.percentile(data, [5, 95]),
        "95": np.percentile(data, [2.5, 97.5]),
        "99":np.percentile(data, [0.5, 99.5])
        }
    return confidence_interval

def generate_mini_report(all_details):
    data=all_details["data"]
    print("================== Full Data ==================")
    mean_population = data.mean(axis=0)
    standard_deviation_population = data.std(axis=0)
    confidence_interval_population = generate_confidence_intervals(data,standard_deviation_population)
    print(f"Poulation Mean: {mean_population},\nPoulation Standard Deviation: {standard_deviation_population}")
    print(f"=> Confidence Interval for full data")
    for ci in confidence_intervals:
        print(f"Confidence Interval for {ci}: {confidence_interval_population[ci]}")
    print("\n================== Sample Means ==================")
    mean_sample_1=calc_sample_mean(data, no_of_sample_means=5000, number_of_samples=300)
    mean_sample_2=calc_sample_mean(data, no_of_sample_means=5000, number_of_samples=3000)
    mean_sample_3=calc_sample_mean(data, no_of_sample_means=5000, number_of_samples=30000)
    print(f"Sample Mean for 300: {np.mean(mean_sample_1)},\nSample Mean for 3000: {np.mean(mean_sample_2)},
      \nSample Mean for 30000: {np.mean(mean_sample_3)}")
    print("\n================== Confidence Intervals ==================")

    confidence_interval_1 =  generate_confidence_intervals(mean_sample_1,standard_deviation_population)
    confidence_interval_2 =  generate_confidence_intervals(mean_sample_2,standard_deviation_population)
    confidence_interval_3 =  generate_confidence_intervals(mean_sample_3,standard_deviation_population)

    for ss in sample_sizes:
        print(f"\n=> Confidence Interval for {ss}")
        for ci in confidence_intervals:
            print(f"Confidence Interval for {ci}: {confidence_interval_1[ci]}")

    output={
        "standard_deviation_population":standard_deviation_population,
        "sample_300":{
            "mean_sample":mean_sample_1,
            "confidence_interval":confidence_interval_1
        },
        "sample_3000":{
            "mean_sample":mean_sample_2,
            "confidence_interval":confidence_interval_2
        },
        "sample_30000":{
            "mean_sample":mean_sample_3,
            "confidence_interval":confidence_interval_3
        },
        "sample_all":{
            "mean_sample":data,
            "confidence_interval":confidence_interval_population
        }
    }
    return output


def plot_purchase_distribution(data_report, labels, ci):
    sample_sizes = ["sample_300", "sample_3000", "sample_30000"]

    fig, axes = plt.subplots(1, 3, figsize=(20, 5), sharey=True)

    for i, sample_size in enumerate(sample_sizes):
        data1 = data_report[0][sample_size]["mean_sample"]
        data2 = data_report[1][sample_size]["mean_sample"]

        sns.kdeplot(data1, ax=axes[i], label=labels[0], color='red')
        sns.kdeplot(data2, ax=axes[i], label=labels[1], color='teal')

        ci1 = data_report[0][sample_size]["confidence_interval"][ci]
        ci2 = data_report[1][sample_size]["confidence_interval"][ci]
```

```
        axes[i].axvline(x=ci1[0], color='tomato', linestyle='--', label=f'{labels[0]} Confidence Interval')
        axes[i].axvline(x=ci1[1], color='tomato', linestyle='--')
        axes[i].axvline(x=ci2[0], color='green', linestyle='--', label=f'{labels[1]} Confidence Interval')
        axes[i].axvline(x=ci2[1], color='green', linestyle='--')

        axes[i].set_xlabel(f'Sample Base Purchase (Sample size = {sample_size.split("_")[1]})')

    plt.legend(bbox_to_anchor=(-0.8, -0.2), loc='upper center', ncol=2)

    axes[0].set_ylabel('Probability Density')
    fig.suptitle(f'Purchase distribution for {labels[0]} and {labels[1]} with Confidence Intervals of {ci}%', fontsize=20)

    plt.show()
```

In [ ]:
```
reduced_data_gender = db.sql("""
    select User_ID, Gender, sum(Purchase) Purchase from df group by User_ID, Gender
""").to_df()
reduced_data_maritial_status = db.sql("""
    select User_ID, Marital_Status, sum(Purchase) Purchase from df group by Marital_Status, User_ID
""").to_df()
reduced_data_age = db.sql("""
    select User_ID, Age, sum(Purchase) Purchase from df group by Age, User_ID
""").to_df()
```

In [ ]:
```
male_purchase_data=reduced_data_gender.loc[reduced_data_gender['Gender']=='M']["Purchase"]
female_purchase_data=reduced_data_gender.loc[reduced_data_gender['Gender']=='F']["Purchase"]
married_purchase_data=reduced_data_maritial_status.loc[reduced_data_maritial_status['Marital_Status']==1]["Purchase"]
single_purchase_data=reduced_data_maritial_status.loc[reduced_data_maritial_status['Marital_Status']==0]["Purchase"]
```

Effect of Gender on Purchases

Define hypothesis

- Ho = Women spend less than or equal to men
- Ha = Women spend more than men

In [ ]:
```
male_purchase_data.describe()
female_purchase_data.describe()
```

Out[ ]:
```
count    4.225000e+03
mean     8.141565e+05
std      8.899806e+05
min      3.502400e+04
25%      2.200000e+05
50%      4.934140e+05
75%      1.068209e+06
max      9.008191e+06
Name: Purchase, dtype: float64
```

Out[ ]:
```
count    1.666000e+03
mean     6.393897e+05
std      7.418809e+05
min      4.394300e+04
25%      1.756582e+05
50%      3.610935e+05
75%      7.862822e+05
max      5.381441e+06
Name: Purchase, dtype: float64
```

Observations

- Above data shows the population means of purchase data for male and female customers

Male data analysis

In [ ]:
```
male_purchase_data_report = generate_mini_report({"data": male_purchase_data})
```

```
================== Full Data ==================
Poulation Mean: 814156.4904142012,
Poulation Standard Deviation: 889980.6238453629
=> Confidence Interval for full data
Confidence Interval for 90: [ 105041.  2644843.2]
Confidence Interval for 95: [  89860.  3395068.2]
Confidence Interval for 99: [  61671.08 4817056.36]

================== Sample Means ==================
Sample Mean for 300: 812791.0653293333,
Sample Mean for 3000: 814318.7961082667,
Sample Mean for 30000: 814261.2487411

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [730285.67383333 900579.70816667]
Confidence Interval for 95: [716156.01575    920001.72183333]
Confidence Interval for 99: [688048.55615    953068.59228333]

=> Confidence Interval for 3000
Confidence Interval for 90: [730285.67383333 900579.70816667]
Confidence Interval for 95: [716156.01575    920001.72183333]
Confidence Interval for 99: [688048.55615    953068.59228333]

=> Confidence Interval for 30000
Confidence Interval for 90: [730285.67383333 900579.70816667]
Confidence Interval for 95: [716156.01575    920001.72183333]
Confidence Interval for 99: [688048.55615    953068.59228333]
```

Observations

- Above data shows the population means and confidence intervals of males for different sample sizes

Female data analysis

In [ ]:
```
female_purchase_data_report = generate_mini_report({"data": female_purchase_data})
```

```
================== Full Data ==================
Poulation Mean: 639389.6548619447,
Poulation Standard Deviation: 741880.916768428
=> Confidence Interval for full data
Confidence Interval for 90: [  92706.75 2212421.25]
Confidence Interval for 95: [  78025.5   2870749.625]
Confidence Interval for 99: [  56585.35 3958352.   ]

================== Sample Means ==================
Sample Mean for 300: 639511.9935346666,
Sample Mean for 3000: 639268.1466572,
Sample Mean for 30000: 639413.1769396601

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [570546.92366667 713737.56083333]
Confidence Interval for 95: [556339.908  728539.2035]
Confidence Interval for 99: [536836.75816667 756421.21296667]

=> Confidence Interval for 3000
Confidence Interval for 90: [570546.92366667 713737.56083333]
Confidence Interval for 95: [556339.908  728539.2035]
Confidence Interval for 99: [536836.75816667 756421.21296667]

=> Confidence Interval for 30000
Confidence Interval for 90: [570546.92366667 713737.56083333]
Confidence Interval for 95: [556339.908  728539.2035]
Confidence Interval for 99: [536836.75816667 756421.21296667]
```

Observations

- Above data shows the population means and confidence intervals of females for different sample sizes

Confidence Interval Analysis

In [ ]:
```
m_all= male_purchase_data_report["sample_all"]["confidence_interval"]["95"][1] - male_purchase_data_report["sample_all"]["confidence_interval"]["95"][0]
f_all= female_purchase_data_report["sample_all"]["confidence_interval"]["95"][1] - female_purchase_data_report["sample_all"]["confidence_interval"]["95"][0]
print(f"Confidence_interval width for men : {m_all}\nConfidence_interval width for females : {f_all}")
```

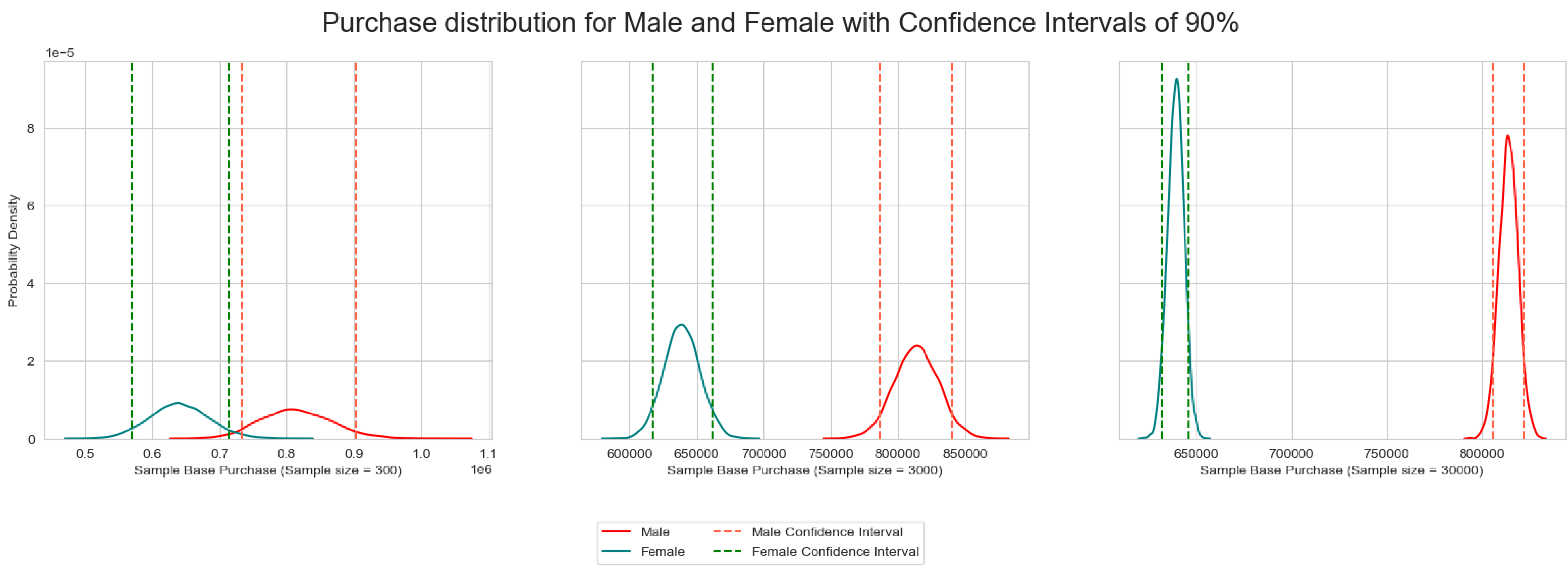
```
Confidence_interval width for men : 3305208.199999999
Confidence_interval width for females : 2792724.125
```

Insights

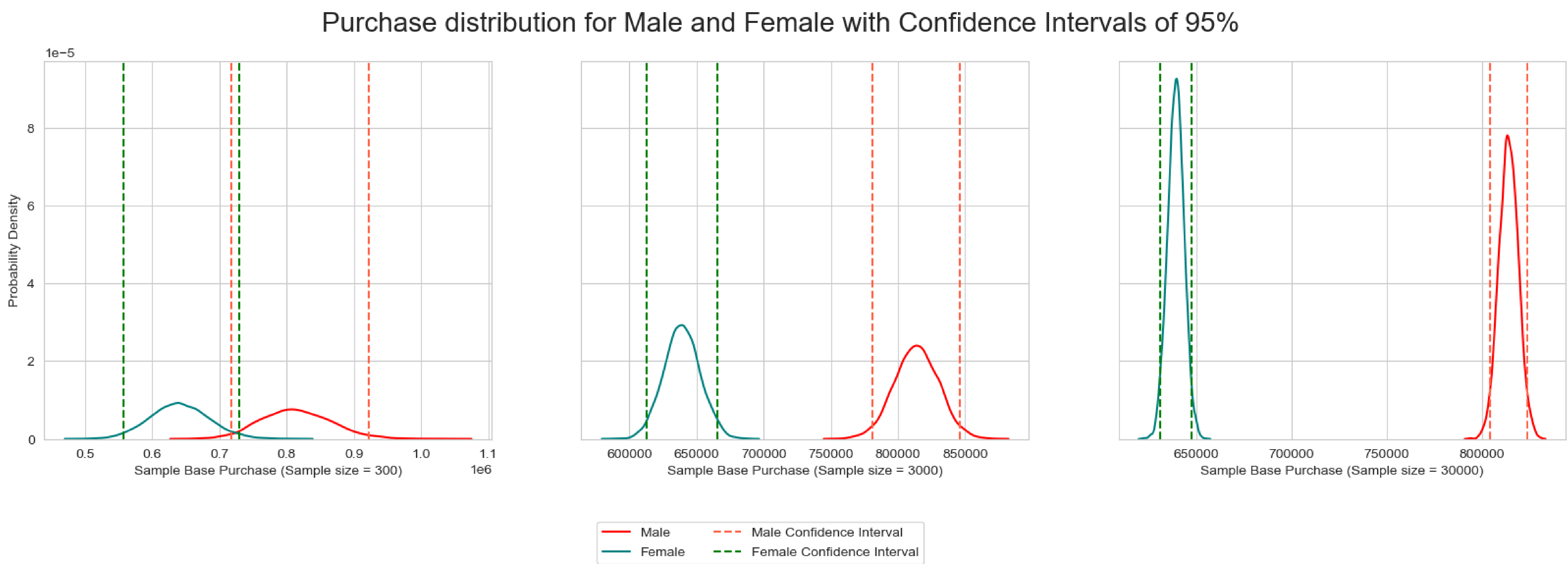
- Above data shows that there is some difference in confidence interval widths between male and female customers for same sample count.
- The confidence interval for male is wider than the confidence interval for female. This is because there is greater variance in purchase data for men

Plotting

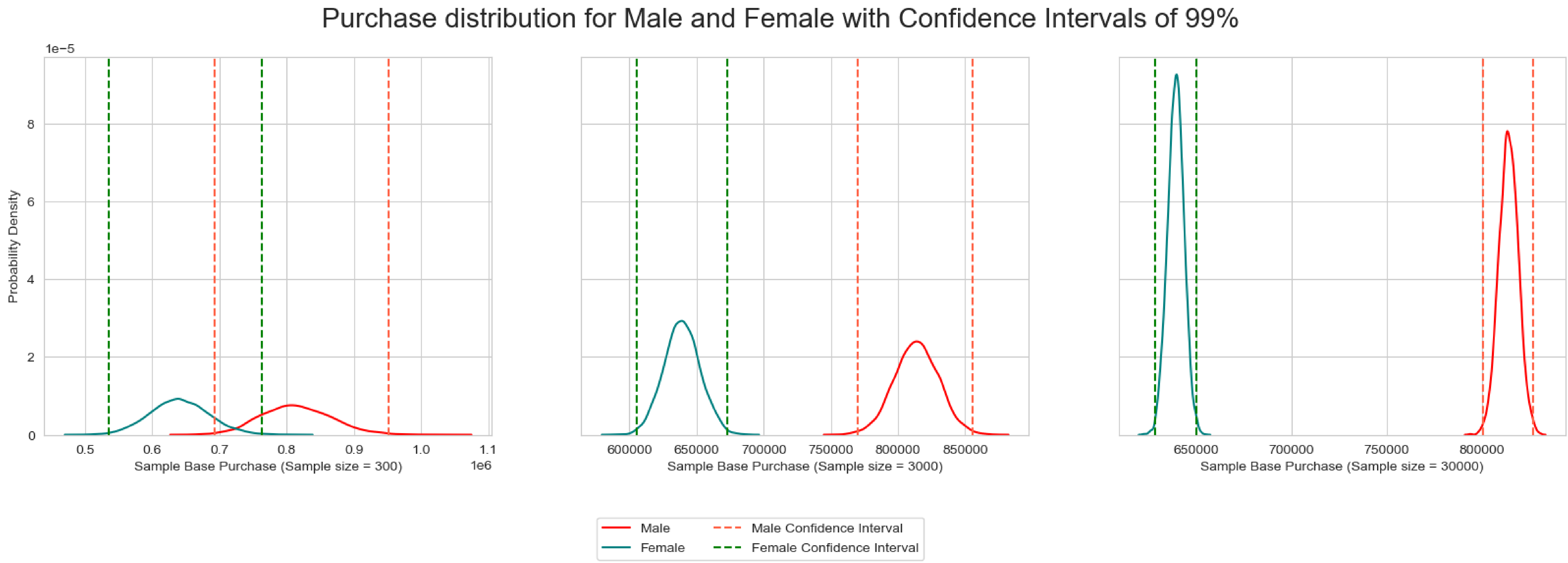
In [ ]:
```
plot_purchase_distribution([male_purchase_data_report, female_purchase_data_report], ['Male', 'Female'], '90')
```



In [ ]:
```
plot_purchase_distribution([male_purchase_data_report, female_purchase_data_report], ['Male', 'Female'], '95')
```



In [ ]:
```
plot_purchase_distribution([male_purchase_data_report, female_purchase_data_report], ['Male', 'Female'], '99')
```



Insights

Above plots show distribution for different sample sizes

- We can see that the purchase distribution of women is less than men.
- We can also see that confidence interval decreases in width as the number of the samples increases.

2 sample Z Test

In [ ]:
```
male_data=male_purchase_data_report["sample_30000"]["mean_sample"]
female_data = female_purchase_data_report["sample_30000"]["mean_sample"]
mean_men = np.mean(male_data)
mean_women = np.mean(female_data)
# se_men = np.std(male_data, ddof=1)/ np.sqrt(len(male_data))
# se_women = np.std(female_data, ddof=1) / np.sqrt(len(female_data))
pooled_SE = np.sqrt(np.std(male_data, ddof=1)**2/len(male_data) +  np.std(female_data, ddof=1)**2/len(female_data))
z_score = (mean_women - mean_men) / pooled_SE
p_value =1-norm.cdf(z_score)
z_score, p_value
```

Out[ ]:
```
(-1866.4479177949604, 1.0)
```

In [ ]:
```
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Fail to reject the null hypothesis
```

T Test

In [ ]:
```
t_stat,pval = ttest_ind(female_purchase_data_report["sample_30000"]["mean_sample"],
                        male_purchase_data_report["sample_30000"]["mean_sample"], alternative="greater")
t_stat,pval
```

Out[ ]:
```
(-1866.4479177949606, 1.0)
```

In [ ]:
```
alpha = 0.05
if pval < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Fail to reject the null hypothesis
```

Insights

- Using both 2 Sample Z test and T test, we failed to reject the null hypothesis with a **p value of 1**. This means that women are spending less than or equal to men.
- Also from the distribution graphs, we can conclude that women are spending less than to men.
- There is no overlap of distribution of purchase data for men and females.

Effect of Maritial Status on Purchases

Define hypothesis

- Ho = No difference in spending amounts between married and single
- Ha = There is a difference in spending amounts between married and single

Married user data analysis

In [ ]:
```
married_purchase_data_report = generate_mini_report({"data": married_purchase_data})
```

```
================== Full Data ==================
Poulation Mean: 746053.6026677445,
Poulation Standard Deviation: 850263.2921095208
=> Confidence Interval for full data
Confidence Interval for 90: [ 100160.5 2518728.2]
Confidence Interval for 95: [  84469.625 3295891.875]
Confidence Interval for 99: [  60640.815 4774602.69 ]

================== Sample Means ==================
Sample Mean for 300: 744804.1585520001,
Sample Mean for 3000: 745899.7278186666,
Sample Mean for 30000: 746042.2245121533

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [668466.74766667 829926.76883333]
Confidence Interval for 95: [654348.98633333 849063.23308333]
Confidence Interval for 99: [622500.53896667 883041.96876667]

=> Confidence Interval for 3000
Confidence Interval for 90: [668466.74766667 829926.76883333]
Confidence Interval for 95: [654348.98633333 849063.23308333]
Confidence Interval for 99: [622500.53896667 883041.96876667]

=> Confidence Interval for 30000
Confidence Interval for 90: [668466.74766667 829926.76883333]
Confidence Interval for 95: [654348.98633333 849063.23308333]
Confidence Interval for 99: [622500.53896667 883041.96876667]
```

Observations

- Above data shows the population means and confidence intervals of married users for different sample sizes

Single user data analysis

```
In [ ]: single_purchase_data_report = generate_mini_report({"data": single_purchase_data})
```

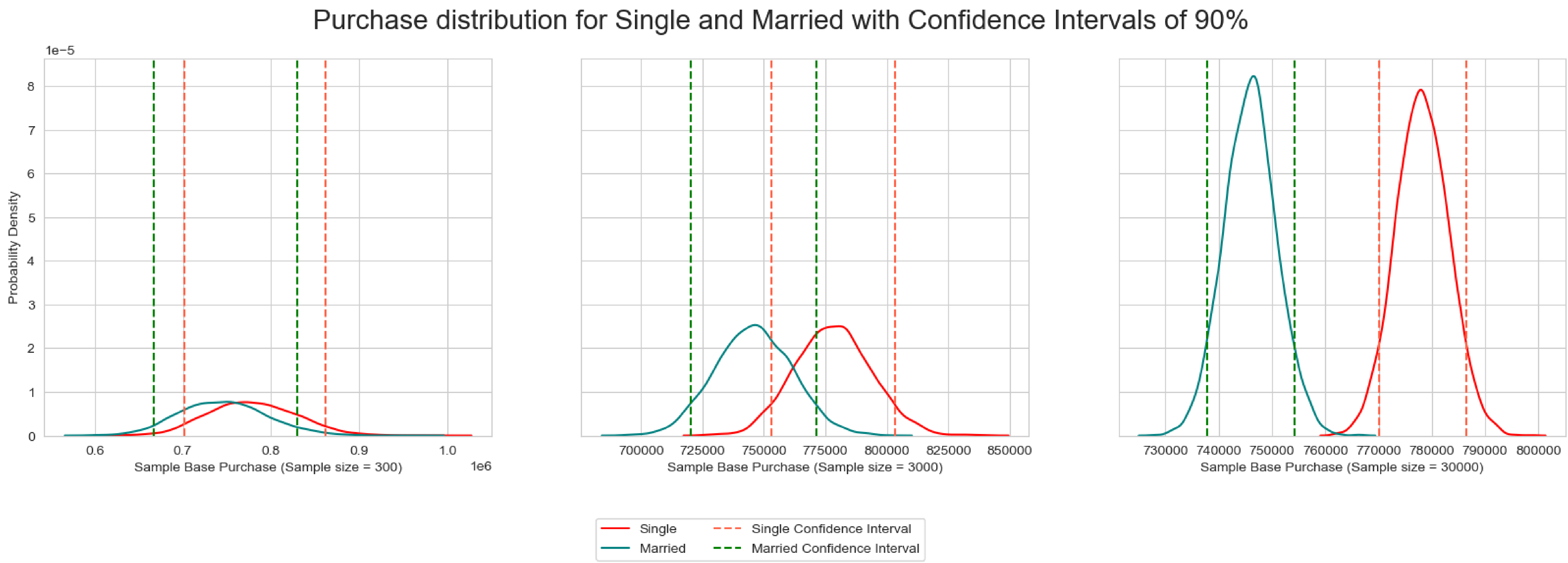
```
================== Full Data ==================
Poulation Mean: 778255.113842552,
Poulation Standard Deviation: 857061.306214359
=> Confidence Interval for full data
Confidence Interval for 90: [ 100754.4 2538139.2]
Confidence Interval for 95: [  85137.2 3208610.8]
Confidence Interval for 99: [  59745.92       4508696.64000001]

================== Sample Means ==================
Sample Mean for 300: 778928.039672,
Sample Mean for 3000: 778365.9602614,
Sample Mean for 30000: 778319.8659684401

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [702885.99716667 861555.925     ]
Confidence Interval for 95: [689218.10691667 878790.87083333]
Confidence Interval for 99: [665069.37485    912167.94438333]

=> Confidence Interval for 3000
Confidence Interval for 90: [702885.99716667 861555.925     ]
Confidence Interval for 95: [689218.10691667 878790.87083333]
Confidence Interval for 99: [665069.37485    912167.94438333]

=> Confidence Interval for 30000
Confidence Interval for 90: [702885.99716667 861555.925     ]
Confidence Interval for 95: [689218.10691667 878790.87083333]
Confidence Interval for 99: [665069.37485    912167.94438333]
```

Observations

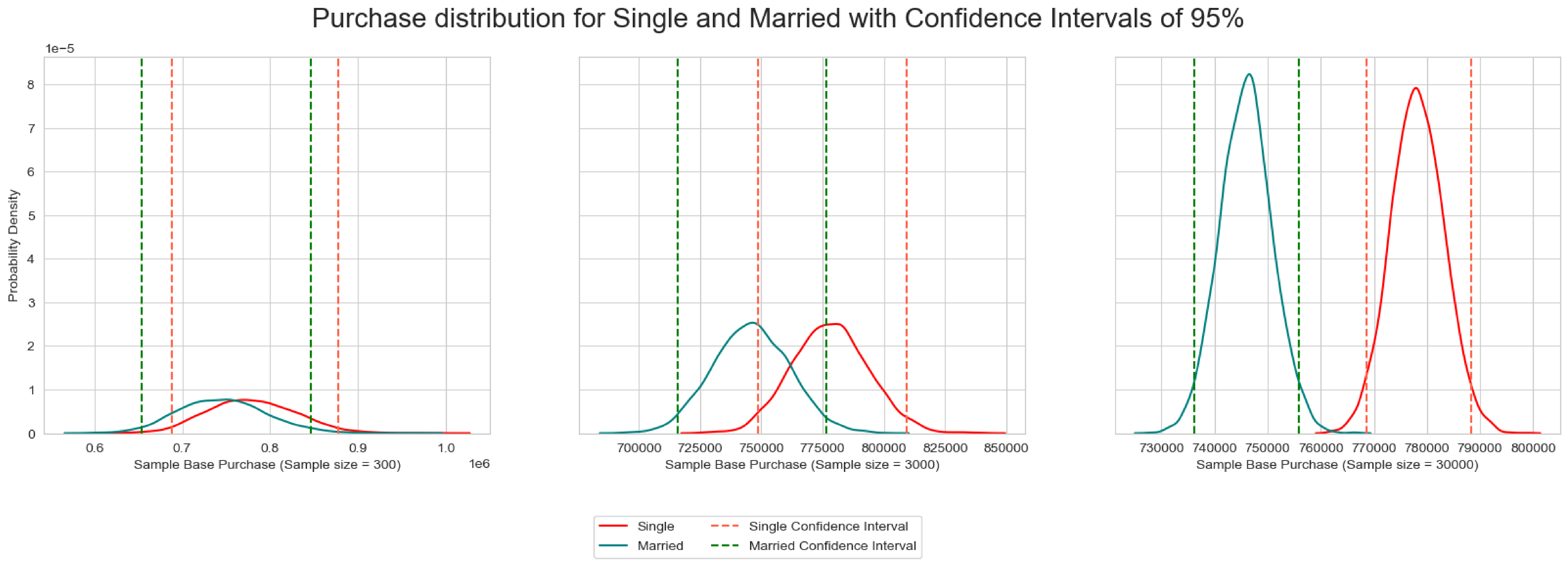
- Above data shows the population means and confidence intervals of single users for different sample sizes

Plotting

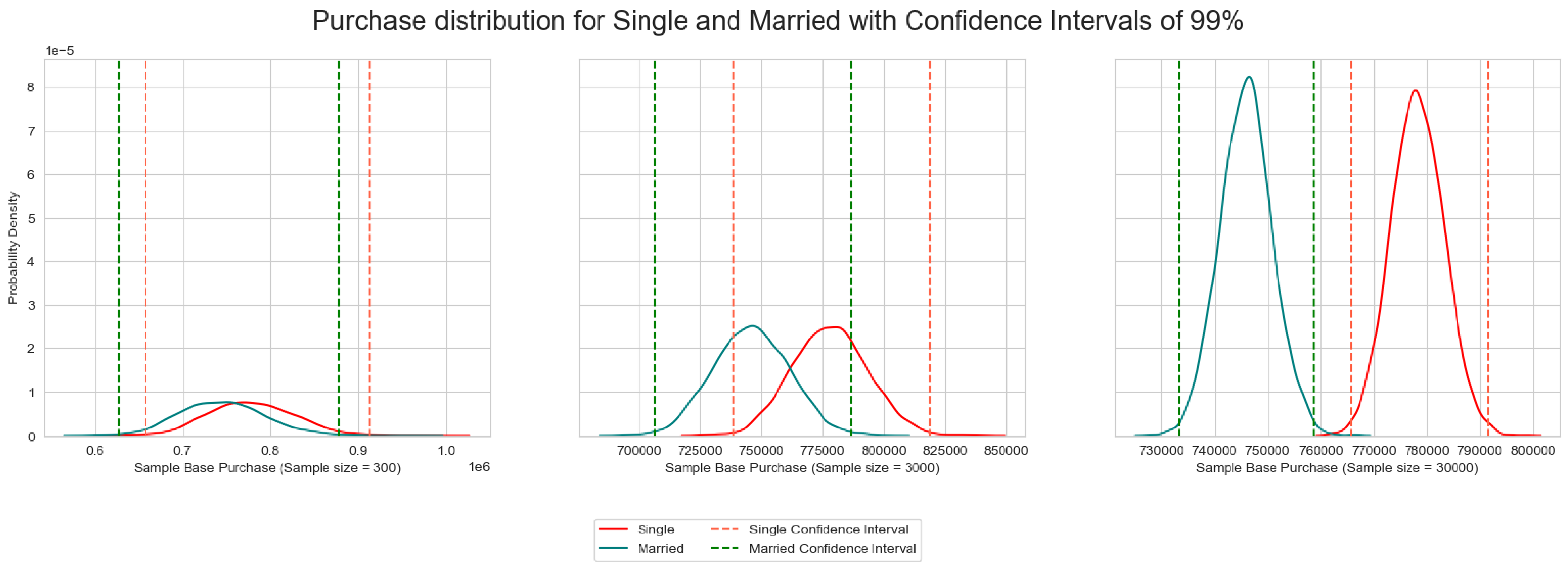
```
In [ ]: plot_purchase_distribution([single_purchase_data_report, married_purchase_data_report], ['Single', 'Married'], '90')
```




```
In [ ]: plot_purchase_distribution([single_purchase_data_report, married_purchase_data_report], ['Single', 'Married'], '95')
```




```
In [ ]: plot_purchase_distribution([single_purchase_data_report, married_purchase_data_report], ['Single', 'Married'], '99')
```



Insights

- Above plots show the distribution of purchase data for married and unmarried users
- Above plot show that there is minor difference between married and unmarried users purchase data
- The width of confidence interval decreases as the number of the samples increases.

T Test

```
In [ ]: t_stat,pval = ttest_ind(single_purchase_data_report["sample_30000"]["mean_sample"],
                                married_purchase_data_report["sample_30000"]["mean_sample"])
        t_stat,pval
```

```
Out[ ]: (322.2793781230446, 0.0)
```

```
In [ ]: alpha = 0.05
        if pval < alpha:
            print("Reject the null hypothesis")
        else:
            print("Fail to reject the null hypothesis")
```

```
Reject the null hypothesis
```

Insights

- Using T test, the p value is **0**. This means that we reject the null hypothesis.It means that there is difference between married and unmarried users purchase data.
- Using both plotting and T test, we found out there is some difference between married and unmarried users purchase data.

Effect of Age on Purchase

Define hypothesis

- Ho = No difference in spending amounts between age groups
- Ha = There is a difference in spending amounts between age groups

```
In [ ]: age_cats=df["Age"].unique().to_list()
        age_cats
```

```
Out[ ]: ['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
```

Age Group Analysis

```
In [ ]: reduced_data_age[reduced_data_age['Age'] == "55+"]["Purchase"].mean()
```

```
Out[ ]: 473105.95967741933
```

```
In [ ]: age_data_dict={}
        for cat in age_cats:
            data = reduced_data_age[reduced_data_age['Age'] == cat]["Purchase"]
            print(f"\n\nAge Category: {cat} ")
            age_data_dict[cat] = generate_mini_report({"data": data})
```

```
Age Category: 0-17
================== Full Data ==================
Poulation Mean: 546507.1651376147,
Poulation Standard Deviation: 620184.3525167555
=> Confidence Interval for full data
Confidence Interval for 90: [  90353.05 1501713.65]
Confidence Interval for 95: [  83321.225 1868191.7  ]
Confidence Interval for 99: [  69622.83 3353020.31]

================== Sample Means ==================
Sample Mean for 300: 546423.0730593334,
Sample Mean for 3000: 546606.2223684667,
Sample Mean for 30000: 546488.0495021

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [489450.00966667 605989.61633333]
Confidence Interval for 95: [480437.21366667 619366.82241667]
Confidence Interval for 99: [459872.2587 644810.3554]

=> Confidence Interval for 3000
Confidence Interval for 90: [489450.00966667 605989.61633333]
Confidence Interval for 95: [480437.21366667 619366.82241667]
Confidence Interval for 99: [459872.2587 644810.3554]

=> Confidence Interval for 30000
Confidence Interval for 90: [489450.00966667 605989.61633333]
Confidence Interval for 95: [480437.21366667 619366.82241667]
Confidence Interval for 99: [459872.2587 644810.3554]


Age Category: 55+
================== Full Data ==================
Poulation Mean: 473105.95967741933,
Poulation Standard Deviation: 555349.855178293
=> Confidence Interval for full data
Confidence Interval for 90: [  89741.05 1373461.05]
Confidence Interval for 95: [  73622.85 1768079.5 ]
Confidence Interval for 99: [  58885.245 3208304.63 ]

================== Sample Means ==================
Sample Mean for 300: 472925.4390173333,
Sample Mean for 3000: 473245.49545706663,
Sample Mean for 30000: 473033.10276254005

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [423688.49533333 525981.16316667]
Confidence Interval for 95: [415156.31866667 538848.58708333]
Confidence Interval for 99: [398513.43091667 566035.02556667]

=> Confidence Interval for 3000
Confidence Interval for 90: [423688.49533333 525981.16316667]
Confidence Interval for 95: [415156.31866667 538848.58708333]
Confidence Interval for 99: [398513.43091667 566035.02556667]

=> Confidence Interval for 30000
Confidence Interval for 90: [423688.49533333 525981.16316667]
Confidence Interval for 95: [415156.31866667 538848.58708333]
Confidence Interval for 99: [398513.43091667 566035.02556667]


Age Category: 26-35
================== Full Data ==================
Poulation Mean: 875995.6244520214,
Poulation Standard Deviation: 932693.7888342171
=> Confidence Interval for full data
Confidence Interval for 90: [ 108394.6        2791297.19999999]
Confidence Interval for 95: [  92493.7 3565330.9]
Confidence Interval for 99: [  59844.74 4931444.58]

================== Sample Means ==================
Sample Mean for 300: 876623.4806813333,
Sample Mean for 3000: 875761.5485639333,
Sample Mean for 30000: 875923.6763440201

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [787262.754      967171.50833333]
Confidence Interval for 95: [770897.23258333 985055.33591667]
Confidence Interval for 99: [ 737215.47581667 1019749.17485   ]

=> Confidence Interval for 3000
Confidence Interval for 90: [787262.754      967171.50833333]
Confidence Interval for 95: [770897.23258333 985055.33591667]
Confidence Interval for 99: [ 737215.47581667 1019749.17485   ]

=> Confidence Interval for 30000
Confidence Interval for 90: [787262.754      967171.50833333]
Confidence Interval for 95: [770897.23258333 985055.33591667]
Confidence Interval for 99: [ 737215.47581667 1019749.17485   ]


Age Category: 46-50
================== Full Data ==================
Poulation Mean: 703653.2071563088,
Poulation Standard Deviation: 844447.408543723
=> Confidence Interval for full data
Confidence Interval for 90: [ 102597.5 2469294. ]
Confidence Interval for 95: [  84823.   3520474.25]
Confidence Interval for 99: [  65669.85 4847140.25]

================== Sample Means ==================
Sample Mean for 300: 703702.2739453333,
Sample Mean for 3000: 703684.1184653334,
Sample Mean for 30000: 703781.863585

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [626100.71133333 785148.86416667]
Confidence Interval for 95: [613394.36516667 802455.13241667]
Confidence Interval for 99: [585429.24636667 837218.86756667]

=> Confidence Interval for 3000
Confidence Interval for 90: [626100.71133333 785148.86416667]
Confidence Interval for 95: [613394.36516667 802455.13241667]
Confidence Interval for 99: [585429.24636667 837218.86756667]

=> Confidence Interval for 30000
Confidence Interval for 90: [626100.71133333 785148.86416667]
Confidence Interval for 95: [613394.36516667 802455.13241667]
Confidence Interval for 99: [585429.24636667 837218.86756667]


Age Category: 51-55
================== Full Data ==================
Poulation Mean: 664805.6507276507,
Poulation Standard Deviation: 707454.0554398618
```

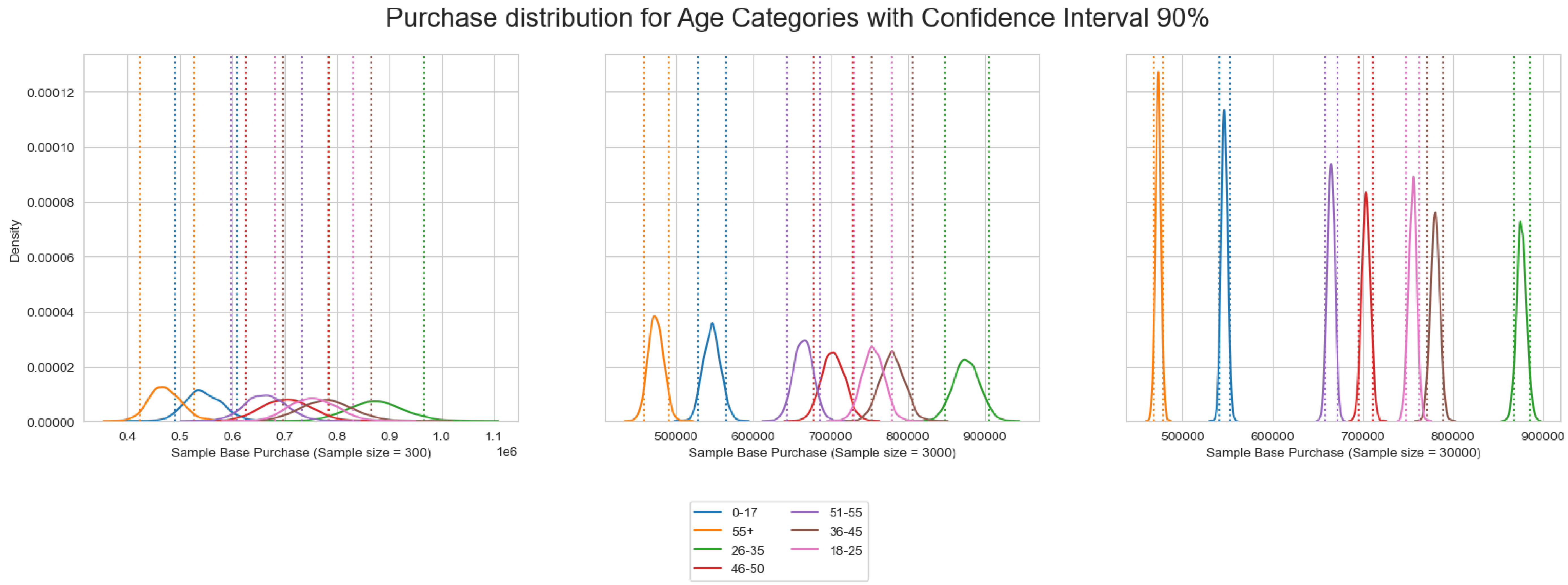
```
=> Confidence Interval for full data
Confidence Interval for 90: [  93826. 2188069.]
Confidence Interval for 95: [  76059. 2821212.]
Confidence Interval for 99: [  56587.         3663273.00000001]

================== Sample Means ==================
Sample Mean for 300: 664945.0025780001,
Sample Mean for 3000: 664331.6421404667,
Sample Mean for 30000: 664852.9776773

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [599514.93466667 733107.01983333]
Confidence Interval for 95: [585823.898  745856.4525]
Confidence Interval for 99: [564232.41346667 774374.24131667]

=> Confidence Interval for 3000
Confidence Interval for 90: [599514.93466667 733107.01983333]
Confidence Interval for 95: [585823.898  745856.4525]
Confidence Interval for 99: [564232.41346667 774374.24131667]

=> Confidence Interval for 30000
Confidence Interval for 90: [599514.93466667 733107.01983333]
Confidence Interval for 95: [585823.898  745856.4525]
Confidence Interval for 99: [564232.41346667 774374.24131667]


Age Category: 36-45
================== Full Data ==================
Poulation Mean: 780302.7180805485,
Poulation Standard Deviation: 893085.6961705135
=> Confidence Interval for full data
Confidence Interval for 90: [ 102802.3 2551975. ]
Confidence Interval for 95: [  86799.35       3332307.99999999]
Confidence Interval for 99: [  66124.63       4844986.24000001]

================== Sample Means ==================
Sample Mean for 300: 779175.6746813334,
Sample Mean for 3000: 780393.7656333334,
Sample Mean for 30000: 780266.19187858

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [696456.1085 867344.9485]
Confidence Interval for 95: [681400.16283333 884131.15766667]
Confidence Interval for 99: [649690.94688333 916820.39235   ]

=> Confidence Interval for 3000
Confidence Interval for 90: [696456.1085 867344.9485]
Confidence Interval for 95: [681400.16283333 884131.15766667]
Confidence Interval for 99: [649690.94688333 916820.39235   ]

=> Confidence Interval for 30000
Confidence Interval for 90: [696456.1085 867344.9485]
Confidence Interval for 95: [681400.16283333 884131.15766667]
Confidence Interval for 99: [649690.94688333 916820.39235   ]


Age Category: 18-25
================== Full Data ==================
Poulation Mean: 755338.3517305893,
Poulation Standard Deviation: 805438.0211720603
=> Confidence Interval for full data
Confidence Interval for 90: [ 100236.4 2480948.4]
Confidence Interval for 95: [  84688.9 3047737.4]
Confidence Interval for 99: [  62329.36 4005230.02]

================== Sample Means ==================
Sample Mean for 300: 755767.4419420001,
Sample Mean for 3000: 755538.2509532667,
Sample Mean for 30000: 755410.1003764999

================== Confidence Intervals ==================

=> Confidence Interval for 300
Confidence Interval for 90: [681103.46183333 833323.13683333]
Confidence Interval for 95: [666459.01441667 851231.75266667]
Confidence Interval for 99: [641266.08275    885218.68943333]

=> Confidence Interval for 3000
Confidence Interval for 90: [681103.46183333 833323.13683333]
Confidence Interval for 95: [666459.01441667 851231.75266667]
Confidence Interval for 99: [641266.08275    885218.68943333]

=> Confidence Interval for 30000
Confidence Interval for 90: [681103.46183333 833323.13683333]
Confidence Interval for 95: [666459.01441667 851231.75266667]
Confidence Interval for 99: [641266.08275    885218.68943333]
```

Observations

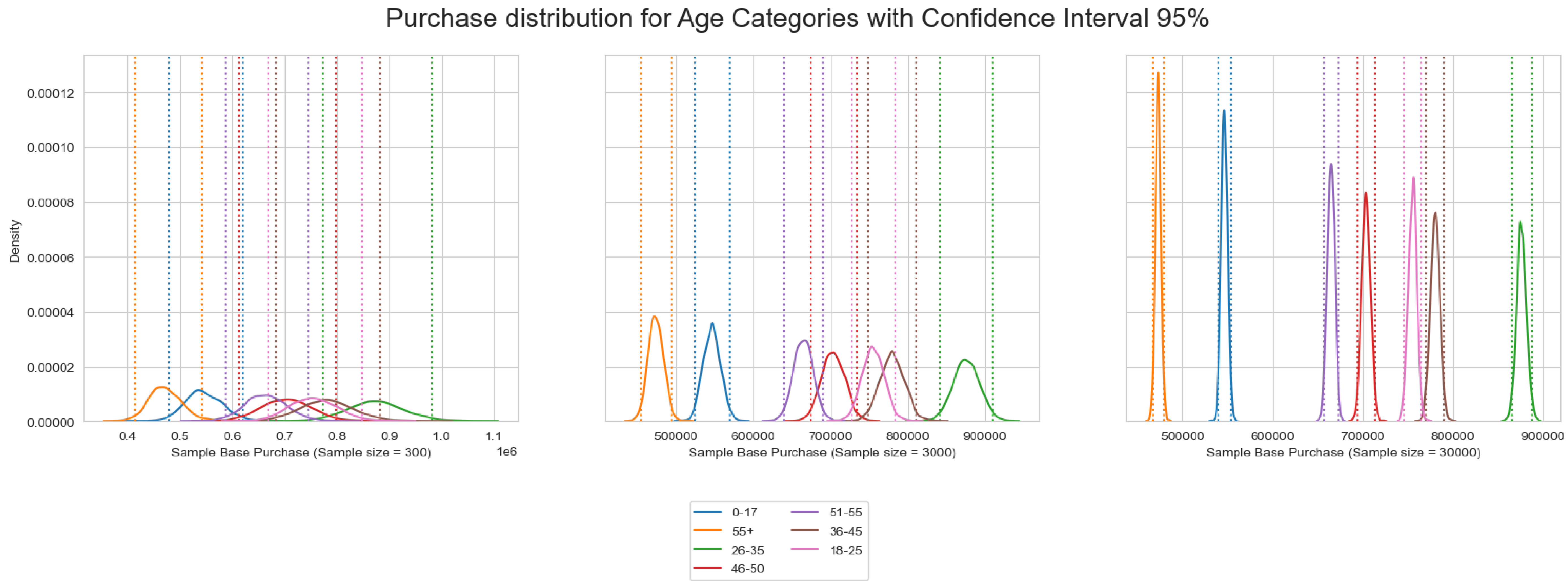
- Above data shows the population means and confidence intervals of different age groups for different sample sizes

In [ ]:
```python
def plot(ci):
    fig, axes = plt.subplots(1, 3, figsize=(20, 5), sharey=True)

    for cat in age_cats:
        kde_plot=sns.kdeplot(age_data_dict[cat]["sample_300"]["mean_sample"], ax=axes[0], label=cat)
        line_color = kde_plot.get_lines()[-1].get_c()
        axes[0].axvline(x=age_data_dict[cat]["sample_300"]["confidence_interval"][ci][0], color=line_color,  linestyle=':')
        axes[0].axvline(x=age_data_dict[cat]["sample_300"]["confidence_interval"][ci][1], color=line_color, linestyle=':');
    axes[0].set_xlabel(f'Sample Base Purchase (Sample size = 300)')

    for cat in age_cats:
        kde_plot=sns.kdeplot(age_data_dict[cat]["sample_3000"]["mean_sample"], ax=axes[1], label=cat)
        line_color = kde_plot.get_lines()[-1].get_c()
        axes[1].axvline(x=age_data_dict[cat]["sample_3000"]["confidence_interval"][ci][0], color=line_color, linestyle=':')
        axes[1].axvline(x=age_data_dict[cat]["sample_3000"]["confidence_interval"][ci][1], color=line_color, linestyle=':');
    axes[1].set_xlabel(f'Sample Base Purchase (Sample size = 3000)')

    for cat in age_cats:
        kde_plot=sns.kdeplot(age_data_dict[cat]["sample_30000"]["mean_sample"], ax=axes[2], label=cat)
        line_color = kde_plot.get_lines()[-1].get_c()
        axes[2].axvline(x=age_data_dict[cat]["sample_30000"]["confidence_interval"][ci][0], color=line_color, linestyle=':')
        axes[2].axvline(x=age_data_dict[cat]["sample_30000"]["confidence_interval"][ci][1], color=line_color, linestyle=':');

    axes[2].set_xlabel(f'Sample Base Purchase (Sample size = 30000)')


    plt.legend(bbox_to_anchor=(-0.8, -0.2), loc='upper center', ncol=2)
    fig.suptitle(f'Purchase distribution for Age Categories with Confidence Interval {ci}%', fontsize=20)

    # plt.tight_layout()
    plt.show();
```

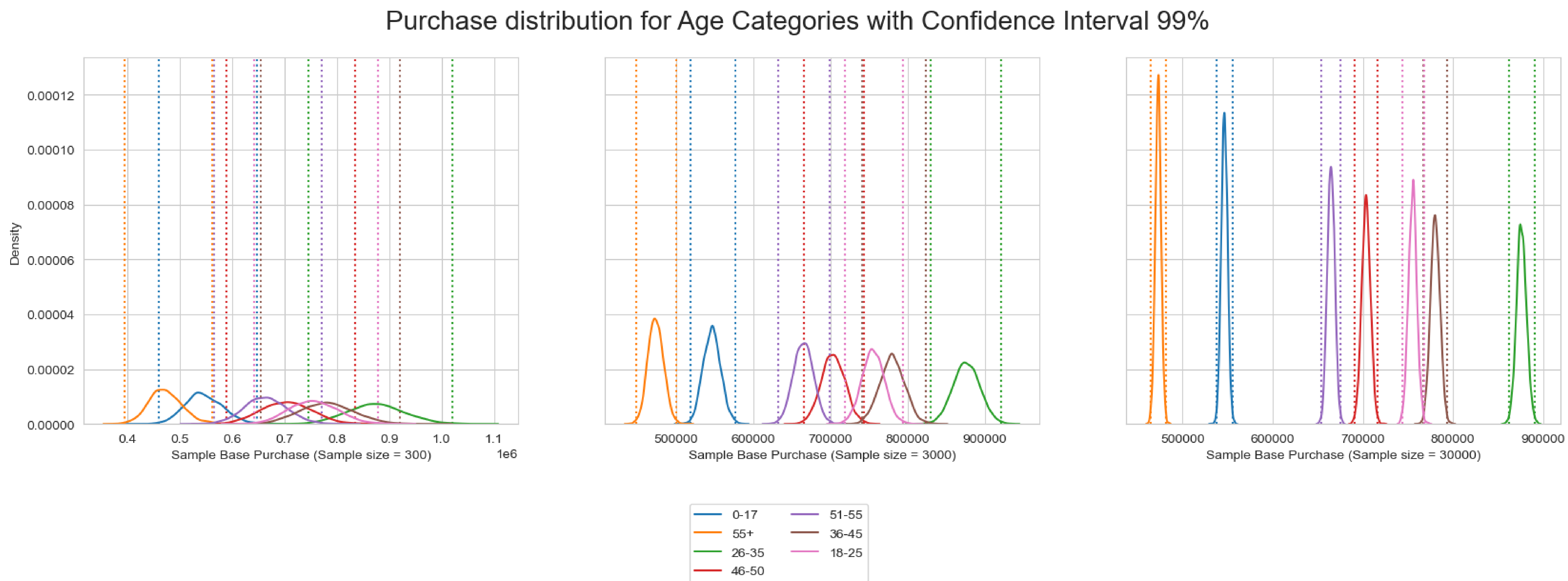
In [ ]:
```python
plot("90")
```



In [ ]:
```
plot("95")
```



In [ ]:
```
plot("99")
```



Insights

- Above plots show distribution of all age group for purchase with different sample count
- Theere appears to be significant difference between age group and purchase data
- Age group 55+ is least spending group
- Age group 26-35 is highest spending group
- As the sample count increases the width of confidence interval decreases.

Anova Test

In [ ]:
```
mean_30000_data=[]
for age_cat in age_cats:
    mean_30000_data.append(age_data_dict[age_cat]["sample_30000"]["mean_sample"])
```

In [ ]:
```
f_statistic, pval = f_oneway(mean_30000_data[0], mean_30000_data[1],mean_30000_data[2],
                             mean_30000_data[3], mean_30000_data[4], mean_30000_data[5], mean_30000_data[6])
f_statistic, pval
```

Out[ ]:
```
(4775042.425122179, 0.0)
```

In [ ]:
```
alpha = 0.05
if pval < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Reject the null hypothesis
```

Insights

- Using annova test we can say that there is a significant difference between age group and purchase data
- By using both plotting and annova test, we found out there is a significant difference in spending between age group

Recommendations

Below is the summary of customer spending patterns and potential strategies to increase sales:

User Related

- Male customers tend to spend more than female customers. In order to increase female customer spending, bigger discounts can be offered to female customers on existing products.
- Married customers tend to spend less than single customers. Offering couple discounts and adding small free toys as a gift for customers with kids can help boost spending among married customers.
- Customers aged 55 and above tend to spend the least. To encourage spending among older customers, bigger discounts can be offered, and delivery fees can be reduced or eliminated entirely.
- Customers between the ages of 26 and 35 tend to spend the most. Increasing social media ads and promotions can help acquire even more customers from this age group.

Product Related

- Product categories 1, 5, and 8 are the most profitable categories. Offering discounts and promotions on other categories and changing their placement in store or online can help increase sales.
- There should be more female only products in each category.

City Related

- City B has highest number of sales, City Category C has the highest number of users, while City A has the lowest user count. To increase sales in City A, higher discounts can be applied, and more promotional campaigns can be implemented.
- Customers who have been living in their current city for one year tend to be the biggest spenders. Customers who have been living in the same city for two, three, or four years tend to spend less. Implementing a loyalty program can help regain these customers.

Occupation Related

- Occupations 0, 4, and 7 have the highest number of users, while Occupation 8 has the lowest. Partnering with organizations or companies to offer promotions or discounts on certain categories for their employees can help increase sales among customers in Occupation 8.