1. Uploading Data

In [22]:

```
import math
import numpy as np
import pandas as pd
import seaborn as sns
from scipy.stats import norm
import matplotlib.pyplot as plt
```

In [23]:

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094 -O walmart_data.csv
```

```
--2023-03-27 17:30:19--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 99.84.192.218, 99.84.192.156, 99.84.192.31, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|99.84.192.218|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23027994 (22M) [text/plain]
Saving to: 'walmart_data.csv'

walmart_data.csv    100%[===================>]  21.96M  87.9MB/s    in 0.2s

2023-03-27 17:30:19 (87.9 MB/s) - 'walmart_data.csv' saved [23027994/23027994]
```

In [24]:

```
!ls
```

```
sample_data  walmart_data.csv
```

In [25]:

```
walmart_data = pd.read_csv("walmart_data.csv")
```

In [26]:

```
walmart_data.head()
```

Out[26]:

	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

In [ ]:

In [ ]:

In [ ]:

2. About Walmart and Problem Statement

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.

Problem Statment

The management team at Walmart Inc. seeks to investigate whether there is a significant difference in the spending habits of male and female customers during Black Friday sales. Specifically, the team wants to analyze the customer purchase behavior, specifically the purchase amount, against the customer's gender and other factors. The objective is to determine if there are statistically significant differences in the amount spent by male and female customers during Black Friday sales, and to provide insights that can help the business make better decisions regarding its marketing and sales strategies. The assumed customer population is 100 million, with 50 million customers being male and 50 million being female.

3. Analysing Data

In [27]:

```
walmart_data.head()
```

Out[27]:

	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

Comment:

> **We can onserve that there are total 10 columns in the Walmart dataset.**
>
> - **User_ID: User id is unique for each customer.**
> - **Product_ID: This defines the product id of the product that they have purchased.**
> - **Gender: This defines the gender of the customer "Male" or "Female".**
> - **Age: This shows the age group in which the customer falls under.**
> - **Occupation: This column has been masked, so we wouldn't be able to really anlyse the data from this column.**
> - **City_Category: Just contains the category represented with A, B and C.**
> - **Stay_In_Current_City_Years: This column contains how long a customer has stayed at their current city of living.**
> - **Marital_Status: This column has values metioned in 0 and 1, where considering 0 is customers who are Single or 1 is customers who are Partnered.**
> - **Product_Category: This column contains the catergory to which a particular product that was purchased by a customer.**
> - **Purchase: This column contains the amount spent by a customer in purchasing a product.**

In [28]:

```
walmart_data.isnull().sum()
```

Out[28]:

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
```

```
dtype: int64
```

In [29]:

```
Total_Null_Values = walmart_data.isnull().sum().sum()
print(f'Total number of null values in provided Walmart Data: {Total_Null_Values}')
```

```
Total number of null values in provided Walmart Data: 0
```

Comment:

> **We can see that none of the columns and also in entire Walmart dataset there are non Nan or Null values.**

In [30]:

```
print(f'Total Number of Records in Walmart Data: {walmart_data.shape[0]}')
print(f'Total Number of Columns in Walmart Data: {walmart_data.shape[1]}')
```

```
Total Number of Records in Walmart Data: 550068
Total Number of Columns in Walmart Data: 10
```

Comment:

> **We already saw above that there are 10 columns or characteristics of a customer in each record.**
>
> **There a total of 550,068 records in the Walmart Dataset**

In [31]:

```
Number_of_duplicates = len(walmart_data[walmart_data.duplicated()])
print(f'Total Number of duplicate records in Walmart Data: {Number_of_duplicates}')
```

```
Total Number of duplicate records in Walmart Data: 0
```

Comment:

> **Here we checked if there are any duplicate records in the entire Walmart Dataset, and we can se there are none.**

In [32]:

```
walmart_data.info()
```

```
<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
```

In [33]:

```
walmart_data["Marital_Status"].unique()
```

Out[33]:

Out[35]:

```
array([0, 1])
```

Comment:

> **As we can observe above that the "Marital_Status" has integer values 0 and 1, let's update them from 0 to Single and 1 to Partnered and convert the column to string.**

In [36]:

```
walmart_data['User_ID'] = walmart_data['User_ID'].astype('str')
walmart_data['Product_Category'] = walmart_data['Product_Category'].astype('str')
walmart_data['Marital_Status'] = walmart_data['Marital_Status'].astype('str')
walmart_data['Marital_Status'] = walmart_data['Marital_Status'].replace(['0', '1'], ['Single','Partnered'])
```

In [37]:

```
walmart_data['Marital_Status'].unique()
```

Out[37]:

```
array(['Single', 'Partnered'], dtype=object)
```

Comment:

> **We have now updated the values of column Marital_Status to data type 'Object', also we have updated the values from 0, 1 to Single, Partnered.**

In [38]:

```
walmart_data.describe(include = 'all').round(2)
```

Out[38]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
count	550068	550068	550068	550068	550068.00	550068	550068	550068	550068	550068.00
unique	5891	3631	2	7	NaN	3	5	2	20	NaN
top	1001680	P00265242	M	26-35	NaN	B	1	Single	5	NaN
freq	1026	1880	414259	219587	NaN	231173	193821	324731	150933	NaN
mean	NaN	NaN	NaN	NaN	8.08	NaN	NaN	NaN	NaN	9263.97
std	NaN	NaN	NaN	NaN	6.52	NaN	NaN	NaN	NaN	5023.07
min	NaN	NaN	NaN	NaN	0.00	NaN	NaN	NaN	NaN	12.00
25%	NaN	NaN	NaN	NaN	2.00	NaN	NaN	NaN	NaN	5823.00
50%	NaN	NaN	NaN	NaN	7.00	NaN	NaN	NaN	NaN	8047.00
75%	NaN	NaN	NaN	NaN	14.00	NaN	NaN	NaN	NaN	12054.00
max	NaN	NaN	NaN	NaN	20.00	NaN	NaN	NaN	NaN	23961.00

In [39]:

```
walmart_data.describe(include = 'object')
```

Out[39]:

	User_ID	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
count	550068	550068	550068	550068	550068	550068	550068	550068
unique	5891	3631	2	7	3	5	2	20
top	1001680	P00265242	M	26-35	B	1	Single	5
freq	1026	1880	414259	219587	231173	193821	324731	150933

Comment:

We can see that the columns mentioned below are all data type object.

```
User_ID
Product_ID
Gender
Age
City_Category
Stay_In_Current_City_Years
Marital Status
Product_Category
```

- **Customer with User_ID 1001680 has made the highest number of purchases.**
- **Top most bought Product_ID is P00265242 with a total unique transactions of 3631.**
- **MALE are the top number of people who have made purchases in the given Walmart Dataset.**
- **Customers who have made most of the purchases fall under the age group of 26-35.**
- **City_Category B has got the highest number of transactions done.**
- **The higher number of customers have stayed in their current city for 5 years.**
- **Singles are the one who made more purchases.**
- **Product_Category 5 is the highest purchase catergory of all the products.**

In [40]:

```
walmart_data.describe().round(2)
```

Out[40]:

	Occupation	Purchase
count	550068.00	550068.00
mean	8.08	9263.97
std	6.52	5023.07
min	0.00	12.00
25%	2.00	5823.00
50%	7.00	8047.00
75%	14.00	12054.00
max	20.00	23961.00

Comment:

The following columns are of data type int: Occupation Purchase

- **The minimum occupation of a customer is 0 and the maximum occupation of customer is 20.**
- **Minimum purchase from Walmart was done at 12.00 the highest purchase was done at 23961.00. The average spent by all customers combined is around 8047.00.**

In [41]:

```
for i in walmart_data.columns:
  print("-" * 30)
  print (i)
  print("-" * 30)
  if walmart_data[i].dtypes == "int64":
    print (f'The MEAN of column {i}: {walmart_data[i].mean().round(2)}')
    print (f'The total number of UNIQUE values in {i}: {walmart_data[i].nunique()}')
    print (f'The total number of UNIQUE values in {i}: {walmart_data[i].unique()}\n')
  else:
    print (f'The MODE of column {i}: {walmart_data[i].mode()[0]}')
```

```
    print (f'The total number of UNIQUE values in {i}: {walmart_data[i].nunique()}')
    print (f'The total number of UNIQUE values in {i}: {walmart_data[i].unique()}\n')
```

```
------------------------------
User_ID
------------------------------
The MODE of column User_ID: 1001680
The total number of UNIQUE values in User_ID: 5891
The total number of UNIQUE values in User_ID: ['1000001' '1000002' '1000003' ... '1004113' '1005391' '1001529']

------------------------------
Product_ID
------------------------------
The MODE of column Product_ID: P00265242
The total number of UNIQUE values in Product_ID: 3631
The total number of UNIQUE values in Product_ID: ['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']

------------------------------
Gender
------------------------------
The MODE of column Gender: M
The total number of UNIQUE values in Gender: 2
The total number of UNIQUE values in Gender: ['F' 'M']

------------------------------
Age
------------------------------
The MODE of column Age: 26-35
The total number of UNIQUE values in Age: 7
The total number of UNIQUE values in Age: ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']

------------------------------
Occupation
------------------------------
The MEAN of column Occupation: 8.08
The total number of UNIQUE values in Occupation: 21
The total number of UNIQUE values in Occupation: [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]

------------------------------
City_Category
------------------------------
The MODE of column City_Category: B
The total number of UNIQUE values in City_Category: 3
The total number of UNIQUE values in City_Category: ['A' 'C' 'B']

------------------------------
Stay_In_Current_City_Years
------------------------------
The MODE of column Stay_In_Current_City_Years: 1
The total number of UNIQUE values in Stay_In_Current_City_Years: 5
The total number of UNIQUE values in Stay_In_Current_City_Years: ['2' '4+' '3' '1' '0']

------------------------------
Marital_Status
------------------------------
The MODE of column Marital_Status: Single
The total number of UNIQUE values in Marital_Status: 2
The total number of UNIQUE values in Marital_Status: ['Single' 'Partnered']

------------------------------
Product_Category
------------------------------
The MODE of column Product_Category: 5
The total number of UNIQUE values in Product_Category: 20
The total number of UNIQUE values in Product_Category: ['3' '1' '12' '8' '5' '4' '2' '6' '14' '11' '13' '15' '7' '16' '18' '10'
 '17' '9' '20' '19']

------------------------------
Purchase
------------------------------
The MEAN of column Purchase: 9263.97
```

```
The total number of UNIQUE values in Purchase: 18105
The total number of UNIQUE values in Purchase: [ 8370 15200  1422 ...   135   123   613]
```

Comment:

> **We can see the mean and median of each column along with number of unique values in the column and the unique values itself.**

In [49]:

```
walmart_data[["Purchase"]].describe().round(2).T
```

Out[49]:

	count	mean	std	min	25%	50%	75%	max
Purchase	550068.0	9263.97	5023.07	12.0	5823.0	8047.0	12054.0	23961.0

4. Non-Graphical Analysis

In [101]:

```
print (f'Number of Unique Gender: {walmart_data["Gender"].count()}')
print (f'Number of Unique Gender: {walmart_data["Gender"].nunique()}')
print (f'Value Counts of Gender:\n{walmart_data["Gender"].value_counts()}')
```

```
Number of Unique Gender: 550068
Number of Unique Gender: 2
Value Counts of Gender:
M    414259
F    135809
Name: Gender, dtype: int64
```

***Comment:* Gender Count in Walmart Data**

> **As we can observe from above that out of the total records of 550,068 records the total records of male are 414,259 and total records of female are 135,809. So we have most records of male over female, which tells us most of the purchases or transactions are done by male customers.**

In [102]:

```
walmart_data.groupby("Gender")["Purchase"].sum().sort_values(ascending = False)
```

Out[102]:

```
Gender
M    3909580100
F    1186232642
Name: Purchase, dtype: int64
```

***Comment:* Total Purchases by Gender in Walmart Data**

> **As we observed already that the total number records or transactions are done by male, we can see the same trend to follow when we try to find the sum of purchases sorted by Gender to understand which gender made the highest purcahses. Undoubtedly Male customers have made the highest number of total purchases.**

In [103]:

```
walmart_data["Age"].value_counts().sort_index()
```

Out[103]:

```
0-17      15102
```

```
18-25      99660
26-35     219587
36-45     110013
46-50      45701
51-55      38501
55+        21504
Name: Age, dtype: int64
```

Comment: **Age Group Count in Walmart Data**

- **We have a total of 7 different age groups, starting from**
  - **0-17 Age Group**
  - **18-25 Age Group**
  - **26-35 Age Group**
  - **36-45 Age Group**
  - **46-50 Age Group**
  - **51-55 Age Group**
  - **55+ Age Group**

> **Out of all these age groups we can understand that the customers who have made the most transactions or purchases fall under the age group of 26 - 35 years with almost 219,587 records.**

In [104]:

```
print (f'Number of Total Records: {walmart_data["Marital_Status"].count()}')
print (f'Number of Total types of Marital Status: {walmart_data["Marital_Status"].nunique()}')
print (f'Types of Unique Marital Status: {walmart_data["Marital_Status"].unique()}')
print (f'Number of Total Singles:{walmart_data["Marital_Status"].value_counts()[0]}')
print (f'Number of Total Couples:{walmart_data["Marital_Status"].value_counts()[1]}')
```

```
Number of Total Records: 550068
Number of Total types of Marital Status: 2
Types of Unique Marital Status: ['Single' 'Partnered']
Number of Total Singles:324731
Number of Total Couples:225337
```

Comment: **Marital Status Counts in Walmart Data**

> **Out of the total 550,068 records we only two types of Marital Status available which are marked as 0 and 1. So 0 being Singles and 1 being Couple, we can see the highest purchases are done by customers who are single in total number of purchases.**

In [105]:

```
print (f'Total purchases made by Singles: {walmart_data.groupby("Marital_Status")["Purchase"].sum()[0]}')
print (f'Total purchases made by Couples: {walmart_data.groupby("Marital_Status")["Purchase"].sum()[1]}')
```

```
Total purchases made by Singles: 2086885295
Total purchases made by Couples: 3008927447
```

Comment: **Total Purchases by Marital Status in Walmart Data**

> **As already observed that total number highest transactions are done more by Singles. Continuing the observation even the total amount of purchases done by Singles is more than Couples.**

In [106]:

```
print (f'Total Number of Unique Products Purchased: {walmart_data["Product_ID"].nunique()}\n')
print (f'Top 10 Products Purchased by total number purchased:\n{walmart_data["Product_ID"].value_counts().sort_values(ascending = False)[:10]}')
```

```
Total Number of Unique Products Purchased: 3631

Top 10 Products Purchased by total number purchased:
P00265242    1880
P00025442    1615
P00110742    1612
```

```
P00112142    1562
P00057642    1470
P00184942    1440
P00046742    1438
P00058042    1422
P00059442    1406
P00145042    1406
Name: Product_ID, dtype: int64
```

Comment: **Top 10 Products Purchased by total number of purchases as per Walmart Data**

> **There are a total of 3,631 products purchased as per the records in Walmart Data. We can observe the top most product prurchase with a total number of purchases of 1,880 is Product ID: P00265242. We can observe the data doesn't contain the product names but just the unique product id we can only know which product ids are highly purchase of the total records.**

In [107]:

```python
print (f'Top 10 Products Purchased by total amount:\n{walmart_data.groupby("Product_ID")["Purchase"].sum().sort_values(ascending = False)[:10]}')
```

```
Top 10 Products Purchased by total amount:
Product_ID
P00025442    27995166
P00110742    26722309
P00255842    25168963
P00059442    24338343
P00184942    24334887
P00112142    24216006
P00110942    23639564
P00237542    23425576
P00057642    23102780
P00010742    22164153
Name: Purchase, dtype: int64
```

Comment: **Top 10 Products Purchase by Total Amount in Walmart Data**

> **Though the Product_ID: P00265242 was the highest purchase product by total number of transactions, after checking which product was the highest total amount spend on then we can observe the Product_ID changed to P00025442 which also happens to be the second highest bought product in the top 10 list pf products purchased by total number of purchases.**

In [108]:

```python
print (f'Number of Total Records: {walmart_data["City_Category"].count()}')
print (f'Number of Total types of City Category: {walmart_data["City_Category"].nunique()}')
print (f'Types of City Category: {walmart_data["City_Category"].unique()}\n')
print (f'Number of Total Purchases by City_Category:\n{walmart_data["City_Category"].value_counts().sort_index()}')
```

```
Number of Total Records: 550068
Number of Total types of City Category: 3
Types of City Category: ['A' 'C' 'B']

Number of Total Purchases by City_Category:
A    147720
B    231173
C    171175
Name: City_Category, dtype: int64
```

Comment: **Total Count of City Category in Walmart Data**

> **There are a total of 3 City Caterogies: A, B and C. Out of the 3 City Category 'B' has got the highest number of purchases.**

In [109]:

```python
print (f'Total Amount Purchased by City Category:\n{walmart_data.groupby("City_Category")["Purchase"].sum()}')
```

```
Total Amount Purchased by City Category:
City_Category
```

```
A    1316471661
B    2115533605
C    1663807476
Name: Purchase, dtype: int64
```

Comment: **Total Amount Purchased by City Category in Walmart Data**

> **Category B not only has the highest number of purchases but also has the highest total amount of purchases as per the data provided.**

In [110]:

```
walmart_data.head()
```

Out[110]:

	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	Single	3	8370
1	1000001	P00248942	F	0-17	10	A	2	Single	1	15200
2	1000001	P00087842	F	0-17	10	A	2	Single	12	1422
3	1000001	P00085442	F	0-17	10	A	2	Single	12	1057
4	1000002	P00285442	M	55+	16	C	4+	Single	8	7969

5. Visual Analysis

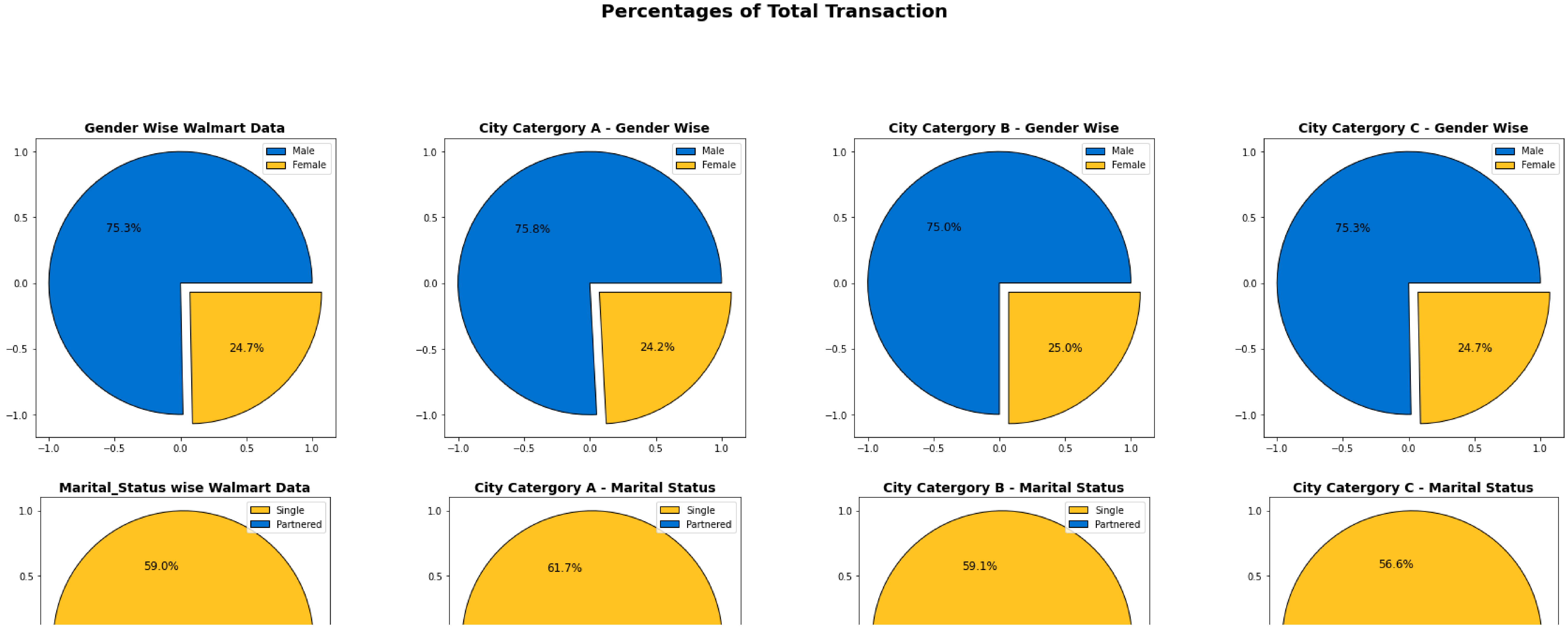
Pie Charts

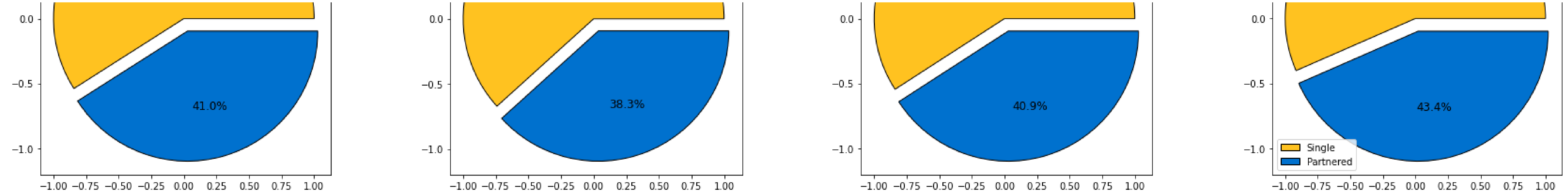
In [115]:

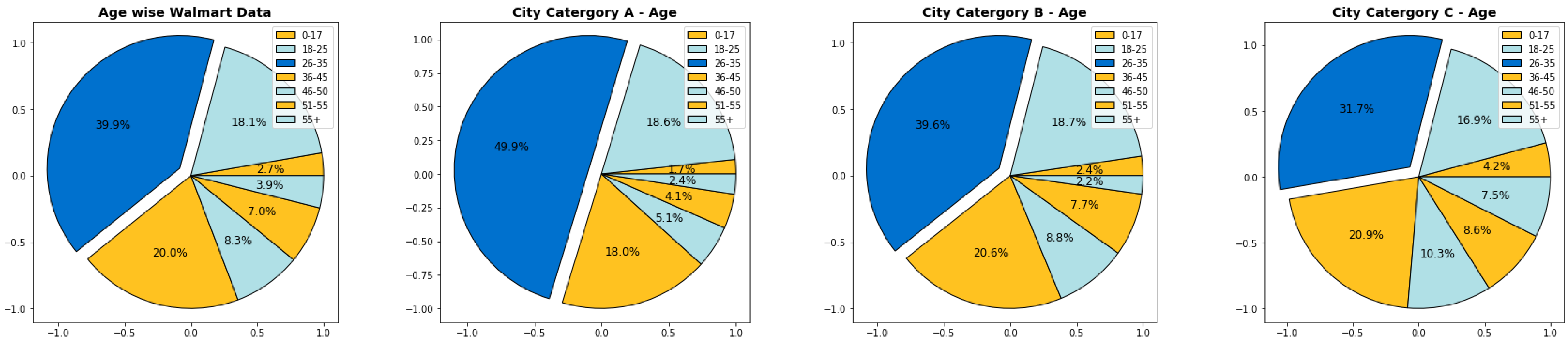
```
plt.figure(figsize = (30,20))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

plt.subplot(3,4,1)
plt.pie(walmart_data['Gender'].value_counts().values,
        colors = [walmart_blue, walmart_orange],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(["Male", "Female"])
plt.title('Gender Wise Walmart Data', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,2)
plt.pie(walmart_data[walmart_data['City_Category'] == 'A']['Gender'].value_counts().values,
        colors = [walmart_blue, walmart_orange],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(["Male", "Female"])
plt.title('City Catergory A - Gender Wise', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,3)
plt.pie(walmart_data[walmart_data['City_Category'] == 'B']['Gender'].value_counts().values,
        colors = [walmart_blue, walmart_orange],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
```

```
plt.legend(["Male", "Female"])
plt.title('City Catergory B - Gender Wise', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,4)
plt.pie(walmart_data[walmart_data['City_Category'] == 'C']['Gender'].value_counts().values,
        colors = [walmart_blue, walmart_orange],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(["Male", "Female"])
plt.title('City Catergory C - Gender Wise', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,5)
plt.pie(walmart_data['Marital_Status'].value_counts().values,
        colors = [walmart_orange, walmart_blue],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data['Marital_Status'].value_counts().index)
plt.title('Marital_Status wise Walmart Data', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,6)
plt.pie(walmart_data[walmart_data['City_Category'] == 'A']['Marital_Status'].value_counts().values,
        colors = [walmart_orange, walmart_blue],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data['Marital_Status'].value_counts().index)
plt.title('City Catergory A - Marital Status', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,7)
plt.pie(walmart_data[walmart_data['City_Category'] == 'B']['Marital_Status'].value_counts().values,
        colors = [walmart_orange, walmart_blue],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data['Marital_Status'].value_counts().index)
plt.title('City Catergory B - Marital Status', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,8)
plt.pie(walmart_data[walmart_data['City_Category'] == 'C']['Marital_Status'].value_counts().values,
        colors = [walmart_orange, walmart_blue],
        explode = (0,0.1),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data['Marital_Status'].value_counts().index)
plt.title('City Catergory C - Marital Status', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,9)
plt.pie(walmart_data['Age'].value_counts().sort_index().values,
        colors = [ walmart_orange, 'powderblue', walmart_blue, walmart_orange, 'powderblue', walmart_orange, 'powderblue'],
        explode = (0,0,0.1,0,0,0,0),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data['Age'].value_counts().sort_index().index, loc = 'upper right')
plt.title('Age wise Walmart Data', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,10)
plt.pie(walmart_data[walmart_data['City_Category'] == 'A']['Age'].value_counts().sort_index().values,
        colors = [ walmart_orange, 'powderblue', walmart_blue, walmart_orange, 'powderblue', walmart_orange, 'powderblue'],
```

```
        explode = (0,0,0.1,0,0,0,0),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data[walmart_data['City_Category'] == 'A']['Age'].value_counts().sort_index().index, loc = 'upper right')
plt.title('City Catergory A - Age', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,11)
plt.pie(walmart_data[walmart_data['City_Category'] == 'B']['Age'].value_counts().sort_index().values,
        colors = [ walmart_orange, 'powderblue', walmart_blue, walmart_orange, 'powderblue', walmart_orange, 'powderblue'],
        explode = (0,0,0.1,0,0,0,0),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data[walmart_data['City_Category'] == 'B']['Age'].value_counts().sort_index().index, loc = 'upper right')
plt.title('City Catergory B - Age', fontsize = 14, fontweight = 'semibold')

plt.subplot(3,4,12)
plt.pie(walmart_data[walmart_data['City_Category'] == 'C']['Age'].value_counts().sort_index().values,
        colors = [ walmart_orange, 'powderblue', walmart_blue, walmart_orange, 'powderblue', walmart_orange, 'powderblue'],
        explode = (0,0,0.1,0,0,0,0),
        autopct = '%.1f%%',
        textprops = {'color': 'black', 'fontsize': 'large'},
        wedgeprops = {'edgecolor': 'black'},
        frame = True)
plt.legend(walmart_data[walmart_data['City_Category'] == 'C']['Age'].value_counts().sort_index().index, loc = 'upper right')
plt.title('City Catergory C - Age', fontsize = 14, fontweight = 'semibold')

plt.suptitle('Percentages of Total Transaction', fontsize = 20, fontweight = 'bold')
plt.show()
```







Comment: **Percentages of total number purchases and total amount purchased by Gender and Marital Status in Walmart Data**

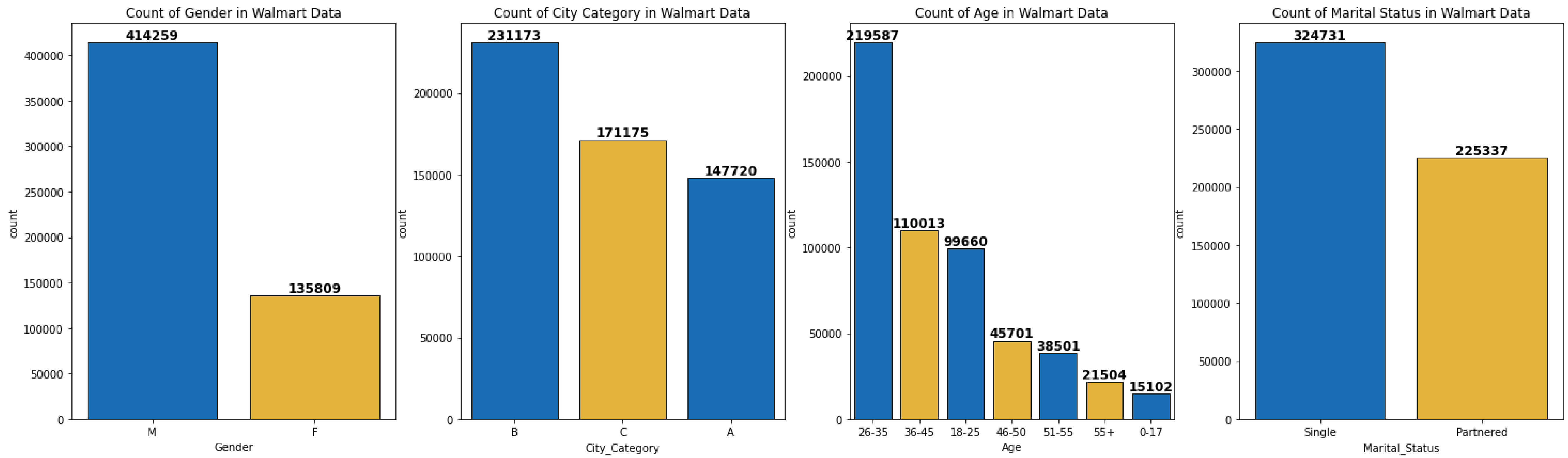
- **When we compare the entire Walmart dataset or City_Category wise, we can observe that the most number of purchases were done by *Gender: Male*.**
- **When we compare the entire Walmart dataset or City_Category wise, we can observe that the most number of purchases were done by *Marital_Status: Single*.**
- **When we compare the entire Walmart dataset or City_Category wise, we can observe that the most number of purchases were done by *Age: 26-35*.**

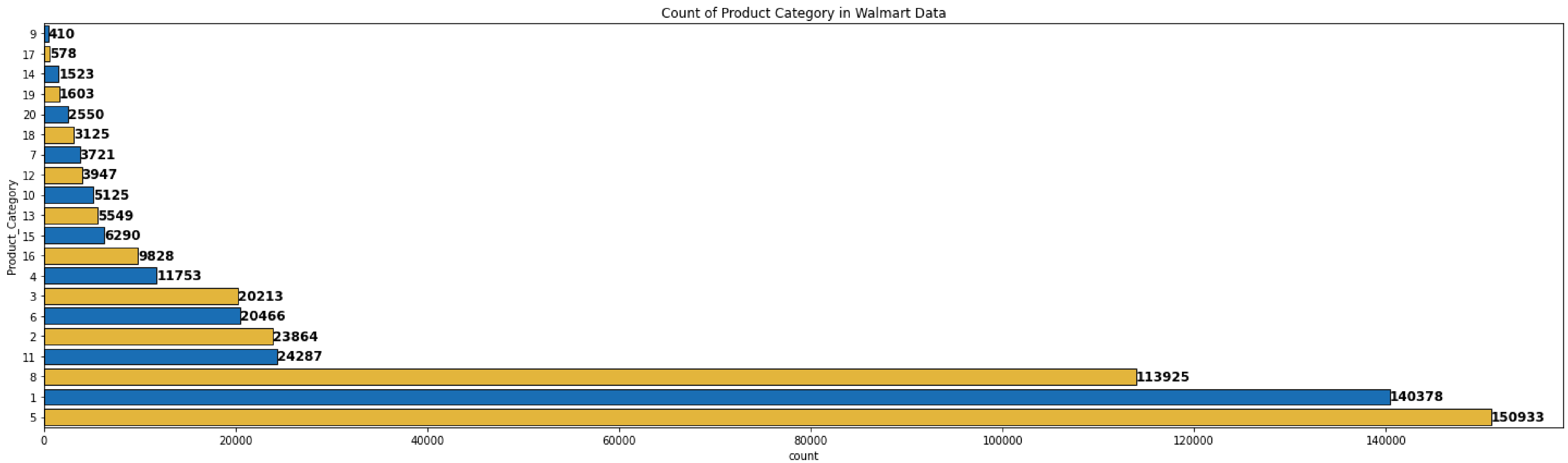
Bar Plots

In [117]:

```
plt.figure(figsize = (25,15))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

plt.subplot(2,4,1)
ax = sns.countplot (data = walmart_data, x = "Gender", order = walmart_data['Gender'].value_counts(ascending = False).index, palette = [walmart_blue, walmart_orange], edgecolor = '0.1')
for count in ax.containers:
  ax.bar_label(count, fontsize = 12, fontweight ='semibold')

plt.title('Count of Gender in Walmart Data')

plt.subplot(2,4,2)
ax = sns.countplot (data = walmart_data, x = "City_Category", order = walmart_data['City_Category'].value_counts(ascending = False).index, palette = [walmart_blue, walmart_orange], edgeco
lor = '0.1')
for count in ax.containers:
  ax.bar_label(count, fontsize = 12, fontweight ='semibold')
plt.title('Count of City Category in Walmart Data')

plt.subplot(2,1,2)
ax = sns.countplot (data = walmart_data, y = "Product_Category", order = walmart_data['Product_Category'].value_counts(ascending = True).index, palette = [walmart_blue, walmart_orange], e
dgecolor = '0.1')
for count in ax.containers:
  ax.bar_label(count, fontsize = 12, fontweight ='semibold')
```

```
plt.title('Count of Product Category in Walmart Data')

plt.subplot(2,4,3)
ax = sns.countplot (data = walmart_data, x = "Age", order = walmart_data['Age'].value_counts(ascending = False).index, palette = [walmart_blue, walmart_orange], edgecolor = '0.1')
for count in ax.containers:
  ax.bar_label(count, fontsize = 12, fontweight ='semibold')
plt.title('Count of Age in Walmart Data')

plt.subplot(2,4,4)
ax = sns.countplot (data = walmart_data, x = "Marital_Status", order = walmart_data['Marital_Status'].value_counts(ascending = False).index, palette = [walmart_blue, walmart_orange], edge
color = '0.1')
for count in ax.containers:
  ax.bar_label(count, fontsize = 12, fontweight ='semibold')
plt.title('Count of Marital Status in Walmart Data')

plt.show()
```



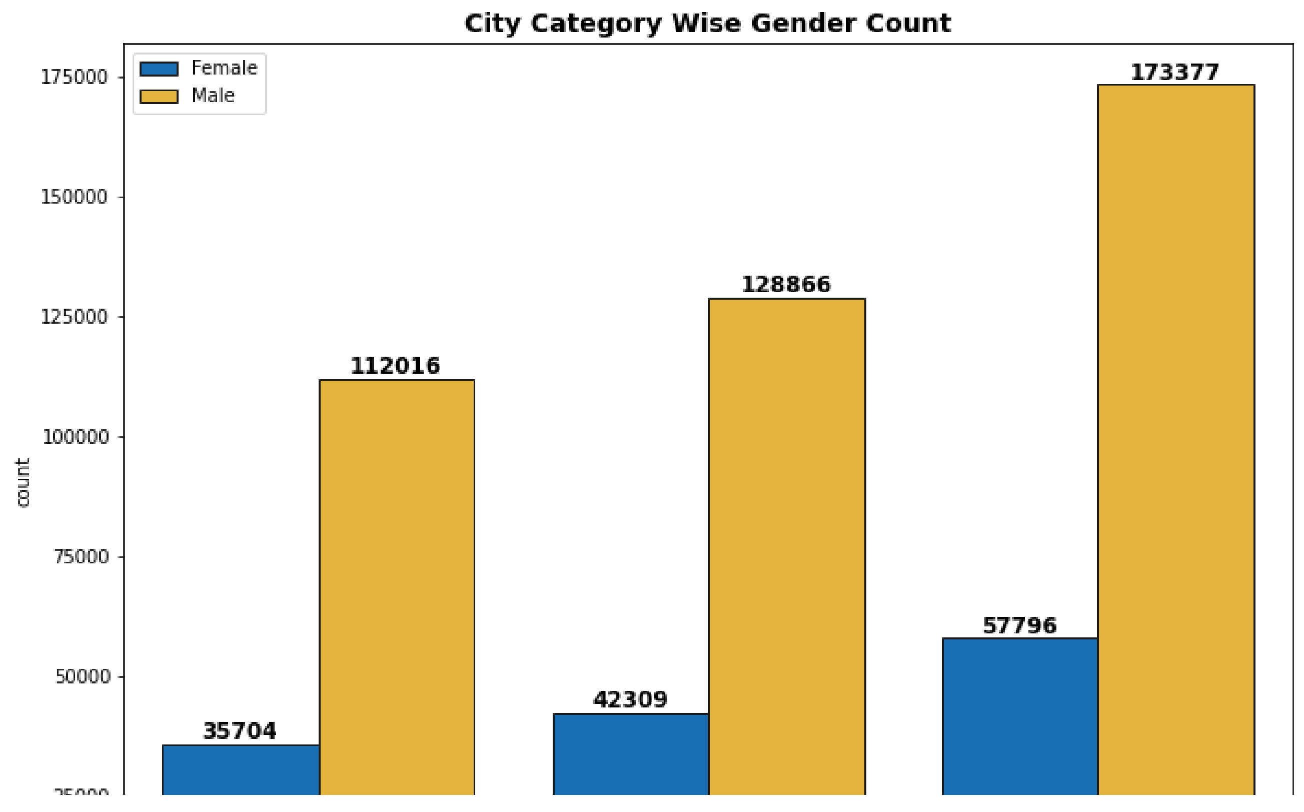


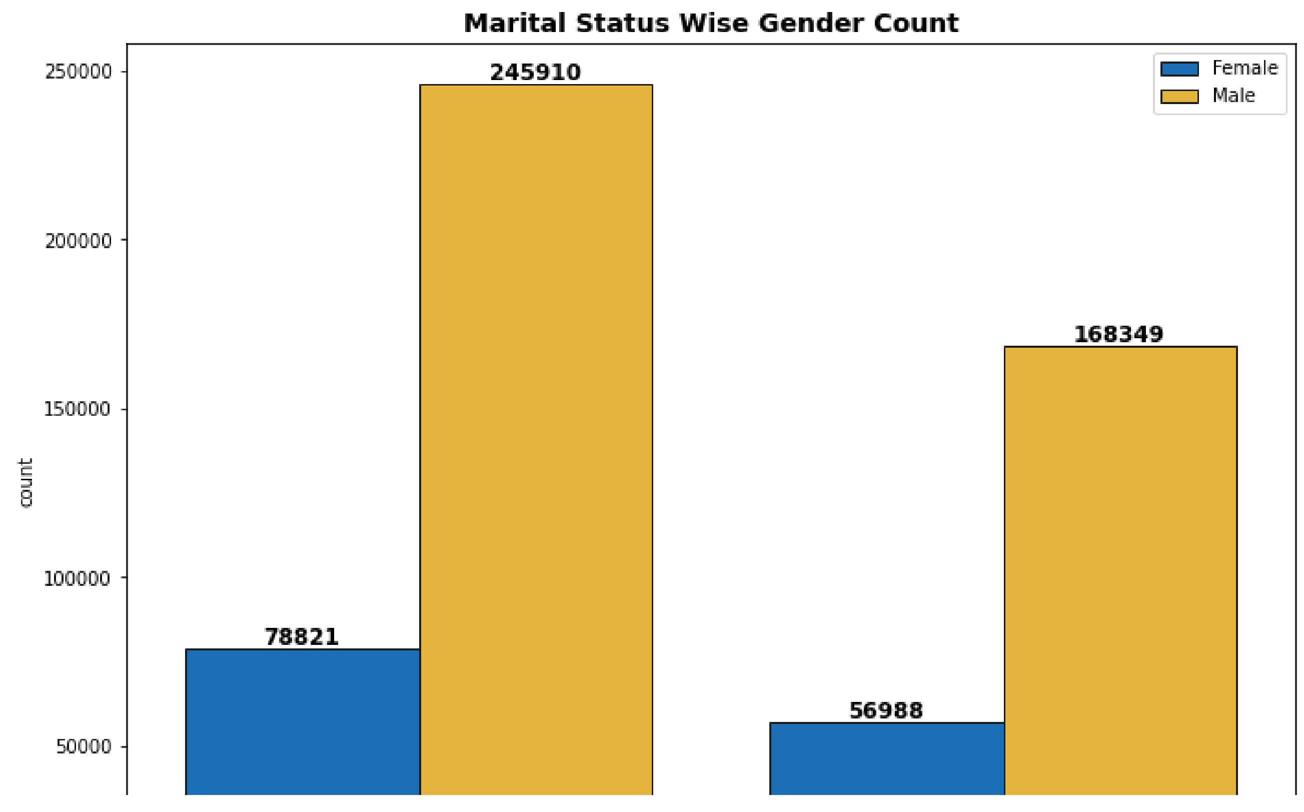
Comment:

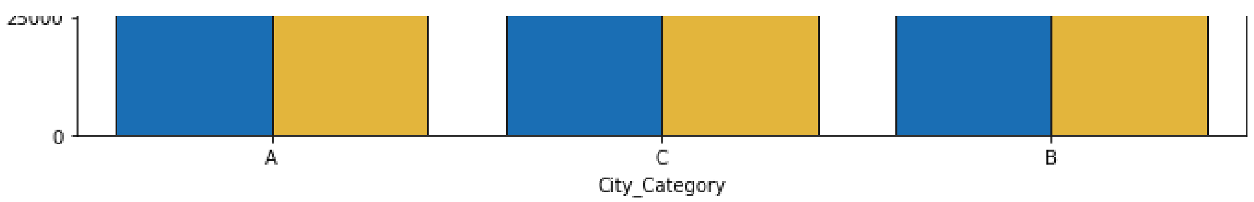
- **The highest count of Male have made the purchases.**
- **City_Category B has recorded the highest number of purchases.**
- **Age 26-35 customers have made the highest number of purchases.**
- **Single customers have made the highest number of purchases.**
- **Top 3 Product_Category which has got the highest purchases are 5, 1, 8 in the order mentioned.**

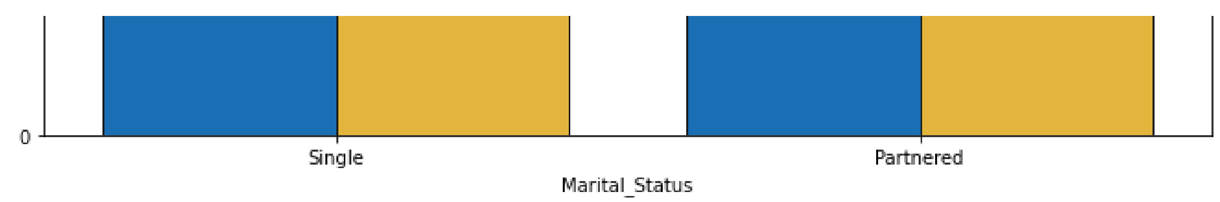
In [50]:

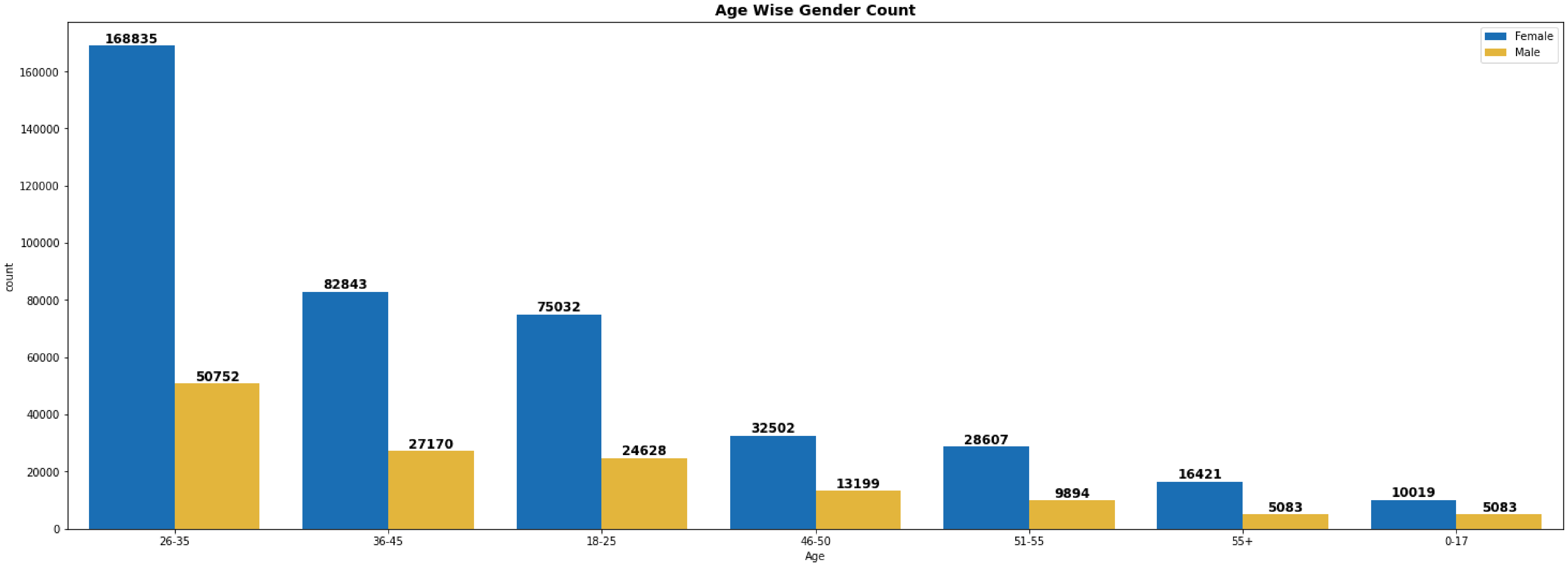
```
plt.figure(figsize = (25, 40))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

plt.subplot(4,2,1)
ax = sns.countplot(data = walmart_data, x = "City_Category", hue = "Gender", palette = [walmart_blue, walmart_orange], edgecolor = '0.1')
for container in ax.containers:
  ax.bar_label(container, fontsize = 12, fontweight = 'semibold')
plt.legend(['Female','Male'])
plt.title('City Category Wise Gender Count', fontsize = 14, fontweight = 'bold')

plt.subplot(4,2,2)
ax = sns.countplot(data = walmart_data, x = "Marital_Status", hue = "Gender", palette = [walmart_blue, walmart_orange], edgecolor = '0.1')
for container in ax.containers:
  ax.bar_label(container, fontsize = 12, fontweight = 'semibold')
plt.legend(['Female','Male'])
plt.title('Marital Status Wise Gender Count', fontsize = 14, fontweight = 'bold')

plt.subplot(4,1,2)
ax = sns.countplot(data = walmart_data, x = "Age", hue = "Gender", order = walmart_data['Age'].value_counts(ascending = False).index, hue_order = walmart_data['Gender'].value_counts(ascen
ding = False).index, palette = [walmart_blue, walmart_orange])
for container in ax.containers:
  ax.bar_label(container, fontsize = 12, fontweight = 'semibold')
plt.legend(['Female','Male'])
plt.title('Age Wise Gender Count', fontsize = 14, fontweight = 'bold')

plt.subplot(4,1,3)
ax = sns.countplot(data = walmart_data, x = "Product_Category", hue = "Gender", order = walmart_data['Product_Category'].value_counts(ascending = False).index,
                  hue_order = walmart_data['Gender'].value_counts(ascending = False).index, palette = [walmart_blue, walmart_orange])
for container in ax.containers:
  ax.bar_label(container, fontsize = 10, fontweight = 'semibold')
plt.legend(['Female','Male'])
plt.title('Product Category wise Gender Count', fontsize = 14, fontweight = 'bold')

plt.show()
```

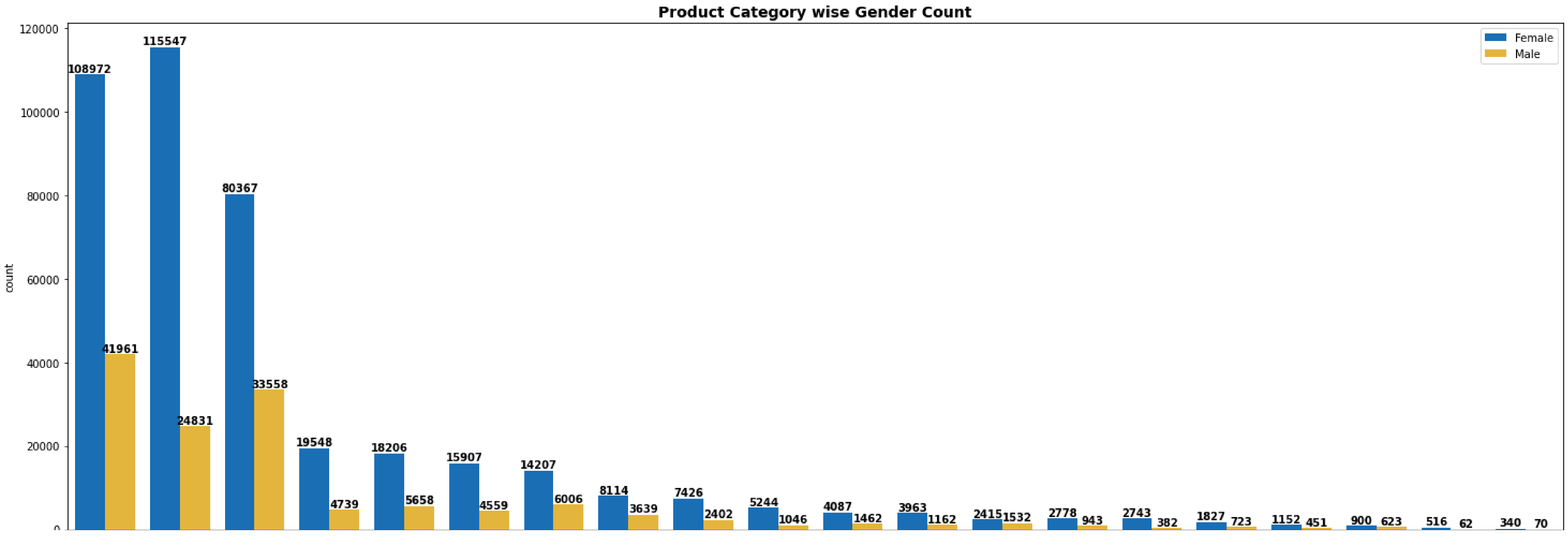





25000
0
A
C
B
City_Category


0
Single
Partnered
Marital_Status


Age Wise Gender Count
Female
Male
count
Age
26-35
36-45
18-25
46-50
51-55
55+
0-17
168835
50752
82843
27170
75032
24628
32502
13199
28607
9894
16421
5083
10019
5083

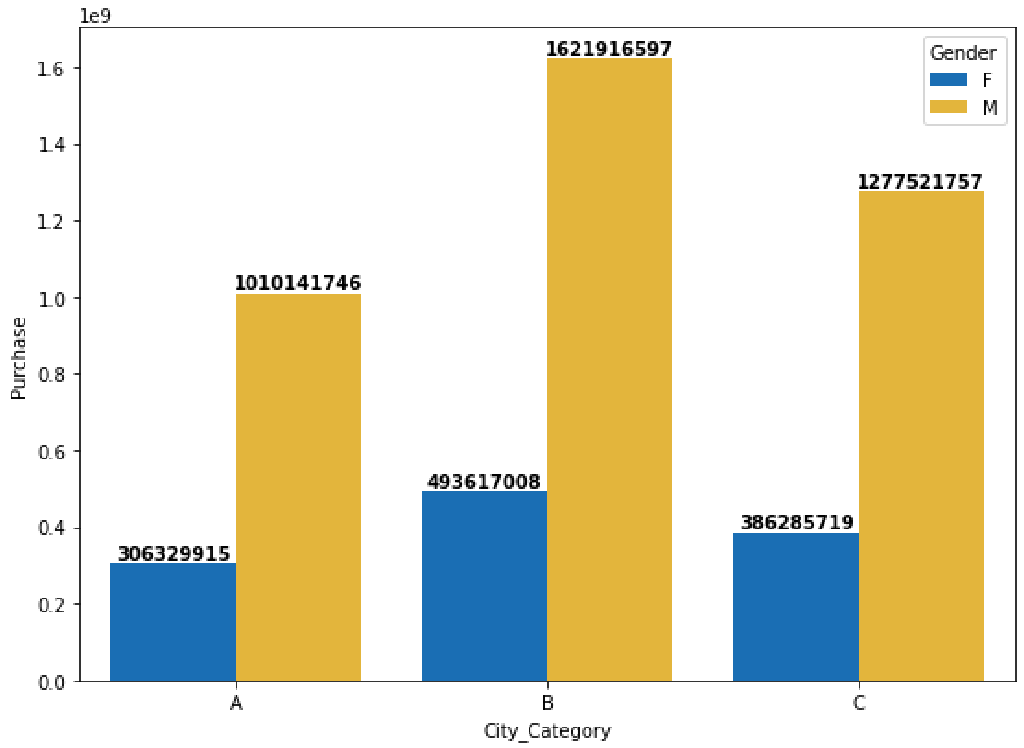

Product Category wise Gender Count
Female
Male
count
108972
41961
115547
24831
80367
33558
19548
4739
18206
5658
15907
4559
14207
6006
8114
3639
7426
2402
5244
1046
4087
1462
3963
1162
2415
1532
2778
943
2743
382
1827
723
1152
451
900
623
516
62
340
70

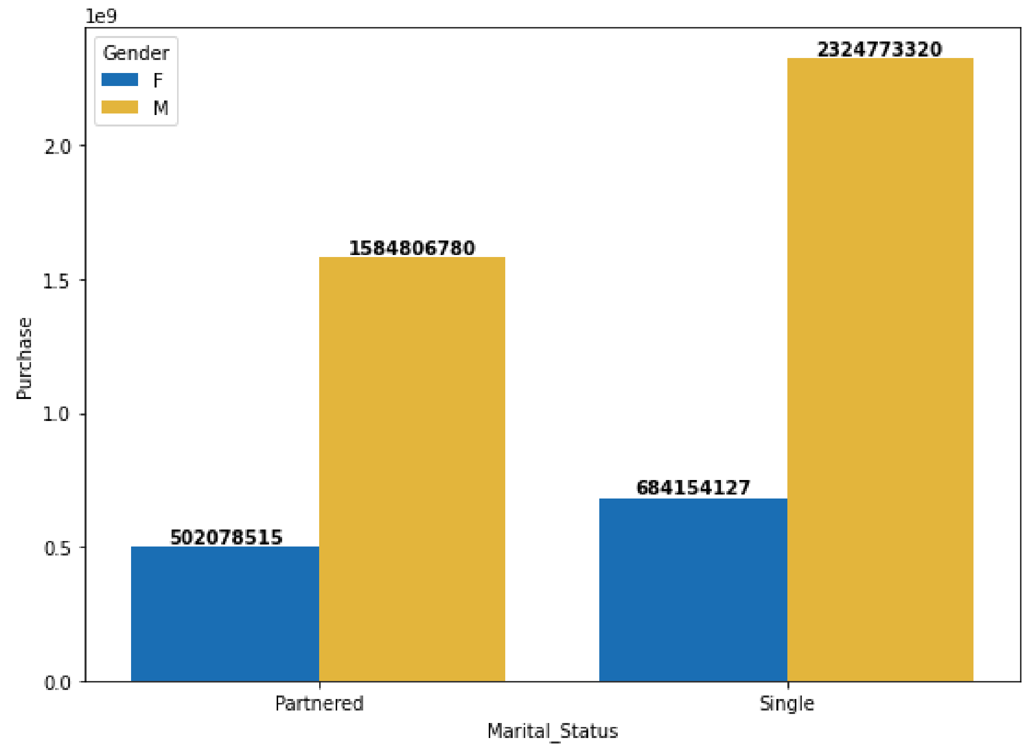
Comment:

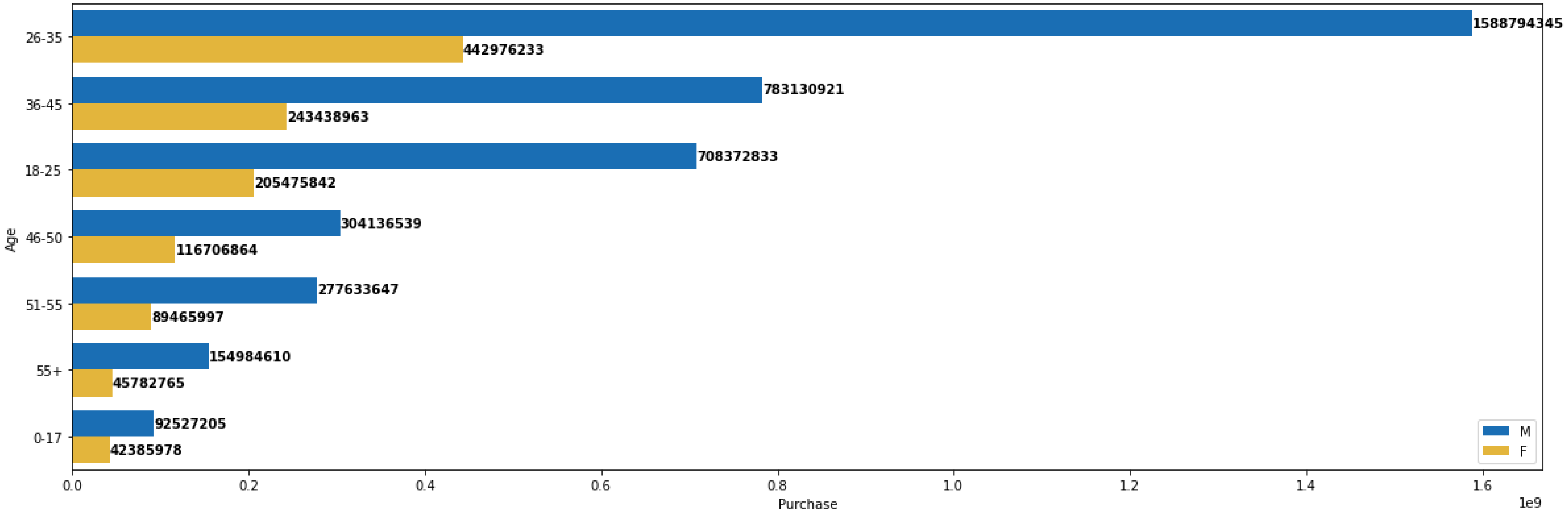
- **Both Male and Female from city Category B have made the higher Purchases, when compared with other City_Category A, C.**
- **Both Male and Female whose Marital_Status is Single have made the highest number of purchases compared to Partnered customers.**
- **Both Male and Femlae who fall under the age of 26-35 have made higher number of purchases.**
- **While Male have made purchases of products from Product_Category 1, Females have made most number of their purchases from Product_Category 5.**

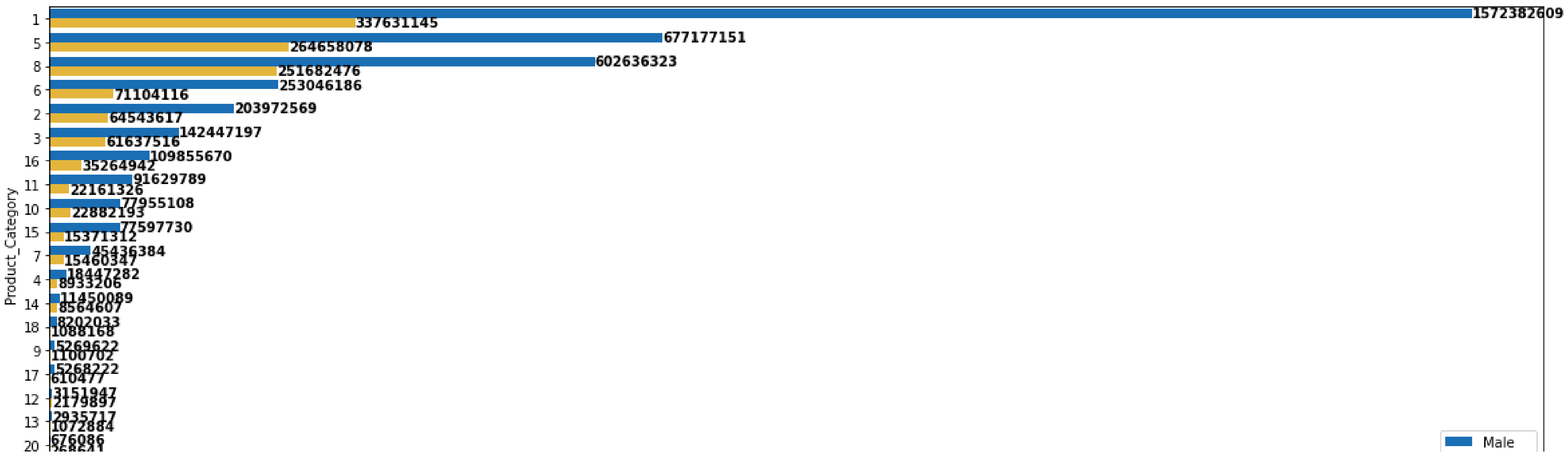
In [52]:

```
plt.figure(figsize = (20,30))

total_purchase_cc_gender = pd.DataFrame(walmart_data.groupby(['City_Category','Gender'])['Purchase'].sum()).reset_index()
total_purchase_ms_gender = pd.DataFrame(walmart_data.groupby(['Marital_Status','Gender'])['Purchase'].sum()).reset_index()
total_purchase_age_gender = pd.DataFrame(walmart_data.groupby(['Age','Gender'])['Purchase'].sum()).reset_index()
total_purchase_pc_gender = pd.DataFrame(walmart_data.groupby(['Product_Category','Gender'])['Purchase'].sum()).reset_index()

plt.subplot(4,2,1)
ax = sns.barplot(data = total_purchase_cc_gender,
                 x = 'City_Category',
                 y = 'Purchase',
                 hue = 'Gender',
                 palette = [walmart_blue, walmart_orange],
                 errorbar = None)
for sum in ax.containers:
  ax.bar_label(sum, fmt = '%.0f', fontsize = 10, fontweight = 'semibold')

plt.subplot(4,2,2)
ax = sns.barplot(data = total_purchase_ms_gender,
                 x = 'Marital_Status',
                 y = 'Purchase',
                 hue = 'Gender',
                 palette = [walmart_blue, walmart_orange],
                 errorbar = None)
for sum in ax.containers:
  ax.bar_label(sum, fmt = '%.0f', fontsize = 10, fontweight = 'semibold')

plt.subplot(4,1,2)
ax = sns.barplot(data = total_purchase_age_gender,
                 y = 'Age',
                 x = 'Purchase',
                 hue = 'Gender',
                 order = walmart_data.groupby('Age')['Purchase'].sum().sort_values(ascending = False).index,
                 hue_order = walmart_data.groupby('Gender')['Purchase'].sum().sort_index(ascending = False).index,
                 palette = [walmart_blue, walmart_orange],
                 errorbar = None,
                 orient = 'h')
for sum in ax.containers:
  ax.bar_label(sum, fmt = '%.0f', fontsize = 10, fontweight = 'semibold')
plt.legend(loc = 'lower right')

plt.subplot(4,1,3)
ax = sns.barplot(data = total_purchase_pc_gender,
                 y = 'Product_Category',
                 x = 'Purchase',
                 order = total_purchase_pc_gender.groupby('Product_Category')['Purchase'].sum().sort_values(ascending = False).index,
                 hue = 'Gender',
                 hue_order = ['M','F'],
                 palette = [walmart_blue, walmart_orange],
                 errorbar = None,
                 orient = 'h')
for sum in ax.containers:
  ax.bar_label(sum, fmt = '%.0f', fontsize = 10, fontweight = 'semibold')
plt.legend(['Male', 'Female'], loc = 'lower right')


plt.show()
```


1e9
Gender
F
M
Purchase
City_Category
A
B
C
306329915
1010141746
493617008
1621916597
386285719
1277521757


1e9
Gender
F
M
Purchase
Marital_Status
Partnered
Single
502078515
1584806780
684154127
2324773320


Age
Purchase
1e9
M
F
26-35
1588794345
442976233
36-45
783130921
243438963
18-25
708372833
205475842
46-50
304136539
116706864
51-55
277633647
89465997
55+
154984610
45782765
0-17
92527205
42385978


Product_Category
Male
1
1572382609
337631145
5
677177151
264658078
8
602636323
251682476
6
253046186
71104116
2
203972569
64543617
3
142447197
61637516
16
109855670
35264942
11
91629789
22161326
10
77955108
22882193
15
77597730
15371312
7
45436384
15460347
4
18447282
8933206
14
11450089
8564607
18
8202033
1088168
9
5269622
1100702
17
5268222
610477
12
3151947
2179897
13
2935717
1072884
20
676086

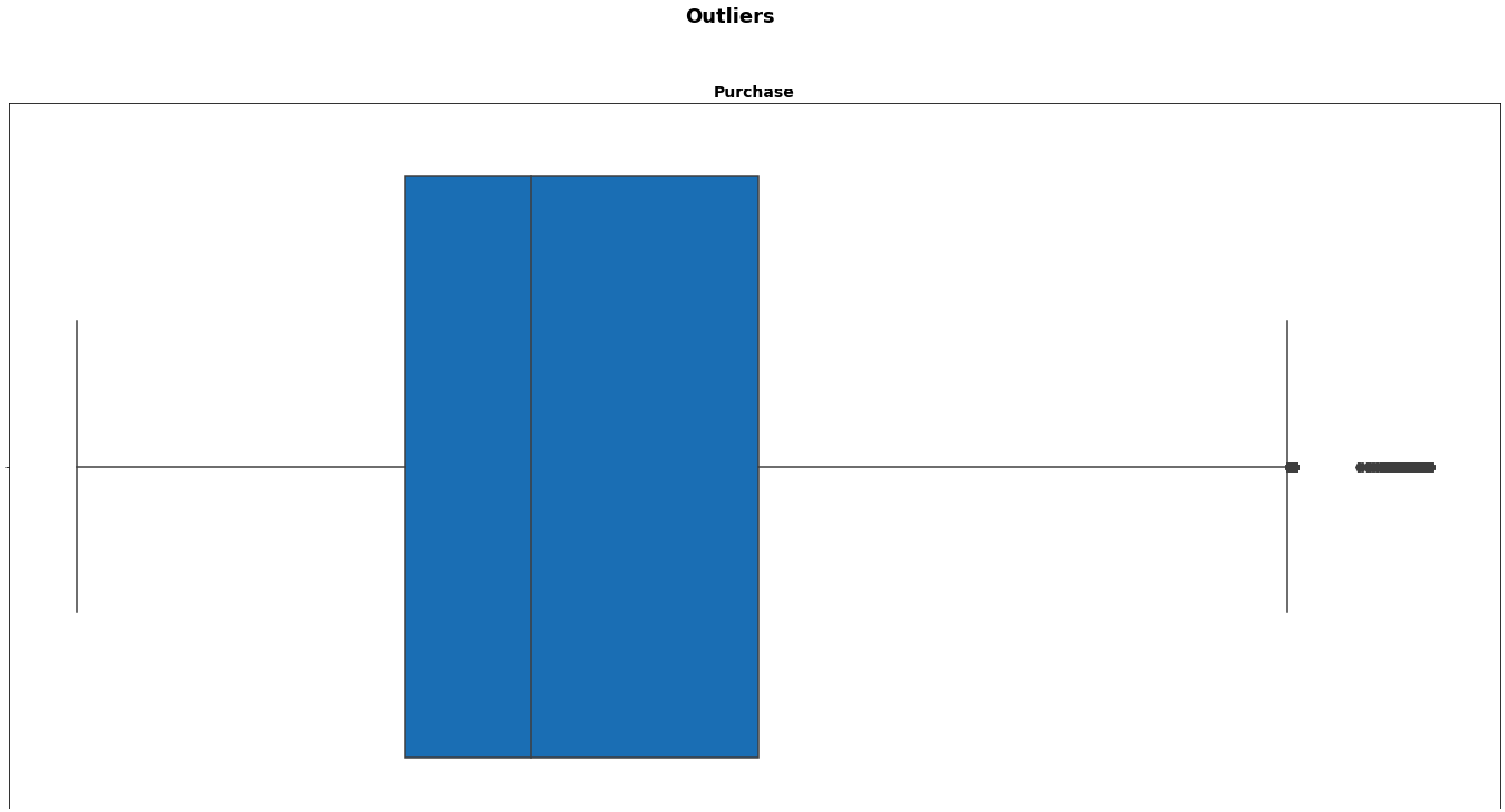
Comment:

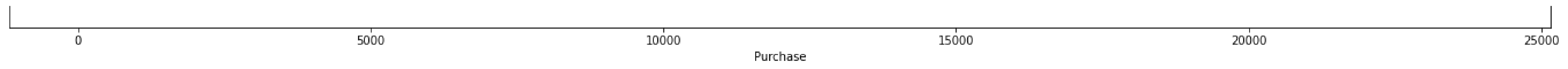
- **Both Male and Female have made the highest total amount of purchasing from City_Category B.**
- **Both Male and Female whose Marital_Status is Single have mde the highest total amount of purchasing.**
- **Both Male and Female who fall under the Age of 26 - 35 have made the highest total amount of purchasing.**
- **Unlike the number of purchasing as we saw above, when we calculate by the total amount spent by Product Category, we can see both Male and Female have spent the highest on Product Cateogry 1.**

Boxplot - Outliers

In [56]:

```
plt.figure(figsize = (24,12))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

sns.boxplot (data = walmart_data, x = 'Purchase', orient = 'h', color = walmart_blue)
plt.title('Purchase', fontweight = 'semibold', fontsize = 14)

plt.suptitle('Outliers', fontweight = 'bold', fontsize = 18)
plt.show()
```





In [58]:

```
walmart_data[["Purchase"]].describe().round(2).T
```

Out[58]:

	count	mean	std	min	25%	50%	75%	max
Purchase	550068.0	9263.97	5023.07	12.0	5823.0	8047.0	12054.0	23961.0

Comment:

- **Looking at the boxplot above we can define that most of the outliers are those of customers who have made purchases above 20000.**
- **The average amount with which a customer is making a purchase is 8047.**
- **The major amount of purchases by all customers fall between 5832 and 12054.**

In [44]:

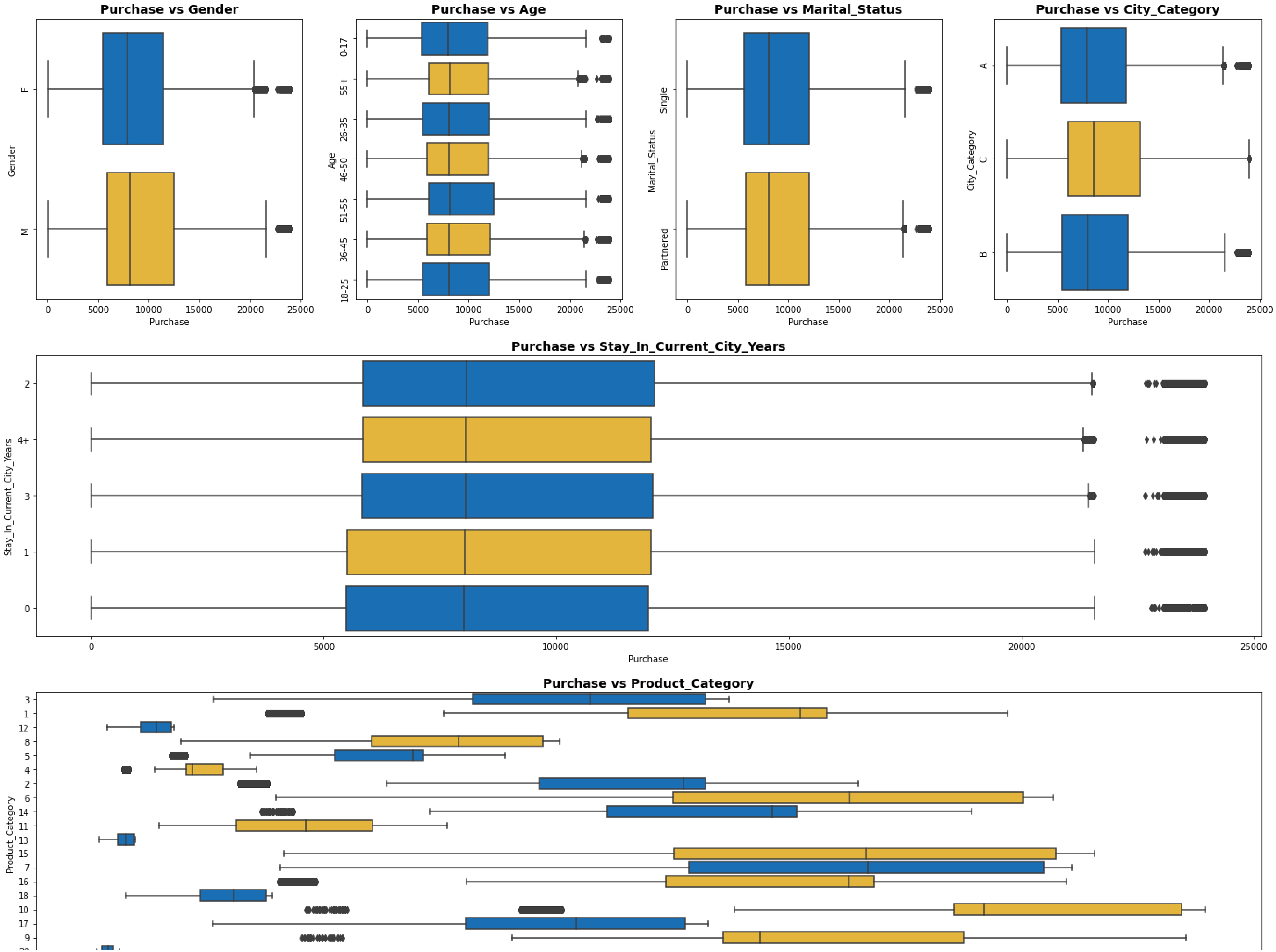
```
walmart_data.info()
```

```
<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
```

In [65]:

```
plt.figure(figsize = (25,20))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

object_columns = ['Gender', 'Age', 'Marital_Status', 'City_Category', 'Stay_In_Current_City_Years', 'Product_Category']
index = 1
idx = 2
for column in object_columns:
  if index >= len(object_columns) - 1:
    plt.subplot(3,1, idx)
    sns.boxplot(data = walmart_data, x = 'Purchase', y = column, orient = 'h', palette = [walmart_blue, walmart_orange])
    plt.title (f'Purchase vs {column}', fontsize = 14, fontweight ='semibold')
    idx += 1
  else:
    plt.subplot(3,4,index)
    sns.boxplot(data = walmart_data, x = 'Purchase', y = column, orient = 'h', palette = [walmart_blue, walmart_orange])
    plt.yticks(rotation = 90)
    plt.xlabel('Purchase', fontweight = 'medium')
    plt.ylabel(column, fontweight = 'medium')
    plt.title (f'Purchase vs {column}', fontsize = 14, fontweight ='semibold')
  index += 1

plt.suptitle('Bi-Varient Outlier Identifiers', fontsize = 18, fontweight = 'bold')
plt.show()
```

Bi-varient Outlier Identifiers



In [99]:

```
columns = ['Gender', 'Age', 'Marital_Status', 'City_Category', 'Product_Category']
df = pd.DataFrame()
for i in columns:
  for j in np.sort(walmart_data[i].unique()):
    df[f'{i} = {j}'] = walmart_data[walmart_data[i] == j][["Purchase"]].describe().round(2)
df.T
```

Out[99]:

	count	mean	std	min	25%	50%	75%	max
Gender = F	135809.0	8734.57	4767.23	12.0	5433.00	7914.0	11400.00	23959.0
Gender = M	414259.0	9437.53	5092.19	12.0	5863.00	8098.0	12454.00	23961.0
Age = 0-17	15102.0	8933.46	5111.11	12.0	5328.00	7986.0	11874.00	23955.0
Age = 18-25	99660.0	9169.66	5034.32	12.0	5415.00	8027.0	12028.00	23958.0
Age = 26-35	219587.0	9252.69	5010.53	12.0	5475.00	8030.0	12047.00	23961.0
Age = 36-45	110013.0	9331.35	5022.92	12.0	5876.00	8061.0	12107.00	23960.0
Age = 46-50	45701.0	9208.63	4967.22	12.0	5888.00	8036.0	11997.00	23960.0
Age = 51-55	38501.0	9534.81	5087.37	12.0	6017.00	8130.0	12462.00	23960.0
Age = 55+	21504.0	9336.28	5011.49	12.0	6018.00	8105.5	11932.00	23960.0
Marital_Status = Partnered	225337.0	9261.17	5016.90	12.0	5843.00	8051.0	12042.00	23961.0
Marital_Status = Single	324731.0	9265.91	5027.35	12.0	5605.00	8044.0	12061.00	23961.0
City_Category = A	147720.0	8911.94	4892.12	12.0	5403.00	7931.0	11786.00	23961.0
City_Category = B	231173.0	9151.30	4955.50	12.0	5460.00	8005.0	11986.00	23960.0
City_Category = C	171175.0	9719.92	5189.47	12.0	6031.50	8585.0	13197.00	23961.0
Product_Category = 1	140378.0	13606.22	4298.83	3790.0	11546.00	15245.0	15812.00	19708.0
Product_Category = 10	5125.0	19675.57	4225.72	4624.0	18546.00	19197.0	23438.00	23961.0
Product_Category = 11	24287.0	4685.27	1834.90	1472.0	3131.00	4611.0	6058.00	7654.0
Product_Category = 12	3947.0	1350.86	362.51	342.0	1071.00	1401.0	1723.00	1778.0
Product_Category = 13	5549.0	722.40	183.49	185.0	578.00	755.0	927.00	962.0
Product_Category = 14	1523.0	13141.63	4069.01	3657.0	11097.00	14654.0	15176.50	18931.0
Product_Category = 15	6290.0	14780.45	5175.47	4148.0	12523.25	16660.0	20745.75	21569.0
Product_Category = 16	9828.0	14766.04	4360.21	4036.0	12354.00	16292.5	16831.00	20971.0
Product_Category = 17	578.0	10170.76	2333.99	2616.0	8063.50	10435.5	12776.75	13264.0
Product_Category = 18	3125.0	2972.86	727.05	754.0	2359.00	3071.0	3769.00	3900.0
Product_Category = 19	1603.0	37.04	16.87	12.0	24.00	37.0	50.00	62.0
Product_Category = 2	23864.0	11251.94	3570.64	3176.0	9645.75	12728.5	13212.00	16504.0
Product_Category = 20	2550.0	370.48	167.12	118.0	242.00	368.0	490.00	613.0
Product_Category = 3	20213.0	10096.71	2824.63	2638.0	8198.00	10742.0	13211.00	13717.0
Product_Category = 4	11753.0	2329.66	812.54	684.0	2058.00	2175.0	2837.00	3556.0
Product_Category = 5	150933.0	6240.09	1909.09	1713.0	5242.00	6912.0	7156.00	8907.0
Product_Category = 6	20466.0	15838.48	4011.23	3981.0	12505.00	16312.0	20051.00	20690.0
Product_Category = 7	3721.0	16365.69	4174.55	4061.0	12848.00	16700.0	20486.00	21080.0
Product_Category = 8	113925.0	7498.96	2013.02	1939.0	6036.00	7905.0	9722.00	10082.0
Product_Category = 9	410.0	15537.38	5330.85	4528.0	13583.50	14388.5	18764.00	23531.0

Comment:

- **Gender vs Purchase:**
  - ***Gender Male:*** **Most of the outliers fall for the product where customer has made a purchase for more than 20000. Most purchase by Male customer was done between the amount of 5433 and 11400.**
  - ***Gender Female:*** **Most of the outliers fall for the product where customer has made a purchase for more than 20000. Most purchase by Male customer was done between the amount of 5863 and 12454.**
  - ***Male vs Female: Though we have observed more number of purchases done by Male customers we can see most Female customers tend to spend a little more amount than Male customers.***

- **Age vs Purchase:**

  Outlier amounts of purchases for all age groups fall above 20000.
  - ***Age 0-17:*** **Most amount spent on Purchasing a product is in between 5328 and 11874.**
  - ***Age 18-25:*** **Most amount spent on Purchasing a product is in between 5415 and 12028.**
  - ***Age 26-35:*** **Most amount spent on Purchasing a product is in between 5475 and 12047.**
  - ***Age 36-45:*** **Most amount spent on Purchasing a product is in between 5876 and 12107.**
  - ***Age 46-50:*** **Most amount spent on Purchasing a product is in between 6017 and 12462.**
  - ***Age 51-55:*** **Most amount spent on Purchasing a product is in between 5328 and 11874.**
  - ***Age 55+:*** **Most amount spent on Purchasing a product is in between 6018 and 11932.**
  - ***Comparion of All Ages: Though the highest number of purchases are done by customers who fall under the age 26-35, we can see the highest range of amount spent is by customer who fall under the age 51 - 55.***

- **Marital Status vs Purchase:**

  Outlier amounts of purchases for all age groups fall above 20000.
  - ***Single:*** **Most amount spent on Purchasing a product is in between 5605 and 12061.**
  - ***Partnered:*** **Most amount spent on Purchasing a product is in between 5843 and 12042.**
  - ***Single vs Partnered: Here the amount spent to make a purchase between single and partnered customers is almost same but may be we can observe singles tend to spend a little extra.***

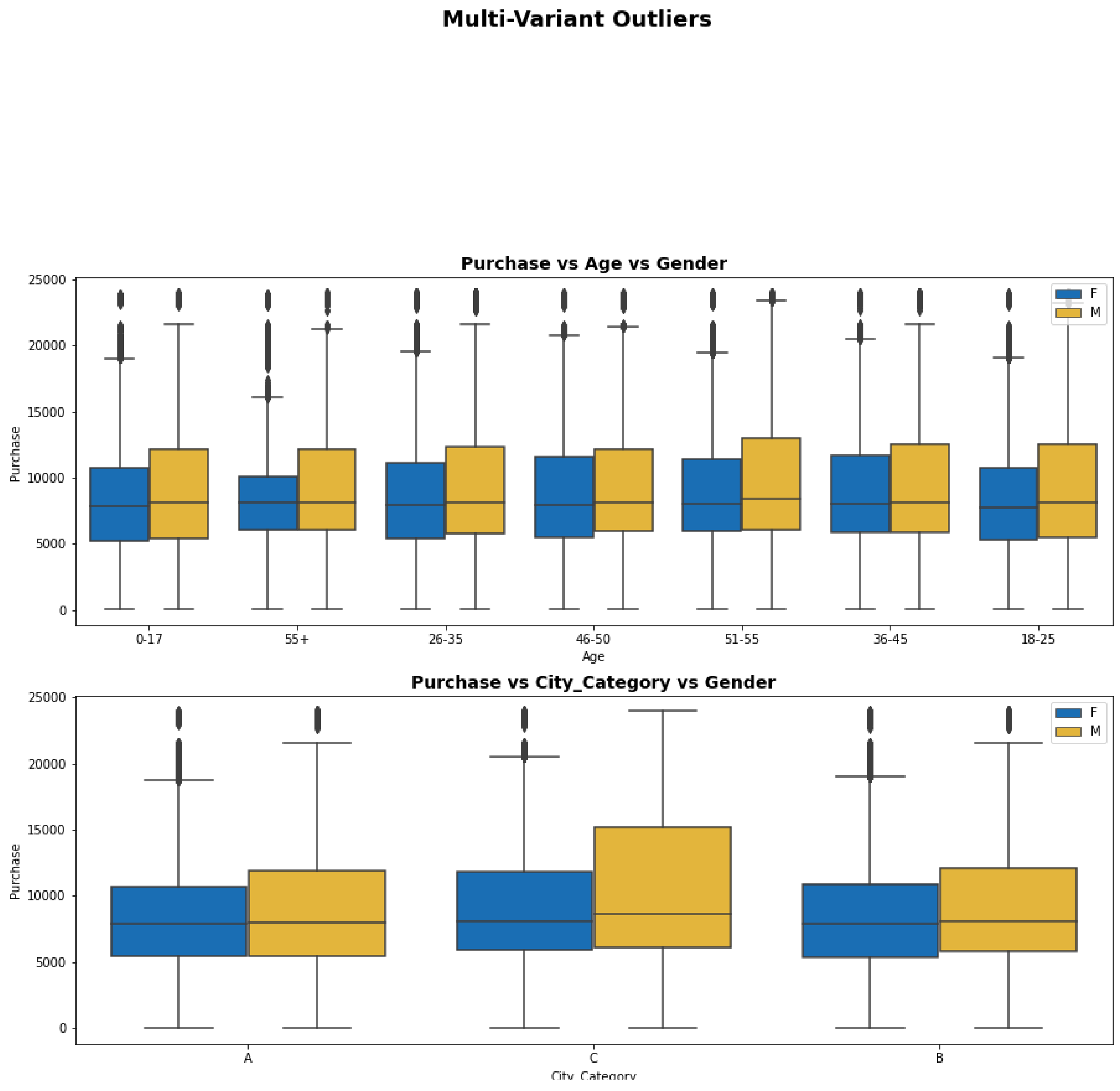
- **City Category vs Purchase:**

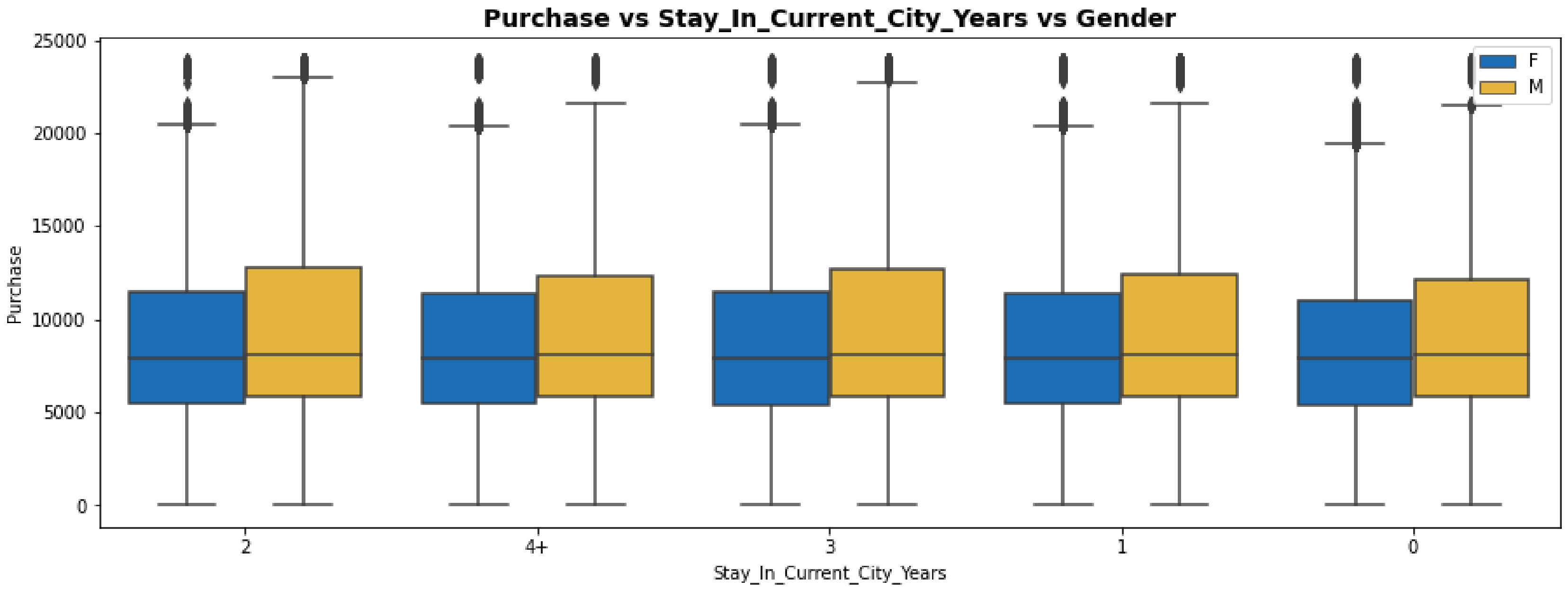
  Outlier amounts of purchases for all City Categories fall above 21000.
  - ***City Category A:*** **Most amount spent on Purchasing a product is in between 5403 and 11786.**
  - ***City Category B:*** **Most amount spent on Purchasing a product is in between 5460 and 11986.**
  - ***City Category C:*** **Most amount spent on Purchasing a product is in between 6031 and 13197.**
  - ***Comparision of All City Category: Though the highest number of purchases are done by City Category B, we can observe the higher amount is being spent by City Category C.***

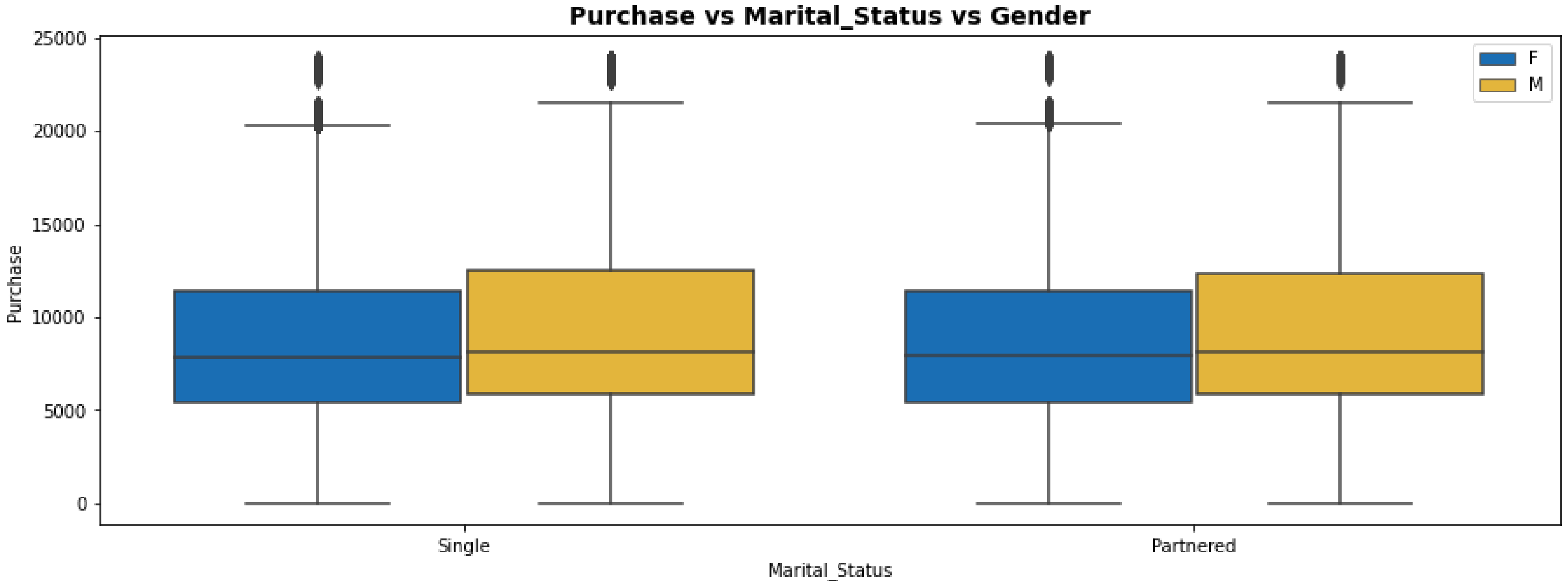
- **Product_Category vs Purchase:**
  - **Product Category 1: While a total of 140378 purchase have been made on this product category, the average purchase amount is 13606.22. We can observe most customers spend between the amounts of 11546 and 15812.**
  - **Product Category 2: While a total of 23864 purchase have been made on this product category, the average purchase amount is 11251.94. We can observe most customers spend between the amounts of 9645.75 and 13212.**
  - **Product Category 3: While a total of 20213 purchase have been made on this product category, the average purchase amount is 10096.71. We can observe most customers spend between the amounts of 8198 and 13211.**
  - **Product Category 4: While a total of 11753 purchase have been made on this product category, the average purchase amount is 2329.66. We can observe most customers spend between the amounts of 2058 and 2837.**
  - **Product Category 5: While a total of 150933 purchase have been made on this product category, the average purchase amount is 6240.09. We can observe most customers spend between the amounts of 5242 and 7156.**
  - **Product Category 6: While a total of 20466 purchase have been made on this product category, the average purchase amount is 15838.48. We can observe most customers spend between the amounts of 12505 and 20051.**
  - **Product Category 7: While a total of 3721 purchase have been made on this product category, the average purchase amount is 16365.69. We can observe most customers spend between the amounts of 12848 and 20486.**
  - **Product Category 8: While a total of 113925 purchase have been made on this product category, the average purchase amount is 7498.96. We can observe most customers spend between the amounts of 6036 and 9722.**
  - **Product Category 9: While a total of 410 purchase have been made on this product category, the average purchase amount is 15537.38. We can observe most customers spend between the amounts of 13583.50 and 18764.**
  - **Product Category 10: While a total of 5125 purchase have been made on this product category, the average purchase amount is 19675.57. We can observe most customers spend between the amounts of 18546 and 23438.**
  - **Product Category 11: While a total of 27287 purchase have been made on this product category, the average purchase amount is 4685.27. We can observe most customers spend between the amounts of 3131 and 6058.**
  - **Product Category 12: While a total of 3947 purchase have been made on this product category, the average purchase amount is 1350.86. We can observe most customers spend between the amounts of 1071 and 1723.**
  - **Product Category 13: While a total of 5549 purchase have been made on this product category, the average purchase amount is 722.40. We can observe most customers spend between the amounts of 578 and 927.**
  - **Product Category 14: While a total of 1523 purchase have been made on this product category, the average purchase amount is 13141.63. We can observe most customers spend between the amounts of 11097 and 15176.50.**
  - **Product Category 15: While a total of 6290 purchase have been made on this product category, the average purchase amount is 14780.45. We can observe most customers spend between the amounts of 12523.25 and 20754.75.**
  - **Product Category 16: While a total of 9828 purchase have been made on this product category, the average purchase amount is 14766.04. We can observe most customers spend between the amounts of 12354 and 16831.**
  - **Product Category 17: While a total of 578 purchase have been made on this product category, the average purchase amount is 10170.76. We can observe most customers spend between the amounts of 8063.50 and 12776.75.**
  - **Product Category 18: While a total of 3125 purchase have been made on this product category, the average purchase amount is 2972.86. We can observe most customers spend between the amounts of 2359 and 3769.**
  - **Product Category 19: While a total of 1603 purchase have been made on this product category, the average purchase amount is 37.04. We can observe most customers spend between the amounts of 24 and 50.**
  - **Product Category 20: While a total of 2550 purchase have been made on this product category, the average purchase amount is 340.48. We can observe most customers spend between the amounts of 242 and 490.**
  - ***Comparing all Product Categories: Top 3 selling product categories are 5,1,8. But if we observe the products sold under the category 1 seem to be making more money as the average price of the product is 13606.22. Also the least purchased products are from product categories 9, 17, 19, 20.***

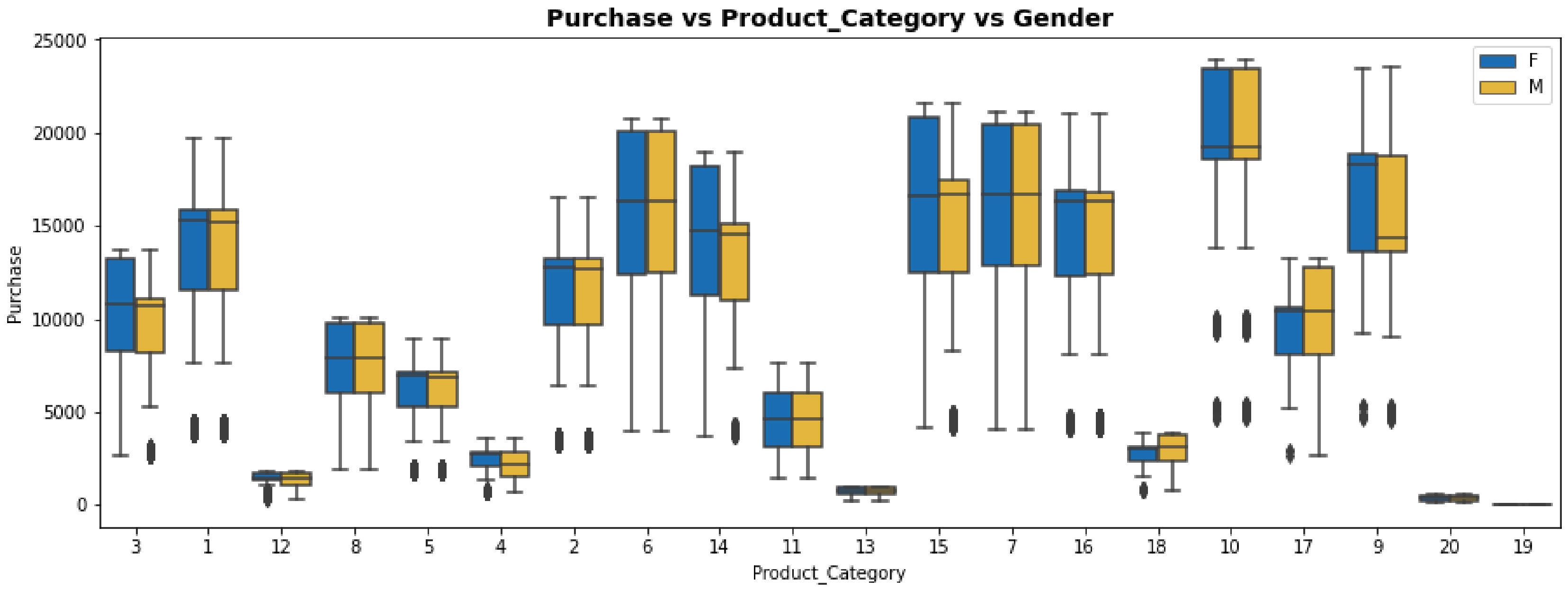
In [101]:

```python
plt.figure(figsize = (15,30))
walmart_blue = "#0071ce"
walmart_orange = "#ffc220"

multi_columns = ['Age', 'City_Category', 'Stay_In_Current_City_Years','Marital_Status', 'Product_Category']
index = 1
for column in multi_columns:
  plt.subplot(len(multi_columns), 1, index)
  sns.boxplot (data = walmart_data, x = column, y = 'Purchase', hue = 'Gender', palette = [walmart_blue, walmart_orange])
  plt.title (f'Purchase vs {column} vs Gender', fontsize = 14, fontweight = 'semibold')
  plt.legend (loc = 'upper right')
  index += 1

plt.suptitle('Multi-Variant Outliers', fontsize = 18, fontweight = 'bold')
plt.show()
```









In [106]:

```
dfmv = pd.DataFrame()
columns = ["Marital_Status", "City_Category", "Age", "Stay_In_Current_City_Years", "Product_Category"]
for i in columns:
  for j in np.sort(walmart_data[i].unique()):
    for k in walmart_data["Gender"].unique():
      dfmv[f'{k} & {j}'] = walmart_data[(walmart_data[i] == j) & (walmart_data["Gender"] == k)][["Purchase"]].describe().round(2)
dfmv.T
```

Out[106]:

	count	mean	std	min	25%	50%	75%	max
F & Partnered	56988.0	8810.25	4803.59	12.0	5456.75	7939.0	11451.00	23959.0
M & Partnered	168349.0	9413.82	5078.03	12.0	5874.00	8094.0	12312.00	23961.0
F & Single	78821.0	8679.85	4740.05	12.0	5417.00	7895.0	11370.00	23955.0
M & Single	245910.0	9453.76	5101.80	12.0	5854.00	8101.0	12543.00	23961.0
F & A	35704.0	8579.71	4670.23	12.0	5413.00	7847.0	10728.25	23948.0
...	...	...	...	...	...	...	...	...
M & 7	2778.0	16355.79	4187.57	4061.0	12843.00	16710.5	20486.00	21080.0
F & 8	33558.0	7499.92	2014.90	1939.0	6037.00	7907.0	9720.00	10082.0
M & 8	80367.0	7498.55	2012.24	1939.0	6036.00	7905.0	9722.00	10082.0
F & 9	70.0	15724.31	5646.63	4633.0	13581.25	18256.0	18836.25	23481.0
M & 9	340.0	15498.89	5271.38	4528.0	13589.50	14361.0	18749.75	23531.0

68 rows × 8 columns

Comment:

- **Gender vs Marital Status vs Purchase:**

> **We can by the box plot that Male customers who are Partnered or Single are spending almost similar range amounts on theor purchases and we can find a similar trend with Female customers.**

- **Gender vs City Category vs Purchase:**

> **We can observe that Male and Female have a higher spending range who hail from City Category C compared to the other two City Categories A, B.**

- **Gender vs Age vs Purchase:**

> **Male of all age groups have a similar range of amount spend while buying a product. Female also have slightly similar but we can a small varience in age groups of 51 - 55 and 55+.**

- **Gender vs Product Category vs Purchase:**

> - **We can see a trend that most Male don't mind spending more amount on products of all product categories except for 4, 18, 12, 13, 19, 20 where the total puchases looks very low.**
> - **We can see a similar trend with Female as well just as male customers.**

Heatmap and Pairplot

In [107]:

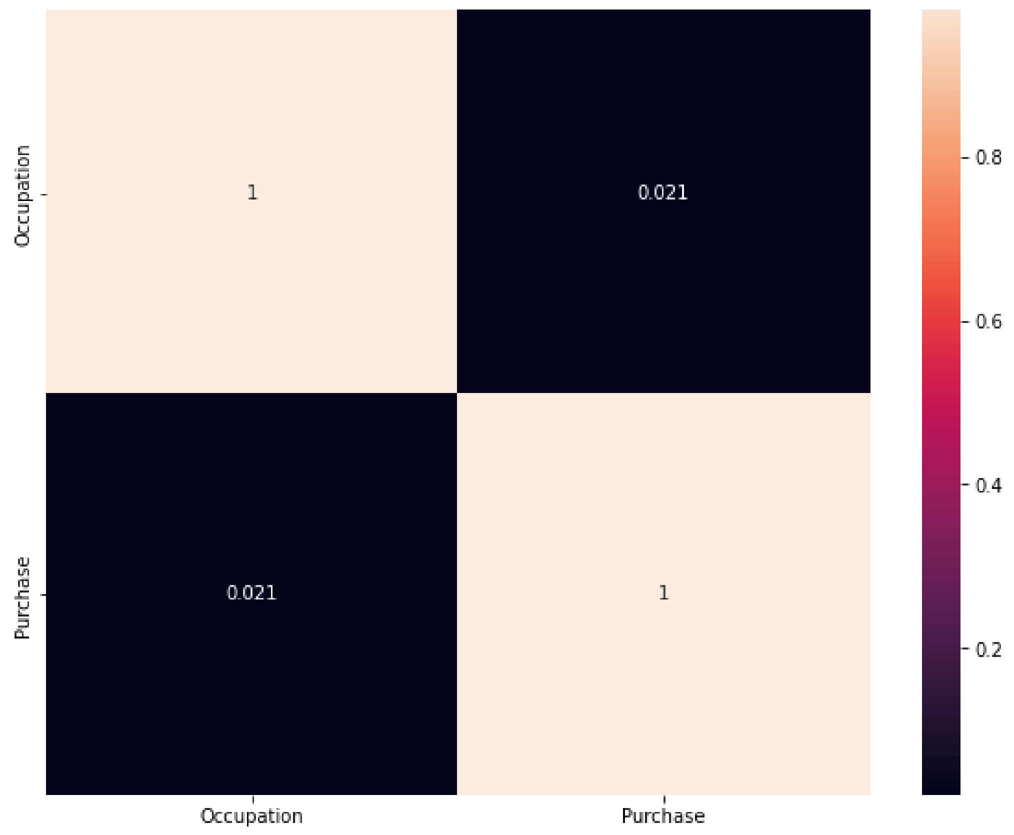
```
walmart_data.corr()
```

Out[107]:

	Occupation	Purchase
Occupation	1.000000	0.020833
Purchase	0.020833	1.000000

In [108]:

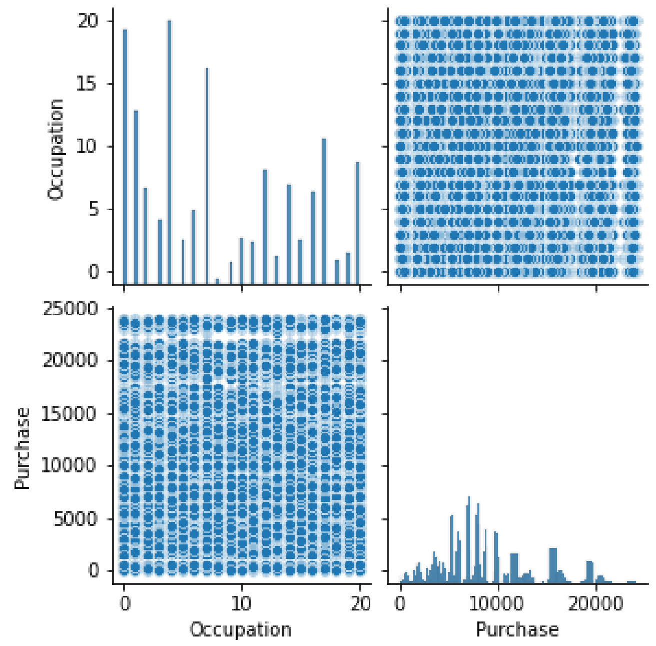
```
plt.figure(figsize = (10,8))
sns.heatmap(walmart_data.corr(), annot = True)
plt.show()
```



In [109]:

```
plt.figure(figsize = (10,8))
sns.pairplot(walmart_data)
plt.show()
```

<Figure size 720x576 with 0 Axes>



6. Confidence Intervals & Central LImit Theorem

In [33]:

```
walmart_data.head()
```

Out[33]:

	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

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
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

In [111]:

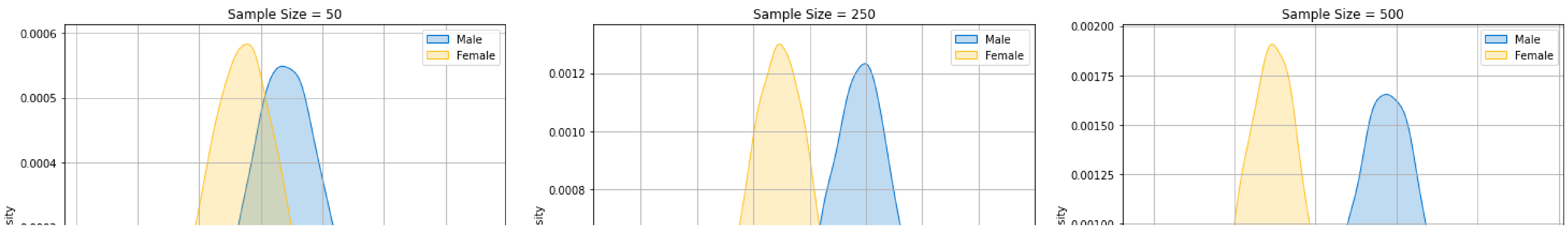
```
def sample_means (size, r, data):
  purchase_mean = np.empty(r)
  for i in range(r):
    sample = np.random.choice(data, size = size)
    purchase_mean[i] = np.mean(sample)
  return purchase_mean
```

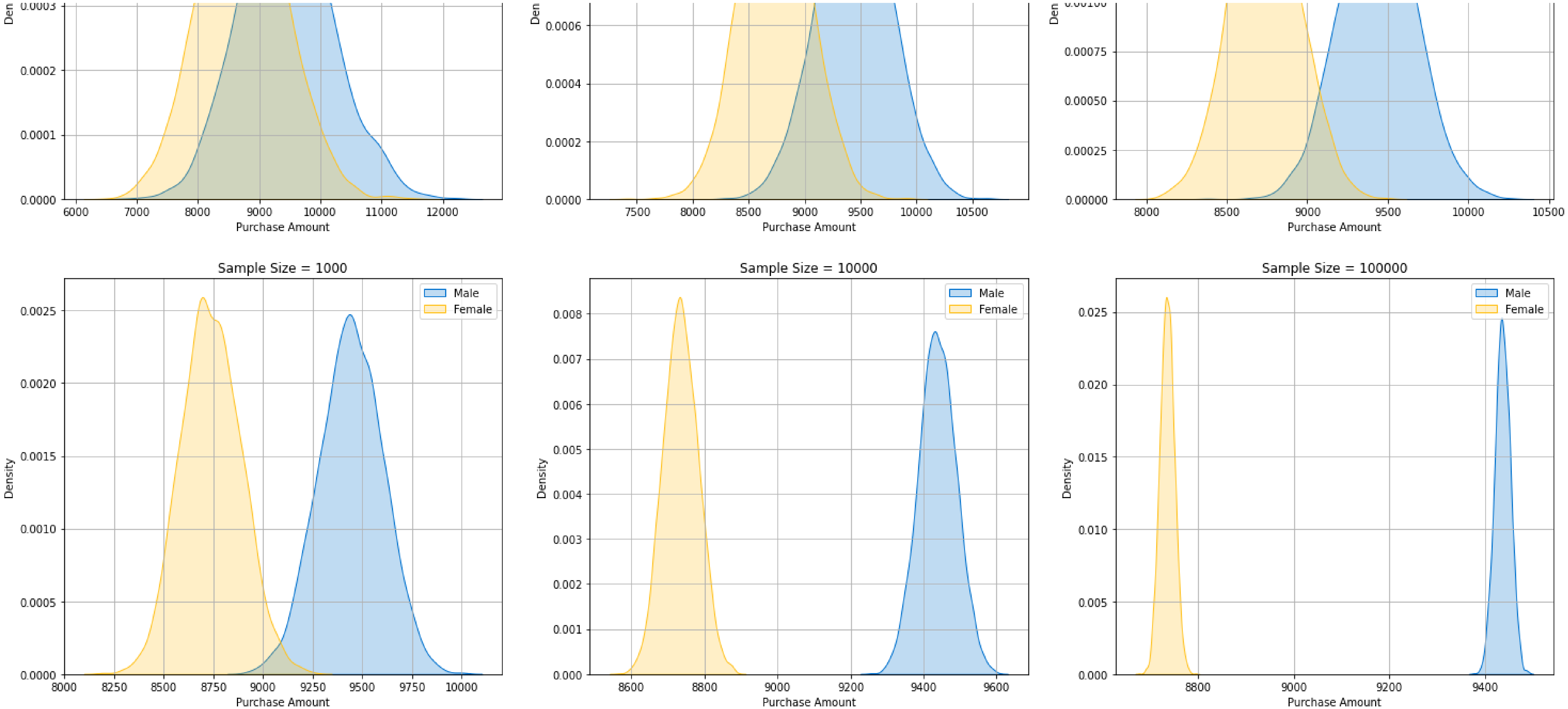
Gender (Male vs Female) Wise Purchase Distribution

In [123]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 5000
index = 1
male_ms_5000 = {'Mean':{} ,'Standard_Error':{}}
female_ms_5000 = {'Mean':{} ,'Standard_Error':{}}
male_ci_5000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
female_ci_5000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  male_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "M"]["Purchase"])
  female_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "F"]["Purchase"])
  male_ms_5000["Mean"][f'Sample Size = {size}'], male_ms_5000["Standard_Error"][f'Sample Size = {size}'] = np.mean(male_purchase_mean).round(2), np.std(male_purchase_mean).round(2)
  female_ms_5000["Mean"][f'Sample Size = {size}'], female_ms_5000["Standard_Error"][f'Sample Size = {size}'] = np.mean(female_purchase_mean).round(2), np.std(female_purchase_mean).round(2
)
  male_ci_5000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (1.96 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (1.96 * np
.std(male_purchase_mean))).round(2))
  female_ci_5000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (1.96 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (1
.96 * np.std(female_purchase_mean))).round(2))
  male_ci_5000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (2.58 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (2.58 * np
.std(male_purchase_mean))).round(2))
  female_ci_5000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (2.58 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (2
.58 * np.std(female_purchase_mean))).round(2))
  sns.kdeplot(male_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(female_purchase_mean, fill = True, color = walmart_orange)
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["Male", "Female"])
  plt.grid()
  index += 1
plt.suptitle('Number of Samples = 5000 & various Sample Sizes to check\nPurchases Distribution among Male vs Female', fontsize = 18, fontweight = 'bold')
plt.show()
```

Number of Samples = 5000 & various Sample Sizes to check
Purchases Distribution among Male vs Female





In [124]:

```
male_ms_5000 = pd.DataFrame(male_ms_5000)
male_ci_5000 = pd.DataFrame(male_ci_5000)
print (f'Total Samples = 5000 & Various Sample Sizes\nMean & Standard Error for Male Customers\n\n{male_ms_5000}')
print ("-" * 50, "\n")
print (f'Total Samples = 5000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{male_ci_5000}')
```

```
Total Samples = 5000 & Various Sample Sizes
Mean & Standard Error for Male Customers

                          Mean  Standard_Error
Sample Size = 50       9433.26          733.57
Sample Size = 250      9442.14          322.88
Sample Size = 500      9437.24          233.95
Sample Size = 1000     9442.70          162.02
Sample Size = 10000    9437.47           51.27
Sample Size = 100000   9436.95           16.15
--------------------------------------------------

Total Samples = 5000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7995.46, 10871.06)    (7540.65, 11325.87)
Sample Size = 250       (8809.28, 10074.99)     (8609.1, 10275.18)
Sample Size = 500         (8978.7, 9895.78)    (8833.65, 10040.83)
Sample Size = 1000       (9125.13, 9760.26)     (9024.67, 9860.72)
Sample Size = 10000      (9336.98, 9537.97)     (9305.19, 9569.76)
Sample Size = 100000       (9405.3, 9468.6)     (9395.29, 9478.61)
```

In [125]:

```
female_ms_5000 = pd.DataFrame(female_ms_5000)
female_ci_5000 = pd.DataFrame(female_ci_5000)
print (f'Total Samples = 5000 & Various Sample Sizes\nMean & Standard Error for Female Customers\n\n{female_ms_5000}')
print ("-" * 50, "\n")
print (f'Total Samples = 5000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{female_ci_5000}')
```

```
Total Samples = 5000 & Various Sample Sizes
Mean & Standard Error for Female Customers

                        Mean  Standard_Error
Sample Size = 50      8735.50          682.62
Sample Size = 250     8733.54          300.49
Sample Size = 500     8736.90          210.77
Sample Size = 1000    8738.07          151.36
Sample Size = 10000   8734.29           47.63
Sample Size = 100000  8734.52           14.90
--------------------------------------------------

Total Samples = 5000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7397.56, 10073.43)    (6974.34, 10496.66)
Sample Size = 250        (8144.59, 9322.49)     (7958.29, 9508.79)
Sample Size = 500        (8323.79, 9150.02)      (8193.11, 9280.7)
Sample Size = 1000       (8441.41, 9034.73)     (8347.57, 9128.57)
Sample Size = 10000      (8640.95, 8827.64)     (8611.42, 8857.17)
Sample Size = 100000     (8705.32, 8763.72)     (8696.09, 8772.96)
```

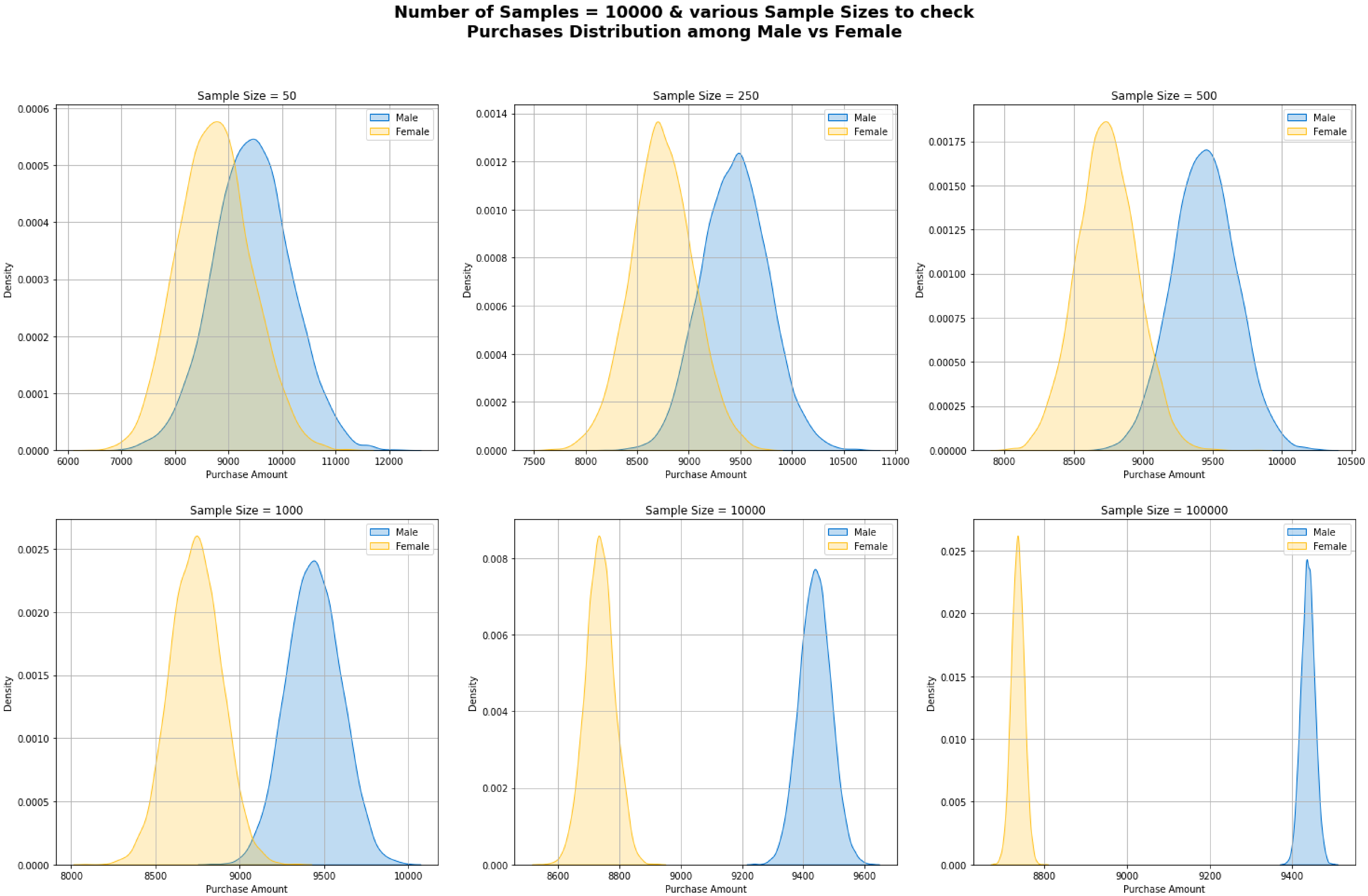
Comment: **Nummber of Samples = 5000 and Various Sample Size**

- **We can observe that when the sample size is as low as 50 which means expecting 50 customers of both Male and Female are making a purchase, then we can see that there is a clear overlap of mean of purchase amount of both male and female.**
- **As the sample size increases we can observe a huge gap between the mean amounts of both male and female, which clearly indicates that the highest amounts of purchases are done by male over female.**
- **We can also observe that the 95% and 99% Confidence Intervals that there is a huge gap in average purchase amount of both male and female and they don't overlap with each other.**

In [130]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 10000
index = 1
male_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
female_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
male_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
female_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  male_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "M"]["Purchase"])
  female_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "F"]["Purchase"])
  male_ms_10000["Mean"][f'Sample Size = {size}'], male_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(male_purchase_mean).round(2), np.std(male_purchase_mean).round(2)
  female_ms_10000["Mean"][f'Sample Size = {size}'], female_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(female_purchase_mean).round(2), np.std(female_purchase_mean).round
(2)
  male_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (1.96 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (1.96 * n
p.std(male_purchase_mean))).round(2))
  female_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (1.96 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (
1.96 * np.std(female_purchase_mean))).round(2))
  male_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (2.58 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (2.58 * n
p.std(male_purchase_mean))).round(2))
  female_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (2.58 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (
2.58 * np.std(female_purchase_mean))).round(2))
  sns.kdeplot(male_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(female_purchase_mean, fill = True, color = walmart_orange)
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["Male", "Female"])
  plt.grid()
  index += 1
```

```
plt.suptitle('Number of Samples = 10000 & various Sample Sizes to check\nPurchases Distribution among Male vs Female', fontsize = 18, fontweight = 'bold')
plt.show()
```



In [131]:

```
male_ms_10000 = pd.DataFrame(male_ms_10000)
male_ci_10000 = pd.DataFrame(male_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for Male Customers\n\n{male_ms_10000}')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{male_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for Male Customers
```

```
                          Mean  Standard_Error
Sample Size = 50       9433.89          718.65
Sample Size = 250      9444.44          320.09
Sample Size = 500      9438.04          228.33
Sample Size = 1000     9436.97          160.53
Sample Size = 10000    9437.89           50.80
Sample Size = 100000   9437.55           16.40
--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (8025.33, 10842.45)    (7579.76, 11288.01)
Sample Size = 250       (8817.06, 10071.83)     (8618.6, 10270.29)
Sample Size = 500        (8990.51, 9885.57)    (8848.94, 10027.13)
Sample Size = 1000        (9122.33, 9751.6)     (9022.81, 9851.13)
Sample Size = 10000      (9338.31, 9537.46)     (9306.81, 9568.96)
Sample Size = 100000     (9405.41, 9469.69)     (9395.24, 9479.86)
```

In [132]:

```
female_ms_10000 = pd.DataFrame(female_ms_10000)
female_ci_10000 = pd.DataFrame(female_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for Female Customers\n\n{female_ms_10000}')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{female_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for Female Customers

                          Mean  Standard_Error
Sample Size = 50       8737.44          668.92
Sample Size = 250      8737.65          300.58
Sample Size = 500      8735.79          213.53
Sample Size = 1000     8735.77          151.47
Sample Size = 10000    8734.82           47.48
Sample Size = 100000   8734.74           15.27
--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7426.36, 10048.52)    (7011.63, 10463.25)
Sample Size = 250        (8148.51, 9326.79)     (7962.15, 9513.15)
Sample Size = 500        (8317.27, 9154.32)     (8184.88, 9286.71)
Sample Size = 1000        (8438.9, 9032.64)     (8344.99, 9126.55)
Sample Size = 10000      (8641.76, 8827.88)     (8612.32, 8857.32)
Sample Size = 100000     (8704.82, 8764.66)     (8695.36, 8774.13)
```

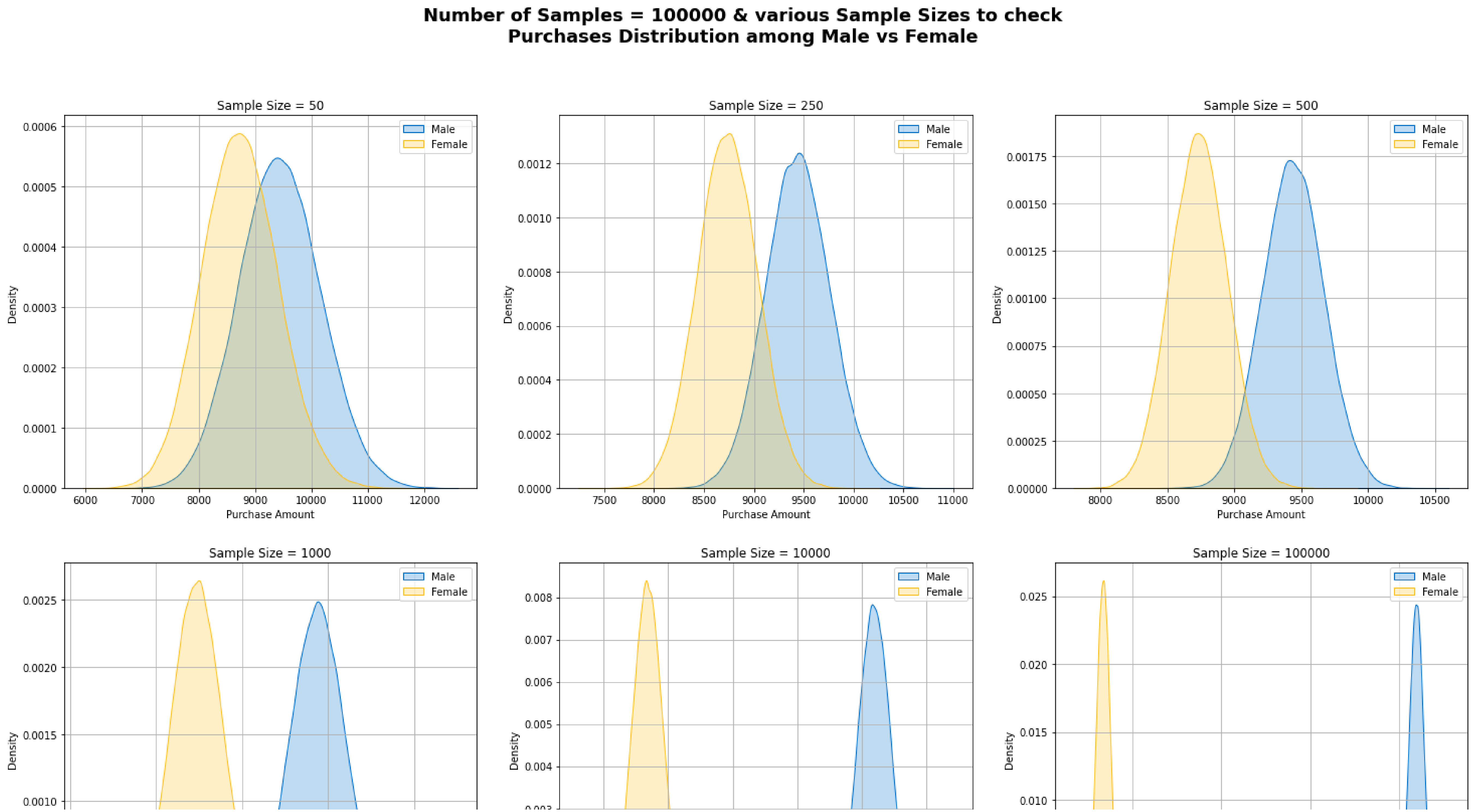
Comment: **Nummber of Samples = 10000 and Various Sample Size**

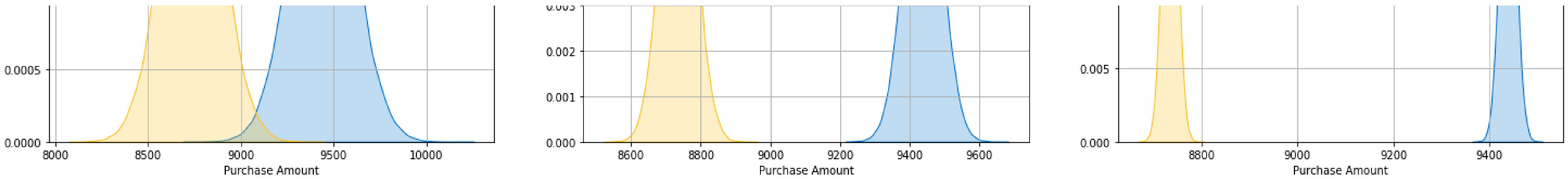
- **We can observe that when the sample size is as low as 50 which means expecting 50 customers of both Male and Female are making a purchase, then we can see that there is a clear overlap of mean of purchase amount of both male and female.**
- **As the sample size increases we can observe a huge gap between the mean amounts of both male and female, which clearly indicates that the highest amounts of purchases are done by male over female.**
- **We can also observe that the 95% and 99% Confidence Intervals that there is a huge gap in average purchase amount of both male and female and they don't overlap with each other.**

In [133]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 100000
index = 1
male_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
female_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
male_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
female_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
```

```
    plt.subplot(2,3,index)
    male_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "M"]["Purchase"])
    female_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Gender"] == "F"]["Purchase"])
    male_ms_100000["Mean"][f'Sample Size = {size}'], male_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(male_purchase_mean).round(2), np.std(male_purchase_mean).round(2)
    female_ms_100000["Mean"][f'Sample Size = {size}'], female_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(female_purchase_mean).round(2), np.std(female_purchase_mean).round(2)
    male_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (1.96 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (1.96 * np.std(male_purchase_mean))).round(2))
    female_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (1.96 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (1.96 * np.std(female_purchase_mean))).round(2))
    male_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(male_purchase_mean) - (2.58 * np.std(male_purchase_mean))).round(2), (np.mean(male_purchase_mean) + (2.58 * np.std(male_purchase_mean))).round(2))
    female_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(female_purchase_mean) - (2.58 * np.std(female_purchase_mean))).round(2), (np.mean(female_purchase_mean) + (2.58 * np.std(female_purchase_mean))).round(2))
    sns.kdeplot(male_purchase_mean, fill = True, color = walmart_blue)
    sns.kdeplot(female_purchase_mean, fill = True, color = walmart_orange)
    plt.xlabel('Purchase Amount')
    plt.title (f'Sample Size = {size}')
    plt.legend(["Male", "Female"])
    plt.grid()
    index += 1
plt.suptitle('Number of Samples = 100000 & various Sample Sizes to check\nPurchases Distribution among Male vs Female', fontsize = 18, fontweight = 'bold')
plt.show()
```





In [134]:

```
male_ms_100000 = pd.DataFrame(male_ms_100000)
male_ci_100000 = pd.DataFrame(male_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for Male Customers\n\n{male_ms_100000}')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{male_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for Male Customers

                        Mean  Standard_Error
Sample Size = 50      9438.49          720.21
Sample Size = 250     9436.85          322.38
Sample Size = 500     9439.26          229.01
Sample Size = 1000    9437.91          161.36
Sample Size = 10000   9437.37           50.90
Sample Size = 100000  9437.54           16.14
--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50         (8026.89, 10850.1)    (7580.36, 11296.62)
Sample Size = 250       (8804.99, 10068.71)    (8605.12, 10268.58)
Sample Size = 500        (8990.41, 9888.11)     (8848.42, 10030.1)
Sample Size = 1000       (9121.65, 9754.16)      (9021.61, 9854.2)
Sample Size = 10000      (9337.61, 9537.12)     (9306.06, 9568.68)
Sample Size = 100000      (9405.9, 9469.19)      (9395.89, 9479.2)
```

In [135]:

```
female_ms_100000 = pd.DataFrame(female_ms_100000)
female_ci_100000 = pd.DataFrame(female_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for female Customers\n\n{female_ms_100000}')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{female_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for female Customers

                        Mean  Standard_Error
Sample Size = 50      8732.36          673.05
Sample Size = 250     8735.12          301.23
Sample Size = 500     8734.63          212.86
Sample Size = 1000    8734.58          150.86
Sample Size = 10000   8734.54           47.71
Sample Size = 100000  8734.54           15.09
--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7413.18, 10051.53)    (6995.89, 10468.82)
Sample Size = 250        (8144.71, 9325.54)      (7957.94, 9512.3)
Sample Size = 500        (8317.43, 9151.84)     (8185.46, 9283.81)
Sample Size = 1000       (8438.89, 9030.28)     (8345.35, 9123.81)
Sample Size = 10000      (8641.04, 8828.05)     (8611.46, 8857.63)
Sample Size = 100000     (8704.97, 8764.11)     (8695.61, 8773.47)
```

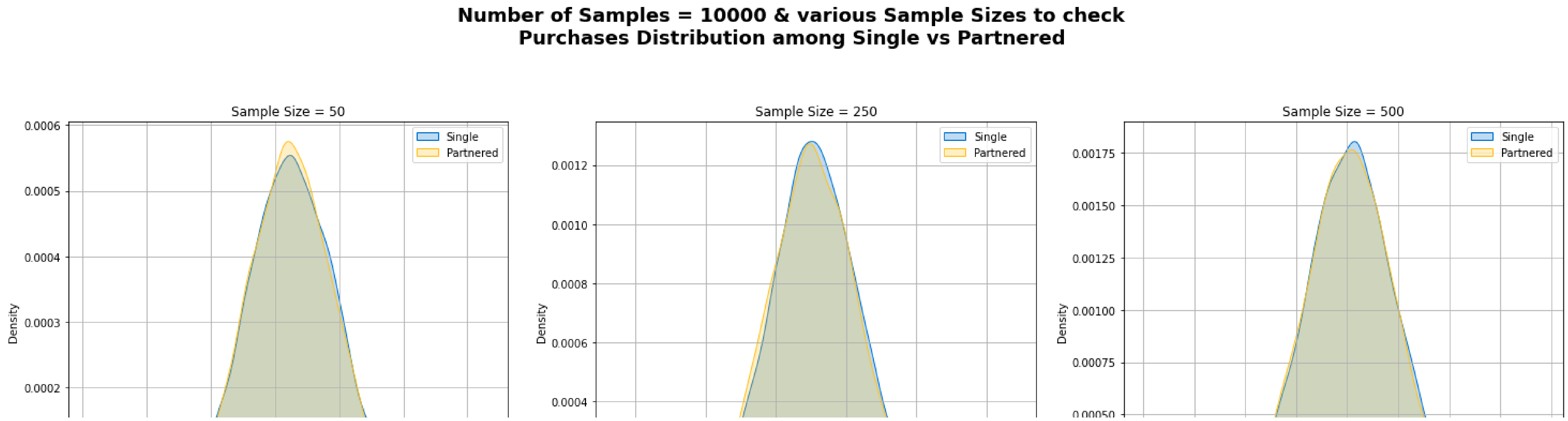
Comment: **Nummber of Samples = 100000 and Various Sample Size**

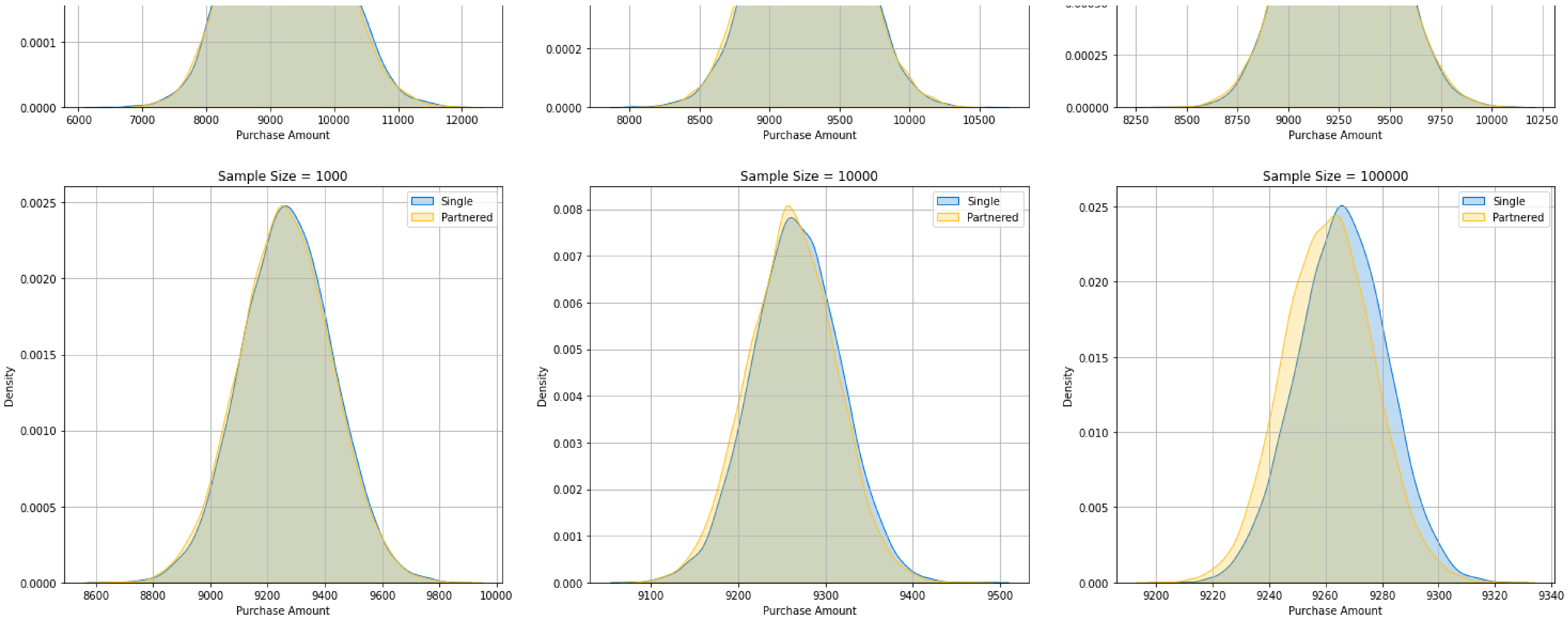
- **We can observe that when the sample size is as low as 50 which means expecting 50 customers of both Male and Female are making a purchase, then we can see that there is a clear overlap of mean of purchase amount of both male and female.**
- **As the sample size increases we can observe a huge gap between the mean amounts of both male and female, which clearly indicates that the highest amounts of purchases are done by male over female.**
- **We can also observe that the 95% and 99% Confidence Intervals that there is a huge gap in average purchase amount of both male and female and they don't overlap with each other.**

Marital Status (Single vs Married) Wise Purchase Distribution

In [136]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 10000
index = 1
single_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
partnered_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
single_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
partnered_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  single_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Marital_Status"] == "Single"]["Purchase"])
  partnered_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Marital_Status"] == "Partnered"]["Purchase"])
  single_ms_10000["Mean"][f'Sample Size = {size}'], single_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(single_purchase_mean).round(2), np.std(single_purchase_mean).round
(2)
  partnered_ms_10000["Mean"][f'Sample Size = {size}'], partnered_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(partnered_purchase_mean).round(2), np.std(partnered_purchase_
mean).round(2)
  single_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(single_purchase_mean) - (1.96 * np.std(single_purchase_mean))).round(2), (np.mean(single_purchase_mean) + (
1.96 * np.std(single_purchase_mean))).round(2))
  partnered_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(partnered_purchase_mean) - (1.96 * np.std(partnered_purchase_mean))).round(2), (np.mean(partnered_purcha
se_mean) + (1.96 * np.std(partnered_purchase_mean))).round(2))
  single_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(single_purchase_mean) - (2.58 * np.std(single_purchase_mean))).round(2), (np.mean(single_purchase_mean) + (
2.58 * np.std(single_purchase_mean))).round(2))
  partnered_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(partnered_purchase_mean) - (2.58 * np.std(partnered_purchase_mean))).round(2), (np.mean(partnered_purcha
se_mean) + (2.58 * np.std(partnered_purchase_mean))).round(2))
  sns.kdeplot(single_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(partnered_purchase_mean, fill = True, color = walmart_orange)
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["Single", "Partnered"])
  plt.grid()
  index += 1
plt.suptitle('Number of Samples = 10000 & various Sample Sizes to check\nPurchases Distribution among Single vs Partnered', fontsize = 18, fontweight = 'bold')
plt.show()
```





In [137]:

```python
single_ms_10000 = pd.DataFrame(single_ms_10000)
single_ci_10000 = pd.DataFrame(single_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for single Customers\n\n{single_ms_10000}')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{single_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for single Customers

                         Mean  Standard_Error
Sample Size = 50      9264.18          714.60
Sample Size = 250     9272.89          315.29
Sample Size = 500     9266.07          222.42
Sample Size = 1000    9267.81          159.63
Sample Size = 10000   9265.98           50.72
Sample Size = 100000  9265.80           15.87
--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50         (7863.57, 10664.8)    (7420.52, 11107.85)
Sample Size = 250        (8654.92, 9890.86)    (8459.43, 10086.34)
Sample Size = 500         (8830.14, 9702.0)      (8692.24, 9839.9)
Sample Size = 1000       (8954.93, 9580.69)     (8855.96, 9679.67)
Sample Size = 10000       (9166.56, 9365.4)     (9135.11, 9396.85)
Sample Size = 100000      (9234.7, 9296.91)     (9224.86, 9306.75)
```

In [138]:

```python
partnered_ms_10000 = pd.DataFrame(partnered_ms_10000)
partnered_ci_10000 = pd.DataFrame(partnered_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for partnered Customers\n\n{partnered_ms_10000}')
```

```
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{partnered_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for partnered Customers

                        Mean  Standard_Error
Sample Size = 50      9248.56          704.63
Sample Size = 250     9261.38          318.92
Sample Size = 500     9262.54          224.93
Sample Size = 1000    9261.22          160.86
Sample Size = 10000   9260.70           50.46
Sample Size = 100000  9261.05           15.93
--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7867.47, 10629.64)     (7430.6, 11066.52)
Sample Size = 250         (8636.3, 9886.45)    (8438.57, 10084.18)
Sample Size = 500        (8821.67, 9703.41)     (8682.21, 9842.87)
Sample Size = 1000       (8945.93, 9576.52)     (8846.19, 9676.26)
Sample Size = 10000       (9161.81, 9359.6)     (9130.52, 9390.88)
Sample Size = 100000     (9229.83, 9292.28)     (9219.95, 9302.16)
```

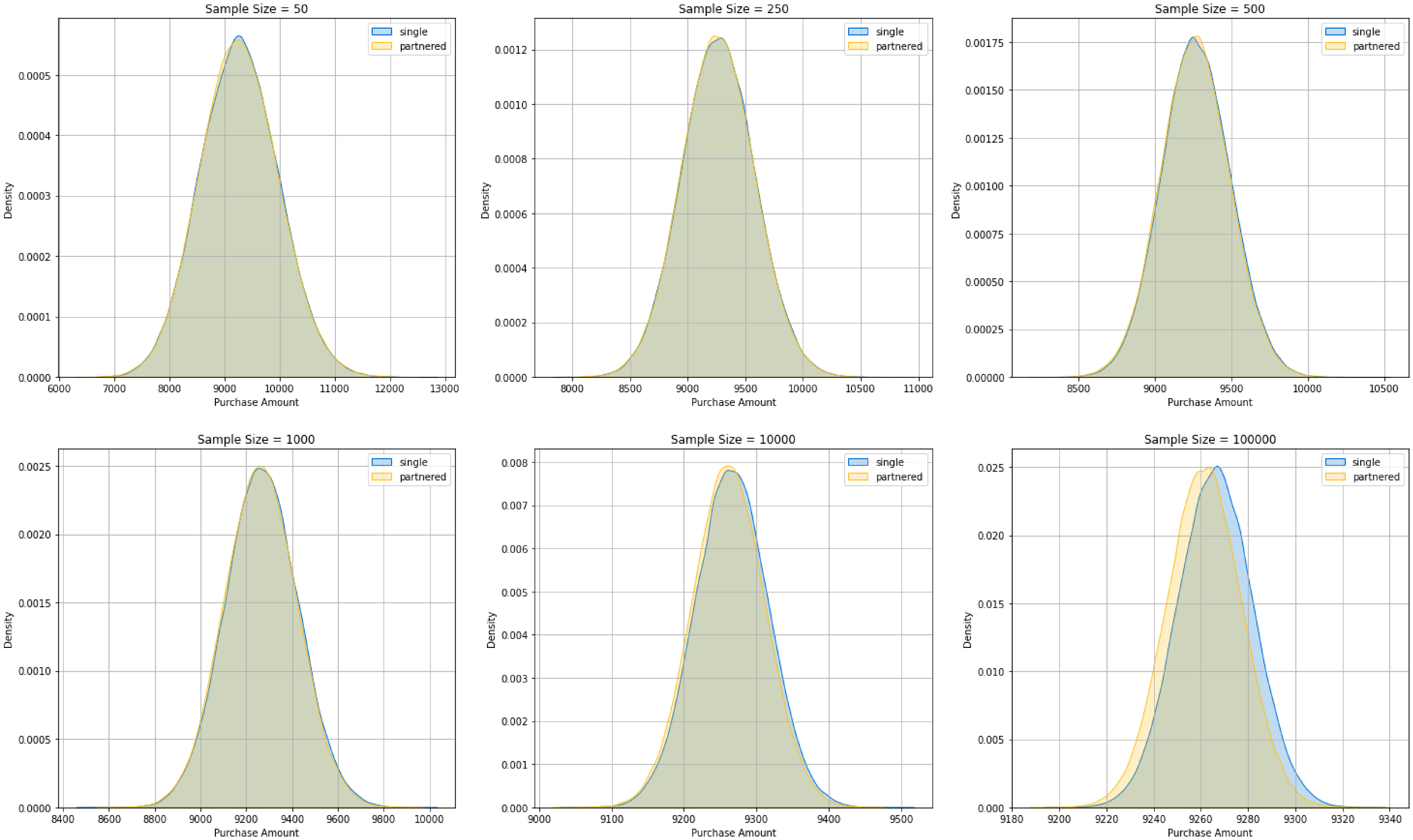
Comment: **Nummber of Samples = 10000 and Various Sample Size**

- **No matter the sample size we can clearly observe that the mean purchase amount for both Single and Partnered customers are very close to similar.**
- **So the purchase partern of either single or partnered are close to similar.**
- **Both in case of 95% Confidence Interval and 99% Confidence Interval we can observe a clear overlap of purchase mean amounts with each other.**

In [139]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 100000
index = 1
single_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
partnered_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
single_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
partnered_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  single_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Marital_Status"] == "Single"]["Purchase"])
  partnered_purchase_mean = sample_means(size, number_samples, walmart_data[walmart_data["Marital_Status"] == "Partnered"]["Purchase"])
  single_ms_100000["Mean"][f'Sample Size = {size}'], single_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(single_purchase_mean).round(2), np.std(single_purchase_mean).round(2)
  partnered_ms_100000["Mean"][f'Sample Size = {size}'], partnered_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(partnered_purchase_mean).round(2), np.std(partnered_purchase_mean).round(2)
  single_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(single_purchase_mean) - (1.96 * np.std(single_purchase_mean))).round(2), (np.mean(single_purchase_mean) + (1.96 * np.std(single_purchase_mean))).round(2))
  partnered_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(partnered_purchase_mean) - (1.96 * np.std(partnered_purchase_mean))).round(2), (np.mean(partnered_purchase_mean) + (1.96 * np.std(partnered_purchase_mean))).round(2))
  single_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(single_purchase_mean) - (2.58 * np.std(single_purchase_mean))).round(2), (np.mean(single_purchase_mean) + (2.58 * np.std(single_purchase_mean))).round(2))
  partnered_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(partnered_purchase_mean) - (2.58 * np.std(partnered_purchase_mean))).round(2), (np.mean(partnered_purchase_mean) + (2.58 * np.std(partnered_purchase_mean))).round(2))
  sns.kdeplot(single_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(partnered_purchase_mean, fill = True, color = walmart_orange)
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["single", "partnered"])
  plt.grid()
  index += 1
plt.suptitle('Number of Samples = 100000 & various Sample Sizes to check\nPurchases Distribution among single vs partnered', fontsize = 18, fontweight = 'bold')
plt.show()
```

Number of Samples = 100000 & various Sample Sizes to check
Purchases Distribution among single vs partnered



In [140]:

```
single_ms_100000 = pd.DataFrame(single_ms_100000)
single_ci_100000 = pd.DataFrame(single_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for single Customers\n\n{single_ms_100000}')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{single_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for single Customers

                        Mean  Standard_Error
Sample Size = 50     9264.09          710.64
Sample Size = 250    9263.35          318.03
Sample Size = 500    9266.88          224.74
Sample Size = 1000   9266.23          159.30
```

```
Sample Size = 10000   9265.63           50.48
Sample Size = 100000  9265.94           15.94
--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7871.23, 10656.95)    (7430.63, 11097.55)
Sample Size = 250        (8640.02, 9886.68)    (8442.84, 10083.86)
Sample Size = 500         (8826.4, 9707.37)      (8687.06, 9846.7)
Sample Size = 1000        (8954.0, 9578.47)     (8855.23, 9677.23)
Sample Size = 10000       (9166.7, 9364.57)      (9135.4, 9395.87)
Sample Size = 100000     (9234.69, 9297.19)     (9224.81, 9307.08)
```

In [141]:

```
partnered_ms_100000 = pd.DataFrame(partnered_ms_100000)
partnered_ci_100000 = pd.DataFrame(partnered_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for partnered Customers\n\n{partnered_ms_100000}')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{partnered_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for partnered Customers

                         Mean  Standard_Error
Sample Size = 50      9259.63          711.26
Sample Size = 250     9260.51          317.48
Sample Size = 500     9259.64          224.81
Sample Size = 1000    9260.67          158.52
Sample Size = 10000   9261.08           50.13
Sample Size = 100000  9261.19           15.88
--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50         (7865.57, 10653.7)    (7424.59, 11094.67)
Sample Size = 250        (8638.24, 9882.77)    (8441.41, 10079.61)
Sample Size = 500        (8819.02, 9700.26)     (8679.64, 9839.64)
Sample Size = 1000       (8949.97, 9571.37)     (8851.69, 9669.66)
Sample Size = 10000      (9162.81, 9359.34)     (9131.73, 9390.42)
Sample Size = 100000     (9230.06, 9292.32)     (9220.21, 9302.16)
```

Comment: **Nummber of Samples = 100000 and Various Sample Size**

- **No matter the sample size we can clearly observe that the mean purchase amount for both Single and Partnered customers are very close to similar.**
- **So the purchase parttern of either single or partnered are close to similar.**
- **Both in case of 95% Confidence Interval and 99% Confidence Interval we can observe a clear overlap of purchase mean amounts with each other.**

Age Wise Purchase Distribution

In [284]:

```
np.sort(walmart_data["Age"].unique())
```

Out[284]:

```
array(['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+'],
      dtype=object)
```

In [142]:

```
age_distribution_data = walmart_data.copy()
```

In [143]:

```python
def age_category (category):
  if category == '0-17' or category == '18-25':
    return 'Below 26'
  elif category == '26-35' or category == '36-45':
    return '26-45'
  else:
    return 'Above 45'

age_distribution_data ["Age_Category"] = age_distribution_data["Age"].apply(age_category)
```

In [144]:

```python
age_distribution_data.head()
```

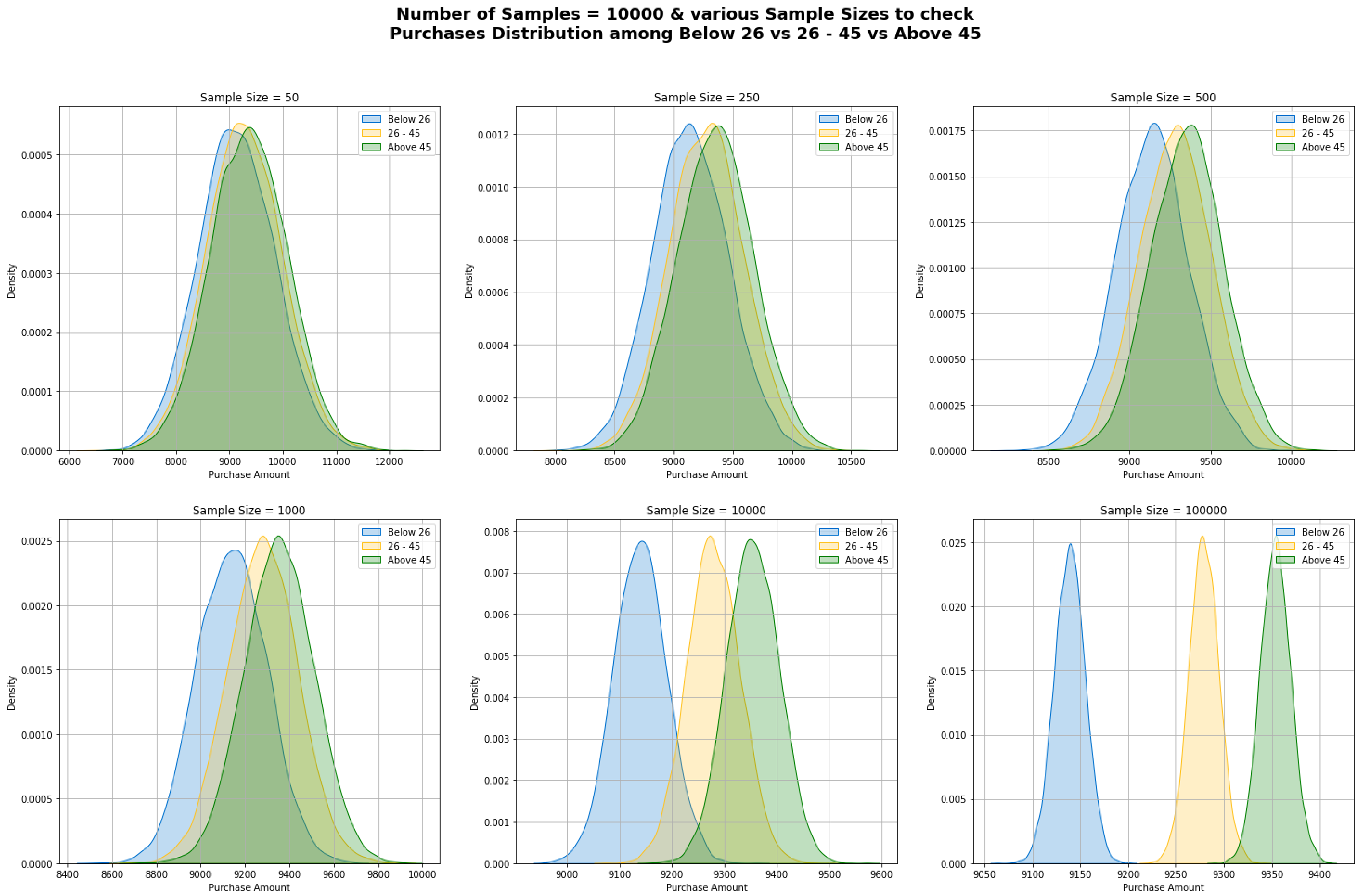
Out[144]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase	Age_Category
0	1000001	P00069042	F	0-17	10	A	2	Single	3	8370	Below 26
1	1000001	P00248942	F	0-17	10	A	2	Single	1	15200	Below 26
2	1000001	P00087842	F	0-17	10	A	2	Single	12	1422	Below 26
3	1000001	P00085442	F	0-17	10	A	2	Single	12	1057	Below 26
4	1000002	P00285442	M	55+	16	C	4+	Single	8	7969	Above 45

In [145]:

```python
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 10000
index = 1
below26_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
above26_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
above45_ms_10000 = {'Mean':{} ,'Standard_Error':{}}
below26_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
above26_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
above45_ci_10000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  below26_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "Below 26"]["Purchase"])
  above26_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "26-45"]["Purchase"])
  above45_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "Above 45"]["Purchase"])
  below26_ms_10000["Mean"][f'Sample Size = {size}'], below26_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(below26_purchase_mean).round(2), np.std(below26_purchase_mean).r
ound(2)
  above26_ms_10000["Mean"][f'Sample Size = {size}'], above26_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(above26_purchase_mean).round(2), np.std(above26_purchase_mean).r
ound(2)
  above45_ms_10000["Mean"][f'Sample Size = {size}'], above45_ms_10000["Standard_Error"][f'Sample Size = {size}'] = np.mean(above45_purchase_mean).round(2), np.std(above45_purchase_mean).r
ound(2)
  below26_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(below26_purchase_mean) - (1.96 * np.std(below26_purchase_mean))).round(2), (np.mean(below26_purchase_mean)
+ (1.96 * np.std(below26_purchase_mean))).round(2))
  above26_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(above26_purchase_mean) - (1.96 * np.std(above26_purchase_mean))).round(2), (np.mean(above26_purchase_mean)
+ (1.96 * np.std(above26_purchase_mean))).round(2))
  above45_ci_10000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(above45_purchase_mean) - (1.96 * np.std(above45_purchase_mean))).round(2), (np.mean(above45_purchase_mean)
+ (1.96 * np.std(above45_purchase_mean))).round(2))
  below26_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(below26_purchase_mean) - (2.58 * np.std(below26_purchase_mean))).round(2), (np.mean(below26_purchase_mean)
+ (2.58 * np.std(below26_purchase_mean))).round(2))
  above26_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(above26_purchase_mean) - (2.58 * np.std(above26_purchase_mean))).round(2), (np.mean(above26_purchase_mean)
+ (2.58 * np.std(above26_purchase_mean))).round(2))
  above45_ci_10000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(above45_purchase_mean) - (2.58 * np.std(above45_purchase_mean))).round(2), (np.mean(above45_purchase_mean)
+ (2.58 * np.std(above45_purchase_mean))).round(2))
  sns.kdeplot(below26_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(above26_purchase_mean, fill = True, color = walmart_orange)
  sns.kdeplot(above45_purchase_mean, fill = True, color = 'g')
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["Below 26", "26 - 45", "Above 45"])
  plt.grid()
  index += 1
plt.suptitle('Number of Samples = 10000 & various Sample Sizes to check\nPurchases Distribution among Below 26 vs 26 - 45 vs Above 45', fontsize = 18, fontweight = 'bold')
```

```
plt.show()
```



In [146]:

```
below26_ms_10000 = pd.DataFrame(below26_ms_10000)
below26_ci_10000 = pd.DataFrame(below26_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for Below 26 aged Customers\n\n{below26_ms_10000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{below26_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for Below 26 aged Customers
```

```
                            Mean  Standard_Error
Sample Size = 50      9132.54          716.93
Sample Size = 250     9138.37          318.24
Sample Size = 500     9136.36          225.45
Sample Size = 1000    9139.16          157.73
Sample Size = 10000   9139.14           50.85
Sample Size = 100000  9138.69           16.11

--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                      95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7727.36, 10537.73)    (7282.86, 10982.23)
Sample Size = 250        (8514.61, 9762.13)      (8317.3, 9959.44)
Sample Size = 500        (8694.48, 9578.24)      (8554.7, 9718.02)
Sample Size = 1000        (8830.02, 9448.3)     (8732.23, 9546.09)
Sample Size = 10000      (9039.47, 9238.81)     (9007.95, 9270.33)
Sample Size = 100000     (9107.12, 9170.26)     (9097.13, 9180.25)
```

In [147]:

```python
above26_ms_10000 = pd.DataFrame(above26_ms_10000)
above26_ci_10000 = pd.DataFrame(above26_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for 26 - 45 aged Customers\n\n{above26_ms_10000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{above26_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for 26 - 45 aged Customers

                            Mean  Standard_Error
Sample Size = 50      9271.50          708.92
Sample Size = 250     9278.09          314.53
Sample Size = 500     9280.96          222.82
Sample Size = 1000    9278.45          157.75
Sample Size = 10000   9278.65           50.53
Sample Size = 100000  9278.96           15.68

--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                      95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7882.02, 10660.97)     (7442.49, 11100.5)
Sample Size = 250        (8661.61, 9894.57)     (8466.6, 10089.58)
Sample Size = 500        (8844.22, 9717.69)     (8706.07, 9855.84)
Sample Size = 1000       (8969.26, 9587.65)     (8871.45, 9685.45)
Sample Size = 10000      (9179.62, 9377.68)     (9148.29, 9409.01)
Sample Size = 100000      (9248.22, 9309.7)      (9238.5, 9319.42)
```

In [148]:

```python
above45_ms_10000 = pd.DataFrame(above45_ms_10000)
above45_ci_10000 = pd.DataFrame(above45_ci_10000)
print (f'Total Samples = 10000 & Various Sample Sizes\nMean & Standard Error for Above 45 aged Customers\n\n{above45_ms_10000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 10000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{above45_ci_10000}')
```

```
Total Samples = 10000 & Various Sample Sizes
Mean & Standard Error for Above 45 aged Customers

                            Mean  Standard_Error
Sample Size = 50      9362.64          714.60
Sample Size = 250     9351.39          319.63
Sample Size = 500     9353.19          224.20
Sample Size = 1000    9353.00          157.85
Sample Size = 10000   9353.62           49.94
Sample Size = 100000  9353.61           15.90
```

```
--------------------------------------------------

Total Samples = 10000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7962.02, 10763.26)    (7518.96, 11206.32)
Sample Size = 250        (8724.91, 9977.87)    (8526.74, 10176.04)
Sample Size = 500        (8913.75, 9792.63)     (8774.75, 9931.64)
Sample Size = 1000       (9043.61, 9662.39)     (8945.74, 9760.26)
Sample Size = 10000       (9255.73, 9451.5)     (9224.77, 9482.47)
Sample Size = 100000     (9322.44, 9384.77)     (9312.58, 9394.63)
```

Comment: **Nummber of Samples = 10000 and Various Sample Size**

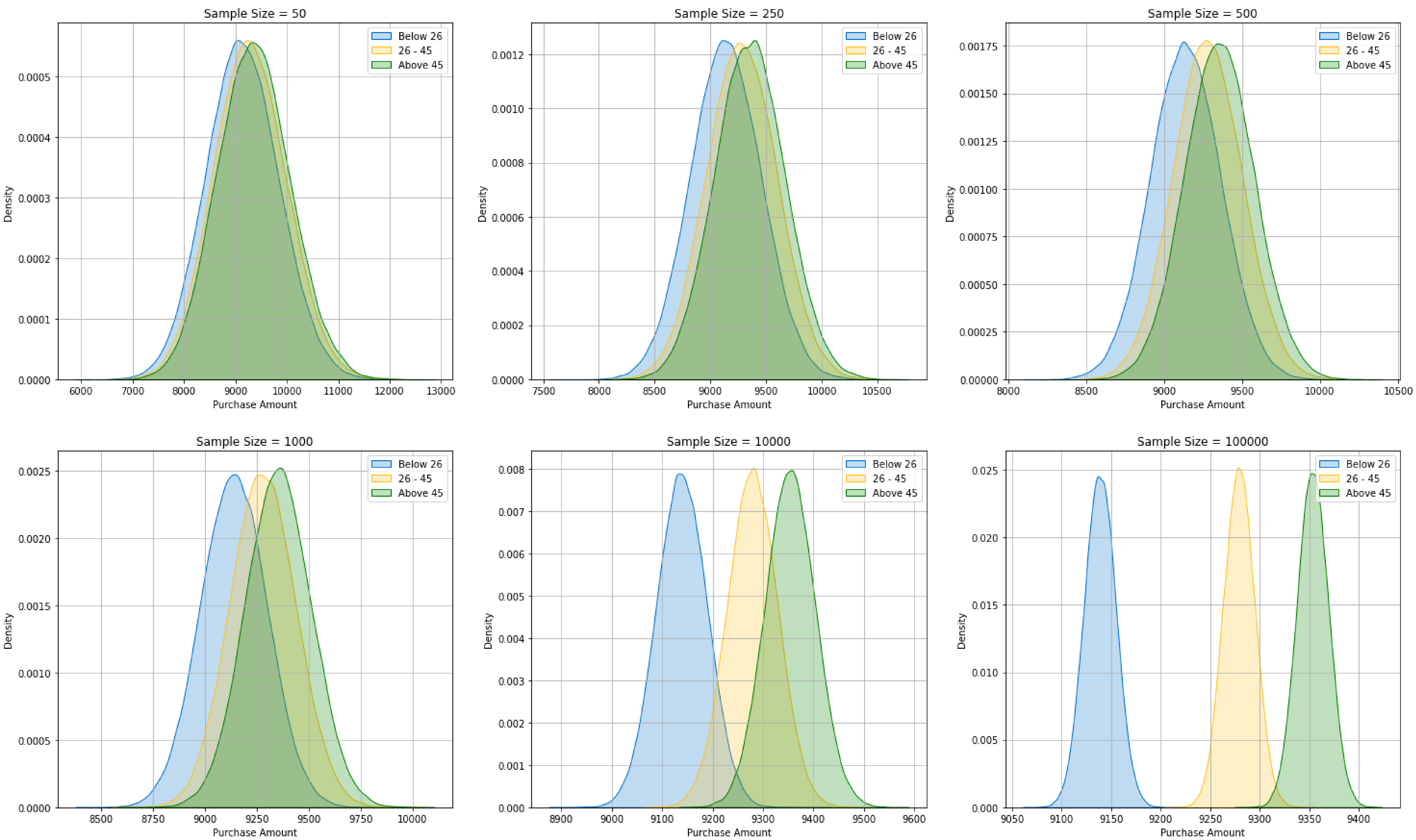
- **Almost till sample size of 1000 we can see all three puchase amount maps overlap with each other and the mean purchase amounts are very close to each other.**
- **But then when we move to a higher Sample Size of Purchase Amounts, we can clearly observe that the overlap of purchase amounts reduce between all three age groups. The mean purchase amount for all three age groups drastically increase.**
- **We can see that people of age group 26-45 spend moderately well while Above 45 customers spend higher amount on a product purchase.**
- **While customers Below 26 tend to spend amount lesser when compared to the other two age groups.**
- **We can also observe that the 95% Confidence Interval and 99% Confidence Intervals of purchase amounts don't overlap between the three age groups when the sample size is 10000 or above.**

In [149]:

```
plt.figure(figsize = (25,15))
sample_sizes = [50, 250, 500, 1000, 10000, 100000]
number_samples = 100000
index = 1
below26_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
above26_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
above45_ms_100000 = {'Mean':{} ,'Standard_Error':{}}
below26_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
above26_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
above45_ci_100000 = {'95% CI (2.5% to 97.5%)':{}, '99% CI (0.5% to 99.5%)':{}}
for size in sample_sizes:
  plt.subplot(2,3,index)
  below26_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "Below 26"]["Purchase"])
  above26_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "26-45"]["Purchase"])
  above45_purchase_mean = sample_means(size, number_samples, age_distribution_data[age_distribution_data ["Age_Category"] == "Above 45"]["Purchase"])
  below26_ms_100000["Mean"][f'Sample Size = {size}'], below26_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(below26_purchase_mean).round(2), np.std(below26_purchase_mean).round(2)
  above26_ms_100000["Mean"][f'Sample Size = {size}'], above26_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(above26_purchase_mean).round(2), np.std(above26_purchase_mean).round(2)
  above45_ms_100000["Mean"][f'Sample Size = {size}'], above45_ms_100000["Standard_Error"][f'Sample Size = {size}'] = np.mean(above45_purchase_mean).round(2), np.std(above45_purchase_mean).round(2)
  below26_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(below26_purchase_mean) - (1.96 * np.std(below26_purchase_mean))).round(2), (np.mean(below26_purchase_mean) + (1.96 * np.std(below26_purchase_mean))).round(2))
  above26_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(above26_purchase_mean) - (1.96 * np.std(above26_purchase_mean))).round(2), (np.mean(above26_purchase_mean) + (1.96 * np.std(above26_purchase_mean))).round(2))
  above45_ci_100000["95% CI (2.5% to 97.5%)"][f'Sample Size = {size}'] = ((np.mean(above45_purchase_mean) - (1.96 * np.std(above45_purchase_mean))).round(2), (np.mean(above45_purchase_mean) + (1.96 * np.std(above45_purchase_mean))).round(2))
  below26_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(below26_purchase_mean) - (2.58 * np.std(below26_purchase_mean))).round(2), (np.mean(below26_purchase_mean) + (2.58 * np.std(below26_purchase_mean))).round(2))
  above26_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(above26_purchase_mean) - (2.58 * np.std(above26_purchase_mean))).round(2), (np.mean(above26_purchase_mean) + (2.58 * np.std(above26_purchase_mean))).round(2))
  above45_ci_100000["99% CI (0.5% to 99.5%)"][f'Sample Size = {size}'] = ((np.mean(above45_purchase_mean) - (2.58 * np.std(above45_purchase_mean))).round(2), (np.mean(above45_purchase_mean) + (2.58 * np.std(above45_purchase_mean))).round(2))
  sns.kdeplot(below26_purchase_mean, fill = True, color = walmart_blue)
  sns.kdeplot(above26_purchase_mean, fill = True, color = walmart_orange)
  sns.kdeplot(above45_purchase_mean, fill = True, color = 'g')
  plt.xlabel('Purchase Amount')
  plt.title (f'Sample Size = {size}')
  plt.legend(["Below 26", "26 - 45", "Above 45"])
  plt.grid()
  index += 1
plt.suptitle('Number of Samples = 100000 & various Sample Sizes to check\nPurchases Distribution among Below 26 vs 26 - 45 vs Above 45', fontsize = 18, fontweight = 'bold')
plt.show()
```

Number of Samples = 100000 & various Sample Sizes to check
Purchases Distribution among Below 26 vs 26 - 45 vs Above 45

Purchases Distribution among Below 26 vs 26 - 45 vs Above 45



In [150]:

```
below26_ms_100000 = pd.DataFrame(below26_ms_100000)
below26_ci_100000 = pd.DataFrame(below26_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for Below 26 aged Customers\n\n{below26_ms_100000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{below26_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for Below 26 aged Customers

                      Mean  Standard_Error
Sample Size = 50     9136.64          711.74
Sample Size = 250    9138.10          318.50
Sample Size = 500    9138.64          226.01
```

```
Sample Size = 1000    9138.42          159.79
Sample Size = 10000   9138.69           50.39
Sample Size = 100000  9138.69           15.97

--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7741.62, 10531.65)    (7300.34, 10972.93)
Sample Size = 250        (8513.84, 9762.37)     (8316.37, 9959.84)
Sample Size = 500        (8695.67, 9581.62)     (8555.54, 9721.74)
Sample Size = 1000       (8825.22, 9451.61)     (8726.15, 9550.68)
Sample Size = 10000      (9039.93, 9237.45)     (9008.69, 9268.68)
Sample Size = 100000     (9107.39, 9169.98)     (9097.49, 9179.88)
```

In [151]:

```
above26_ms_100000 = pd.DataFrame(above26_ms_100000)
above26_ci_100000 = pd.DataFrame(above26_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for 26 - 45 aged Customers\n\n{above26_ms_100000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{above26_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for 26 - 45 aged Customers

                         Mean  Standard_Error
Sample Size = 50      9278.94          710.86
Sample Size = 250     9279.70          317.50
Sample Size = 500     9278.06          223.78
Sample Size = 1000    9279.04          158.87
Sample Size = 10000   9278.81           50.11
Sample Size = 100000  9279.05           15.82

--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval

                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7885.66, 10672.22)    (7444.92, 11112.95)
Sample Size = 250        (8657.39, 9902.01)    (8460.54, 10098.86)
Sample Size = 500        (8839.45, 9716.68)      (8700.7, 9855.43)
Sample Size = 1000       (8967.65, 9590.42)     (8869.15, 9688.92)
Sample Size = 10000       (9180.6, 9377.01)     (9149.54, 9408.08)
Sample Size = 100000     (9248.05, 9310.05)     (9238.25, 9319.86)
```

In [152]:

```
above45_ms_100000 = pd.DataFrame(above45_ms_100000)
above45_ci_100000 = pd.DataFrame(above45_ci_100000)
print (f'Total Samples = 100000 & Various Sample Sizes\nMean & Standard Error for Above 45 aged Customers\n\n{above45_ms_100000}\n')
print ("-" * 50, "\n")
print (f'Total Samples = 100000 & Various Sample Sizes\n95% Confidence Interval & 99% Confidence Interval\n\n{above45_ci_100000}')
```

```
Total Samples = 100000 & Various Sample Sizes
Mean & Standard Error for Above 45 aged Customers

                         Mean  Standard_Error
Sample Size = 50      9354.12          712.27
Sample Size = 250     9353.07          316.75
Sample Size = 500     9354.32          223.95
Sample Size = 1000    9353.21          159.01
Sample Size = 10000   9353.60           50.10
Sample Size = 100000  9353.39           15.92

--------------------------------------------------

Total Samples = 100000 & Various Sample Sizes
95% Confidence Interval & 99% Confidence Interval
```

```
                     95% CI (2.5% to 97.5%) 99% CI (0.5% to 99.5%)
Sample Size = 50        (7958.07, 10750.17)    (7516.47, 11191.78)
Sample Size = 250        (8732.24, 9973.91)     (8535.85, 10170.3)
Sample Size = 500        (8915.37, 9793.26)     (8776.52, 9932.11)
Sample Size = 1000       (9041.56, 9664.87)     (8942.98, 9763.45)
Sample Size = 10000       (9255.4, 9451.79)     (9224.34, 9482.85)
Sample Size = 100000     (9322.19, 9384.59)     (9312.32, 9394.46)
```

Comment: **Nummber of Samples = 100000 and Various Sample Size**

- **Almost till sample size of 1000 we can see all three puchase amount maps overlap with each other and the mean purchase amounts are very close to each other.**
- **But then when we move to a higher Sample Size of Purchase Amounts, we can clearly observe that the overlap of purchase amounts reduce between all three age groups. The mean purchase amount for all three age groups drastically increase.**
- **We can see that people of age group 26-45 spend moderately well while Above 45 customers spend higher amount on a product purchase.**
- **While customers Below 26 tend to spend amount lesser when compared to the other two age groups.**
- **We can also observe that the 95% Confidence Interval and 99% Confidence Intervals of purchase amounts don't overlap between the three age groups when the sample size is 10000 or above.**

Recommendations

- **To attract a younger demographic and boost sales during Black Friday, Walmart should consider creating interactive games that appeal to this demographic. These games can be located throughout the store and should be designed to provide an enjoyable and engaging experience for younger customers.**
- **Since women tend to spend less than men on Black Friday, Walmart should consider offering incentives such as discounts or special offers that are specifically targeted towards women. By doing so, Walmart can encourage women to spend more and increase their overall sales during Black Friday.**
- **To draw in more young customers during Black Friday, Walmart can provide kid-friendly games and activities that are both fun and engaging. These games can be designed to appeal to children of all ages, and should be located in areas of the store that are easily accessible and visible to customers.**
- **To boost sales during Black Friday, Walmart can create various promotions and offers that are targeted towards children aged 0-17. This can include special discounts, offers, and events that are specifically designed to appeal to this demographic.**
- **Since men tend to spend more than women on Black Friday, Walmart should consider targeting their marketing efforts towards men. This can include offering promotions and incentives that are specifically designed to appeal to male customers, such as discounts on popular men's products during Black Friday.**
- **Since demand for certain products seems to be high during Black Friday, Walmart should ensure that they are properly stocked up in warehouses. This can help to ensure that they are able to meet customer demand and avoid running out of popular items during the busy sales period.**
- **To cater to young and middle-aged customers during Black Friday, Walmart should introduce new products that are specifically designed to appeal to these demographics. This can help to attract new customers and increase sales during the sales period.**
- **Based on the data showing that males tend to experience more transgressions than females, Walmart should consider implementing measures to address this issue during Black Friday. This can include increasing security measures in areas where male customers tend to frequent during the busy sales period.**
- **Since male spending tends to be higher than female spending during Black Friday, Walmart should consider offering promotions and incentives that are specifically targeted towards male customers. This can help to increase sales and boost overall revenue during the sales period.**
- **To attract more female customers during Black Friday, Walmart can launch a marketing effort that is specifically designed to appeal to women. This can include offering promotions and incentives that are tailored to female customers, such as discounts on popular women's products during the sales period.**