In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats as st
from scipy.stats import binom,norm,poisson,expon,geom
from google.colab import files
```

In [2]:
```
walmart_file = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving walmart_data.csv to walmart_data.csv
```

Problem statement - The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers

Analysing Basic Metrics - Exploratory Data Analysis(EDA)

Reading the Dataset

In [4]:
```
walmart = pd.read_csv("walmart_data.csv")
```

In [5]:
```
walmart.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 [6]:
```
type(walmart)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
walmart.dtypes
```

Out[7]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

There are 10 columns - 5 columns of type object and 5 columns of type int

In [8]:
```
walmart.columns
```

Out[8]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [9]:
```
walmart.shape
```

Out[9]:
```
(550068, 10)
```

There are 5,50,068 transaction details given in the dataset

In [10]:
```
walmart.describe()
```

Out[10]:

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

Checking for NA values

In [11]:
```
walmart.isna().sum()
```

Out[11]:
```
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
```

There are no missing/NA values in the given dataset

Conversion of categorical attributes to 'category'

Categorizing purchase column

In [12]:
```
print("Min_Purchase =",min(walmart.Purchase))
print("Max_Purchase =",max(walmart.Purchase))
```

```
Min_Purchase = 12
Max_Purchase = 23961
```

In [13]:
```
def purchase_bins(purchase):
  if (purchase <= 5000): return 'Minimum_transaction'
  if (purchase > 5000 and purchase <= 15000): return 'Medium_transaction'
  if (purchase > 15000): return 'High_transaction'

walmart['Transaction_category'] = walmart['Purchase'].apply(purchase_bins)
walmart.head()
```

Out[13]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mar
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

Observations

- Given dataset is about Walmart.
- The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday
- Dataframe as 10 columns and 5,50,068 rows(Transactions happened)

- There are 5 columns of integer datatype and 5 columns of object datatype
- Looking at the describe data - purchase column has some outliers, other column data doesn't have any outliers
- Dataframe doesn't have any missing values.
- Post checking the dataframe, purchase column has been categorized as -

```
* purchase <= 5000 - Minimum transaction value
* 5000 < purchase <= 15000 - Medium transaction value
* purchase > 15000 - High transaction value
```

Non-Graphical Analysis - Value counts and unique attributes

In [14]:
```
walmart.Gender.value_counts()
```

Out[14]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [15]:
```
walmart.Product_ID.value_counts()
```

Out[15]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

In [16]:
```
walmart.Product_ID.nunique()
```

Out[16]:
```
3631
```

In [17]:
```
walmart.User_ID.value_counts()
```

Out[17]:
```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

In [18]:
```
walmart.User_ID.nunique()
```

Out[18]:
```
5891
```

In [23]:
```
walmart[["City_Category","Gender"]].value_counts(sort=False)
```

Out[23]:
```
City_Category  Gender
A              F          35704
               M         112016
B              F          57796
               M         173377
C              F          42309
               M         128866
dtype: int64
```

In [25]:
```
walmart.Product_Category.value_counts(sort=False)
```

Out[25]:
```
3      20213
1     140378
12      3947
8     113925
5     150933
4      11753
2      23864
6      20466
14      1523
11     24287
13      5549
15      6290
7       3721
16      9828
18      3125
10      5125
17       578
9        410
20      2550
19      1603
Name: Product_Category, dtype: int64
```

In [27]:
```
walmart.Product_Category.nunique()
```

Out[27]:
```
20
```

In [26]:
```
walmart.Transaction_category.value_counts()
```

Out[26]:
```
Medium_transaction     344622
High_transaction       110523
Minimum_transaction     94923
Name: Transaction_category, dtype: int64
```

In [34]:
```
print("Min_transaction amount =",min(walmart.Purchase)," ","Max_transaction amount
```

```
Min_transaction amount = 12   Max_transaction amount = 23961
```

In [35]:
```
walmart.Age.value_counts()
```

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

Observations

- In total there are 5891 customers who purchased on black friday.

- There are 4,14,259 transactions made by male customers and 1,35,809 transactions made by female customers.
- Accross 3 cities, male customers have purchased more than female.
- 3631 unique products are purchased by customers.
- Min_transaction amount = 12 Max_transaction amount = 23961.
- More transactions made by the customers who's age group is 26-25 followed by 36-45.

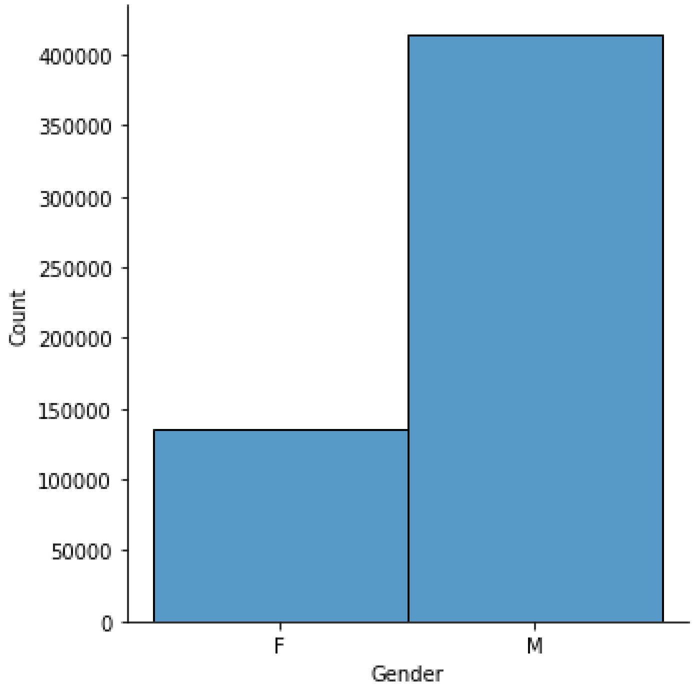
Visual Analysis - Univariate & Bivariate

Univariate Analysis

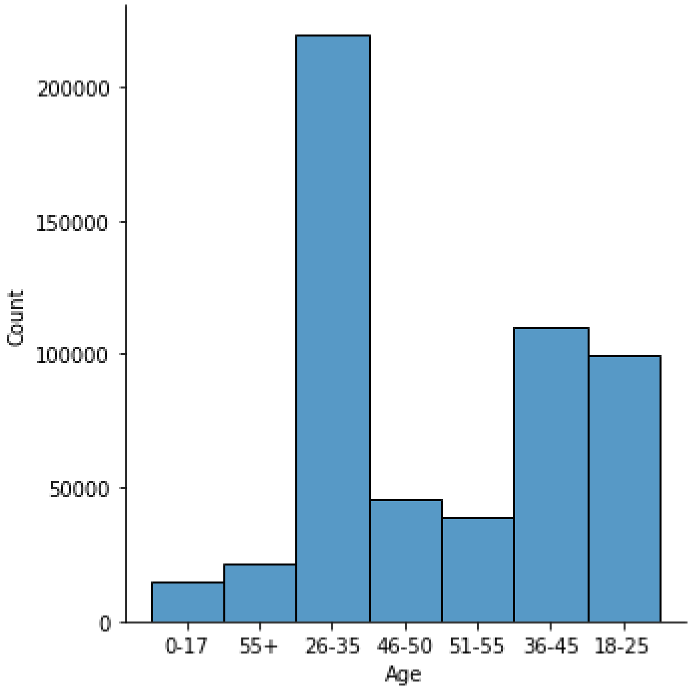
Displot

(Displot has been used instead of distplot as the function has been deprecated and will be removed in seaborn v0.14.0. It has been replaced by histplot() and displot())

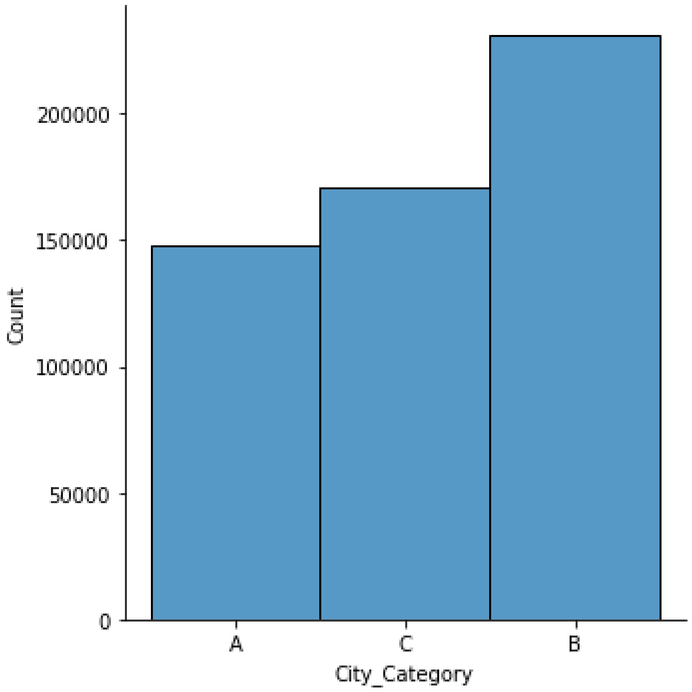
In [39]:
```
sns.displot(walmart.Gender)
plt.show()
```



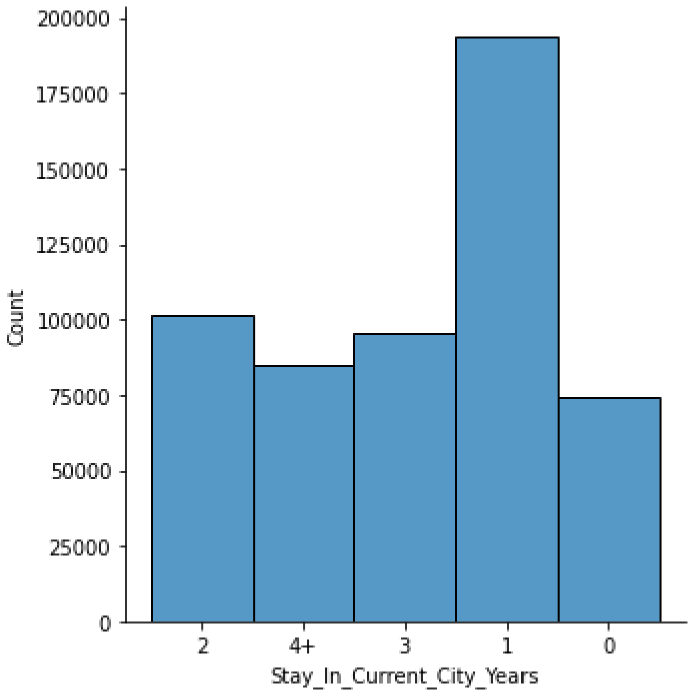
In [40]:
```
sns.displot(walmart.Age)
plt.show()
```



In [41]:
```
sns.displot(walmart.City_Category)
plt.show()
```



In [42]:
```
sns.displot(walmart.Stay_In_Current_City_Years)
plt.show()
```



Countplot

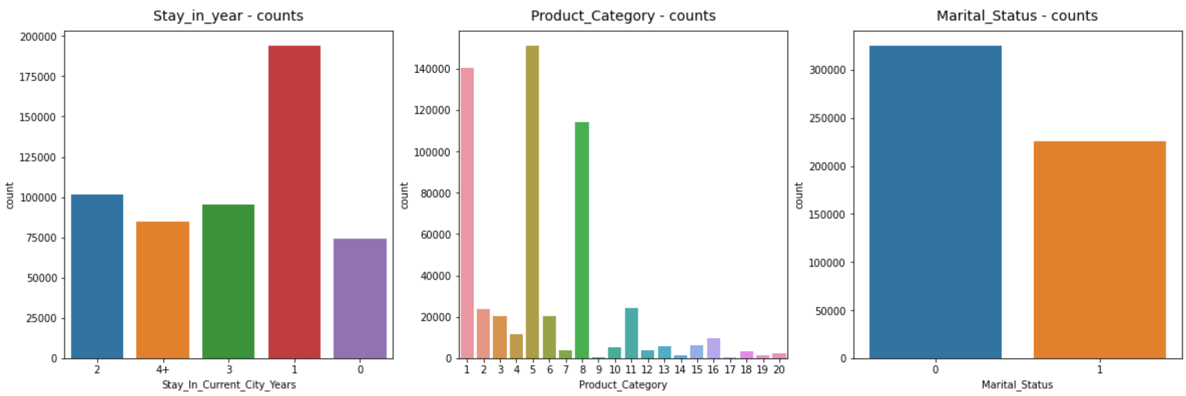
In [44]:

```python
fig, axs = plt.subplots(nrows=1, ncols = 3, figsize=(20,6))

sns.countplot(data=walmart, x='Stay_In_Current_City_Years', ax=axs[0])
sns.countplot(data=walmart, x='Product_Category', ax=axs[1])
sns.countplot(data=walmart, x='Marital_Status', ax=axs[2])

axs[0].set_title("Stay_in_year - counts", pad=10, fontsize=14)
axs[1].set_title("Product_Category - counts", pad=10, fontsize=14)
axs[2].set_title("Marital_Status - counts", pad=10, fontsize=14)

plt.show()
```



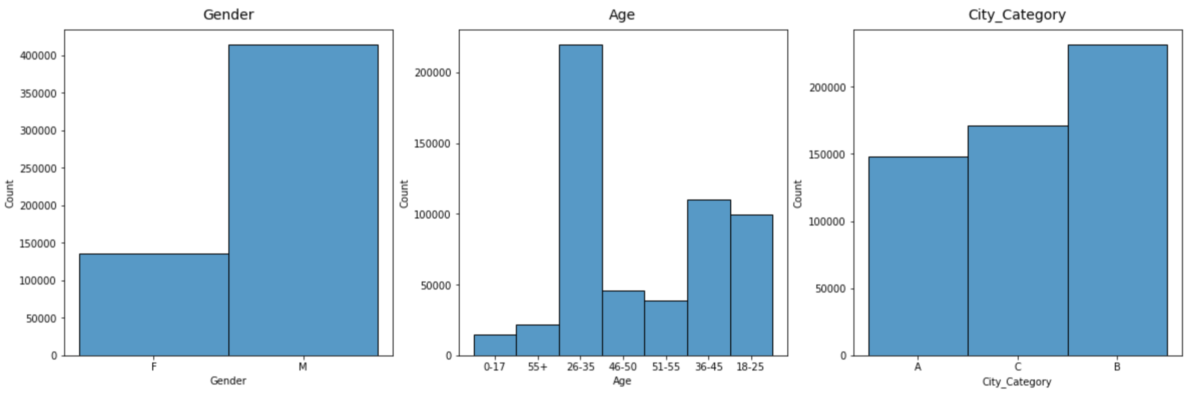
Histplot

In [45]:

```python
fig, axs = plt.subplots(nrows=1, ncols = 3, figsize=(20,6))
sns.histplot(walmart['Gender'],ax = axs[0])
sns.histplot(walmart['Age'],ax = axs[1])
sns.histplot(walmart['City_Category'],ax = axs[2])

axs[0].set_title("Gender", pad=10, fontsize=14)
axs[1].set_title("Age", pad=10, fontsize=14)
axs[2].set_title("City_Category", pad=10, fontsize=14)
```

```
plt.show()
```



Observations

Displot

- More male customers purchased on black friday
- 26-35 Age group customers have made more puchases followed by 36-45 and 18-25
- Accross the cities, more transactions are made in city - B
- Customers who recently moved in to the city have made more purchases

Countplot

- Unmarried customers have made more purchases than married
- Products from category - 5,1 and 8 are pruchased more.
- Customers living in the city below 1 year have made more transactions.

Histplot

- Surprisingly more male customers have made purchases than female.
- Young aged customers have purchased more.

Bivariate Analysis

Countplot

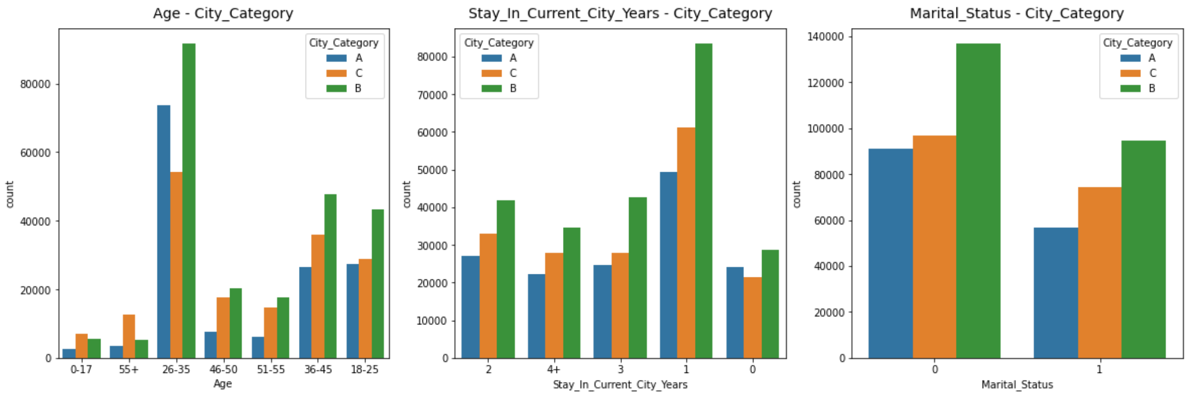
In [46]:
```
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(20, 6))

sns.countplot(data=walmart, x = 'Age', hue='City_Category', ax =axs[0])
sns.countplot(data=walmart, x='Stay_In_Current_City_Years',hue='City_Category',ax=
sns.countplot(data=walmart, x='Marital_Status',hue='City_Category',ax=axs[2])

axs[0].set_title("Age - City_Category", pad=10, fontsize=14)
```

```
axs[1].set_title("Stay_In_Current_City_Years - City_Category", pad=10, fontsize=14
axs[2].set_title("Marital_Status - City_Category", pad=10, fontsize=14)

plt.show()
```



Histplot

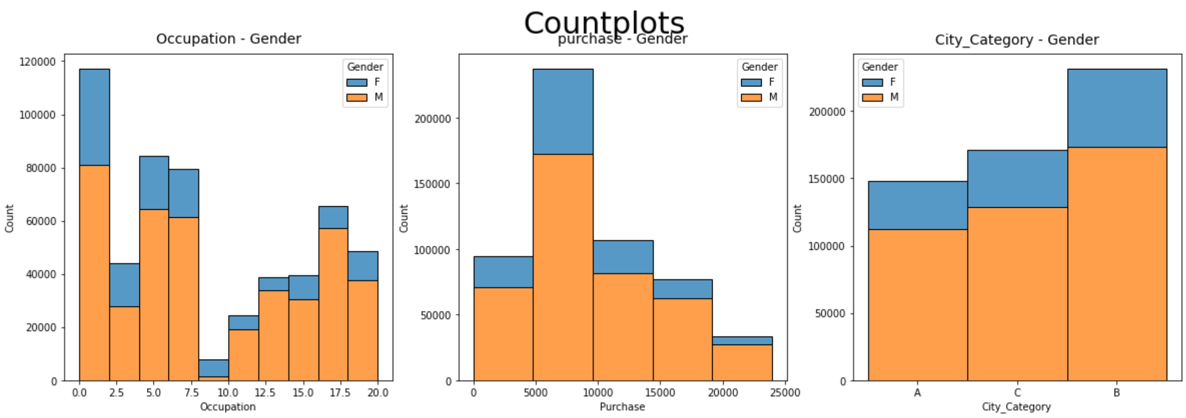
In [47]:

```
fig, axs = plt.subplots(nrows=1, ncols = 3, figsize=(20,6))

sns.histplot(x="Occupation", data = walmart, hue="Gender",bins=10,ax=axs[0],multip
sns.histplot(data = walmart, x='Purchase', hue='Gender',bins=5,ax=axs[1],multiple=
sns.histplot(data = walmart, x='City_Category', hue='Gender',ax=axs[2],multiple='s

axs[0].set_title("Occupation - Gender", pad=10, fontsize=14)
axs[1].set_title("purchase - Gender", pad=10, fontsize=14)
axs[2].set_title("City_Category - Gender", pad=10, fontsize=14)

fig.suptitle("Countplots",fontsize=30)
plt.show()
```



Observations

Countplot

- In all the categories - Age,Marital_status,stay in current city - More transactions/purchases happened in city B

Histplot

- In all cities transactions/purchases made by Male are high than female.

Boxplot

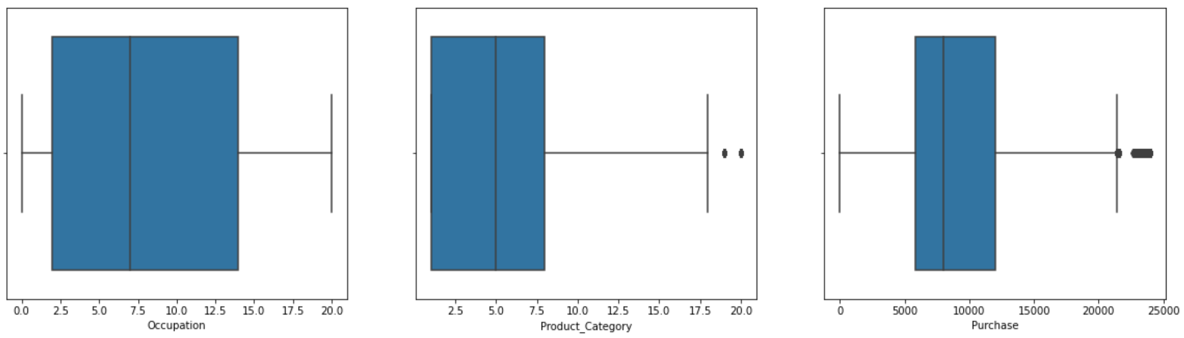
Univariate Analysis

In [48]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(20, 4))
fig.subplots_adjust(top=1.1)


sns.boxplot(data=walmart, x="Occupation", orient='h',ax=axis[0])
sns.boxplot(data=walmart, x="Product_Category", orient='h', ax=axis[1])
sns.boxplot(data=walmart, x="Purchase", orient='h', ax=axis[2])

#axs[0].set_title("Occupation", pad=10, fontsize=14)
#axs[1].set_title("Product_Category", pad=10, fontsize=14)
#axs[2].set_title("Purchase", pad=10, fontsize=14)


plt.show()
```

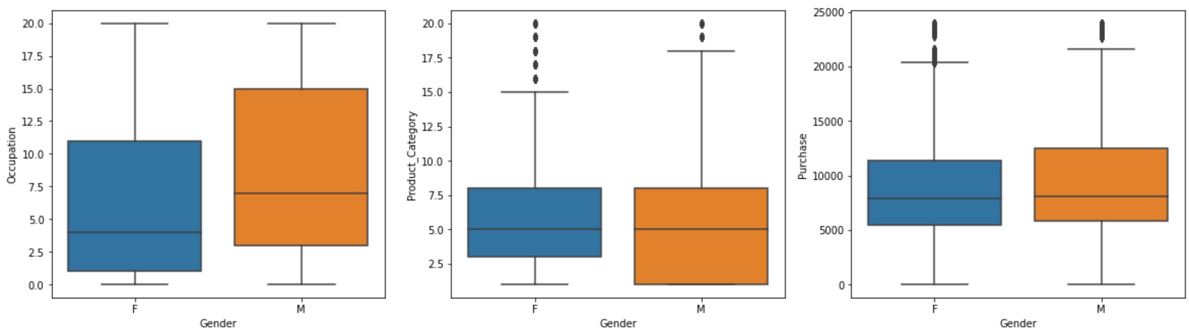


Bivariate Analysis

In [49]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(20, 4))
fig.subplots_adjust(top=1.1)

sns.boxplot(data=walmart, x="Gender", y="Occupation",ax=axis[0])
sns.boxplot(data=walmart, x="Gender", y="Product_Category", ax=axis[1])
sns.boxplot(data=walmart, x="Gender", y="Purchase", ax=axis[2])

plt.show()
```



Observations

Univariate Analysis

- There aren't any outlier values in column - Occupation. Few outliers in product category
- Appears only 1 huge transaction was made by customers than normal.

Bivariate Analysis

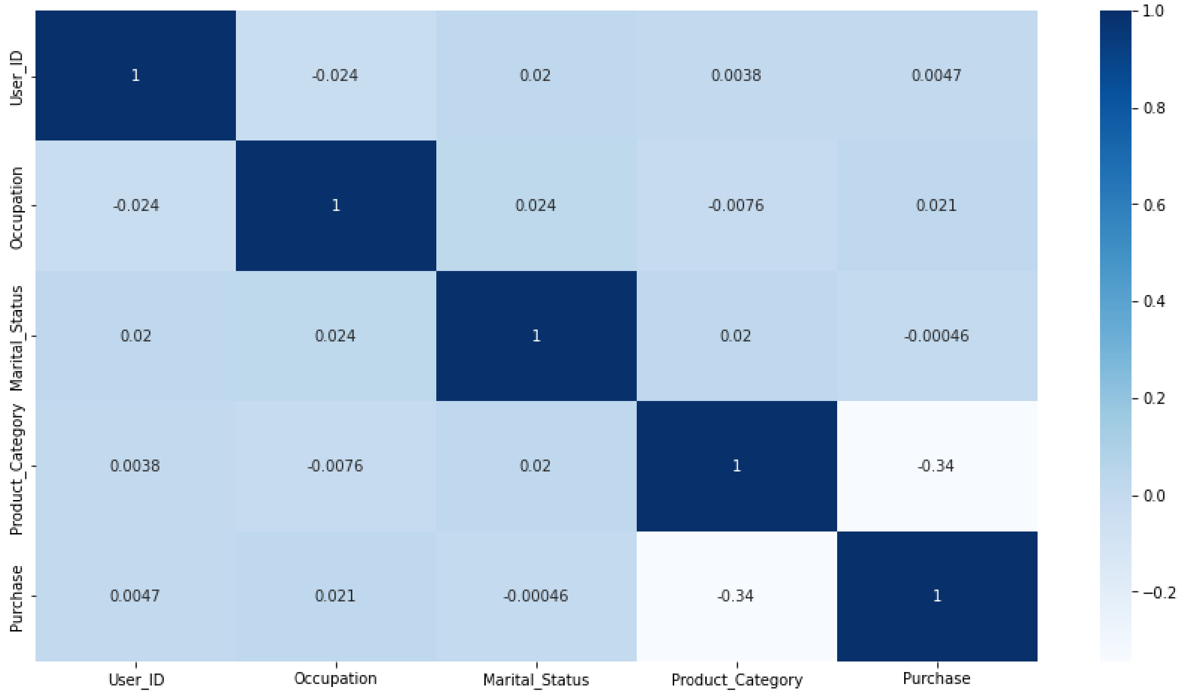
- Only few female customers have purchased products between 16 and 20 category.
- Similarly only products between 18-20 are purchsed by few male customers.
- There are only 2 huge transactions made by female customers and only 1 by male customer.

Outlier Detection

- Only purchase and product category column has outliers.
- Outliers cannot be removed as they contribute to some insights on products purchased and transactions made.

Correlation using heat map

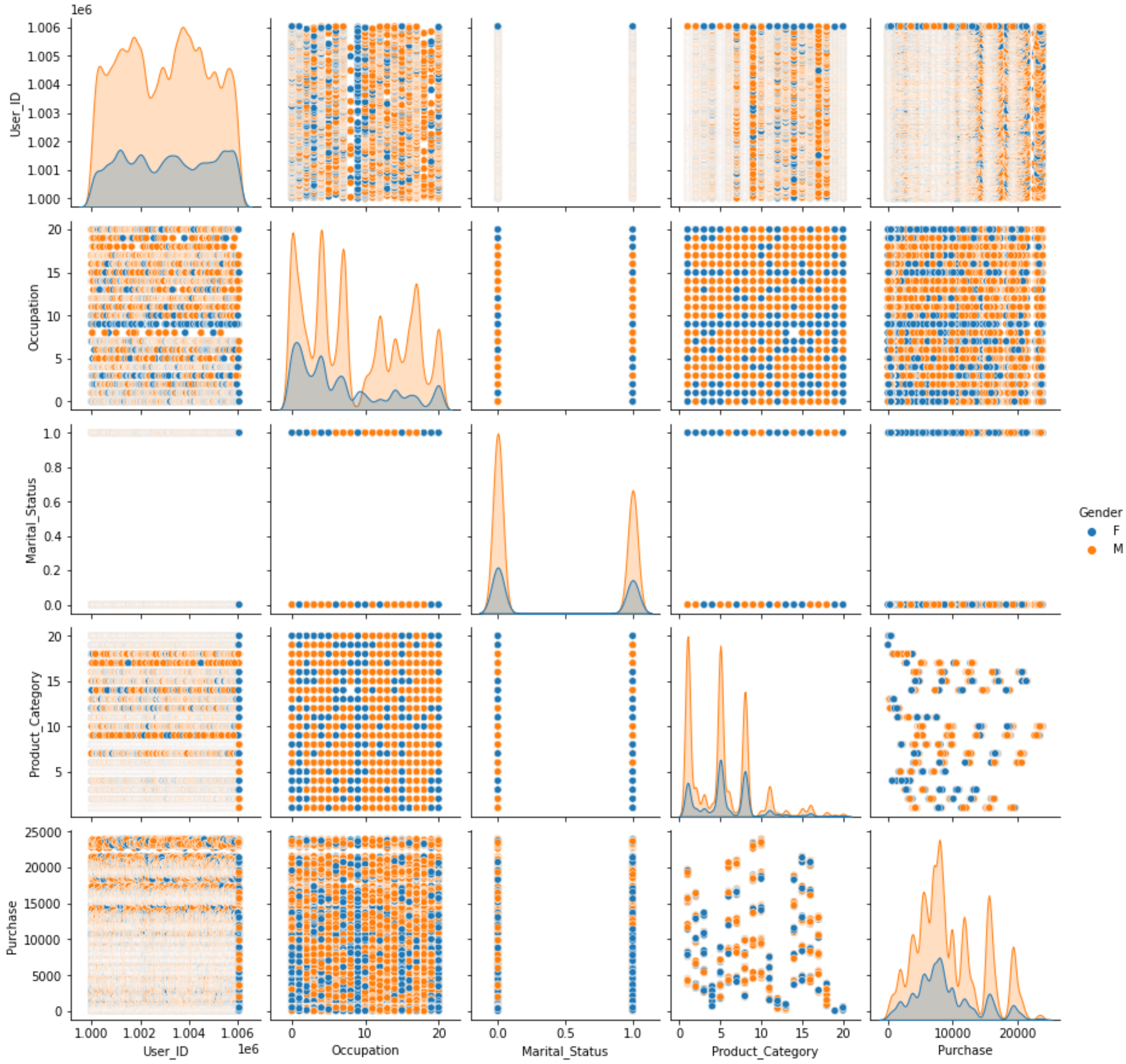
In [50]:
```
plt.figure(figsize =(15,8))
sns.heatmap(walmart.corr(),cmap="Blues",annot=True)
plt.show()
```



Correlation using pairplot

In [51]:
```
plt.figure(figsize =(15,8))
sns.pairplot(data = walmart, hue = 'Gender')
plt.show()
```

```
<Figure size 1080x576 with 0 Axes>
```



Central Limit Theorem

Let us try to compute the sample mean for purchases Made by male and female

Male customers

In [52]:
```
walmart_male=walmart[walmart["Gender"]=="M"]
walmart_male.head()
```

Out[52]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mar
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	

Fetching the samples of Male customers using Bootstrapping

In [53]:
```
n=5000
bootstrapped_walmart_male_purchase = []
for reps in range(10000):
    bootstrapped_samples_male = np.random.choice(walmart_male["Purchase"], size=n)
    bootstrapped_mean_male = np.mean(bootstrapped_samples_male)
    bootstrapped_walmart_male_purchase.append(bootstrapped_mean_male)
```

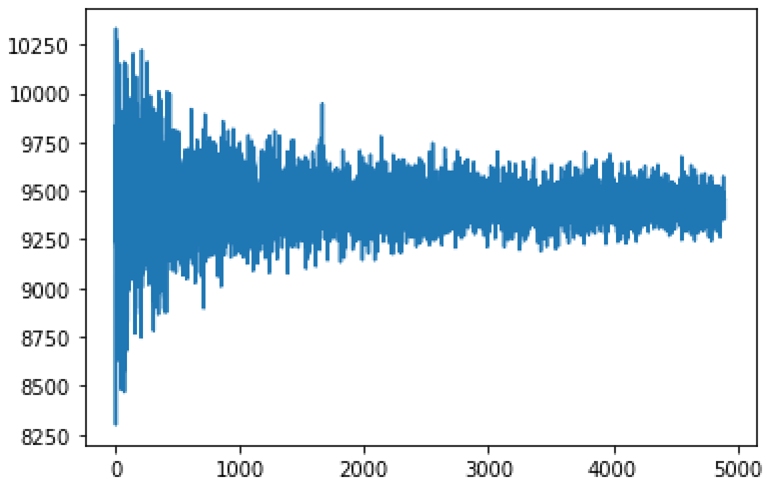
Changing the sample size to observe the distribution of the mean

In [89]:
```
sample_mean_trend_male_purchase = []

for num_samples in range(100, 5000):
    sample_male_purchase = walmart_male["Purchase"].sample(num_samples)
    sample_mean_male = np.mean(sample_male_purchase)
    sample_mean_trend_male_purchase.append(sample_mean_male)

plt.plot(sample_mean_trend_male_purchase)
```

Out[89]: [<matplotlib.lines.Line2D at 0x7f2c95c4ee80>]



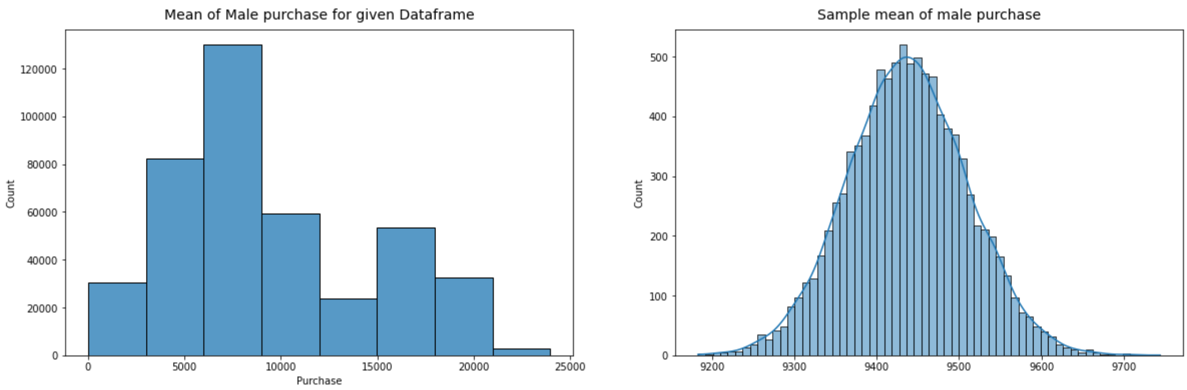
Comparsion of mean through Histplot for Male purchases

In [54]:
```python
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_male['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_male_purchase,kde = True,ax = axs[1])

axs[0].set_title("Mean of Male purchase for given Dataframe", pad=10, fontsize=14)
axs[1].set_title("Sample mean of male purchase", pad=10, fontsize=14)

plt.show()
```



Confidence Interval for purchases made by male

sample mean,standard deviation and standard error

In [55]:
```python
sample_mean_male = np.mean(bootstrapped_walmart_male_purchase)
sample_stddev_male = np.std(bootstrapped_walmart_male_purchase)
se_male = sample_stddev_male / np.sqrt(n)

print("sample_mean_male = ",sample_mean_male)
print("sample_stddev_male =",sample_stddev_male)
print("se_male =",se_male)
```

```
sample_mean_male =  9437.622849340001
sample_stddev_male = 72.7704638212799
se_male = 1.0291297687623469
```

CI

In [56]:
```python
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_male, scale=se_
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_male, scale=se_
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_male, scale=se_
```

```
90% CI -  (9435.93008150725, 9439.315617172753)
95% CI -  (9435.60579205781, 9439.639906622193)
99% CI -  (9434.97198672447, 9440.273711955533)
```

Female customers

In [57]:
```python
walmart_female=walmart[walmart["Gender"]=="F"]
walmart_female.head()
```

Out[57]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
14	1000006	P00231342	F	51-55	9	A	1	

Fetching the samples of female customers using Bootstrapping

In [58]:
```
n=5000
bootstrapped_walmart_female_purchase = []
for reps in range(10000):
    bootstrapped_samples_female = np.random.choice(walmart_female["Purchase"], siz
    bootstrapped_mean_female = np.mean(bootstrapped_samples_female)
    bootstrapped_walmart_female_purchase.append(bootstrapped_mean_female)
```

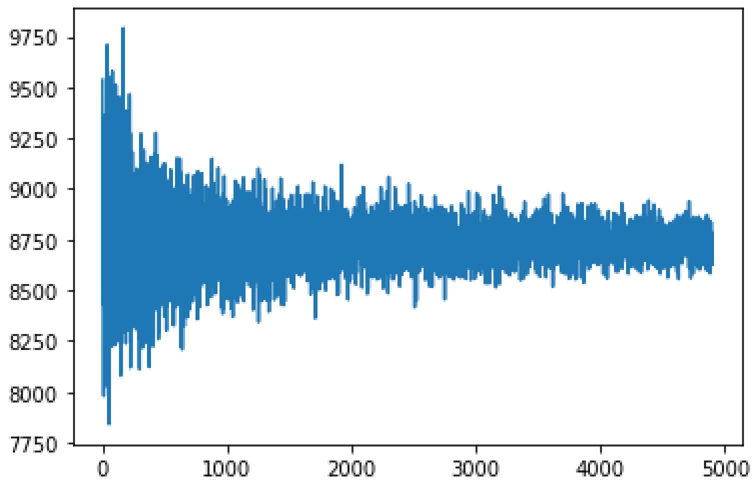
Changing the sample size to observe the distribution of the mean

In [90]:
```
sample_mean_trend_female_purchase = []

for num_samples in range(100, 5000):
    sample_female_purchase = walmart_female["Purchase"].sample(num_samples)
    sample_mean_female = np.mean(sample_female_purchase)
    sample_mean_trend_female_purchase.append(sample_mean_female)

plt.plot(sample_mean_trend_female_purchase)
```

Out[90]: [<matplotlib.lines.Line2D at 0x7f2c95b095e0>]



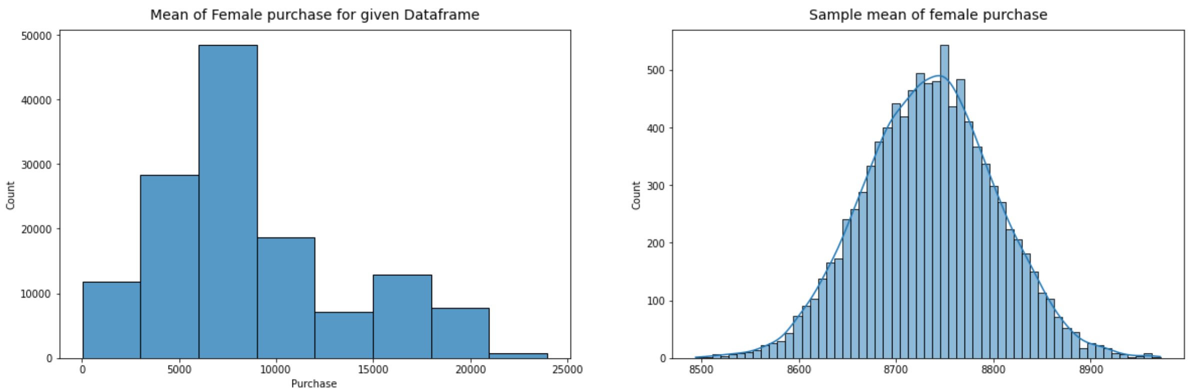
Comparsion of mean through Histplot for Female purchases

In [59]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_female['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_female_purchase,kde = True,ax = axs[1])

axs[0].set_title("Mean of Female purchase for given Dataframe", pad=10, fontsize=14
axs[1].set_title("Sample mean of female purchase", pad=10, fontsize=14)

plt.show()
```



sample mean,standard deviation and standard error

In [61]:
```
sample_mean_female = np.mean(bootstrapped_walmart_female_purchase)
sample_stddev_female = np.std(bootstrapped_walmart_female_purchase)
se_female = sample_stddev_female / np.sqrt(n)

print("sample_mean_female = ",sample_mean_female)
print("sample_stddev_female =",sample_stddev_female)
print("se_female =",se_female)
```

```
sample_mean_female =  8734.56392342
sample_stddev_female = 67.80604621424656
se_female = 0.9589223016708435
```

Confidence Interval for purchases made by female

In [62]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_female, scale=
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_female, scale=
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_female, scale=
```

```
90% CI -  (8732.986636594133, 8736.141210245869)
95% CI -  (8732.684470244754, 8736.443376595247)
99% CI -  (8732.09390325553, 8737.03394358447)
```

Observations

Male customers

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Looking at the graph we can see that it forms normal distribution with mean = 9437.622 and sigma = 72.77
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9200 and 9300 approximately.

Confidence intervals of 50 million male customers average spending

```
*    90% CI -  (9435.9300, 9439.3156)
*    95% CI -  (9435.6057, 9439.6399)
*    99% CI -  (9434.9719, 9440.2737)
```

Female customers

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Looking at the graph we can see that it forms normal distribution with mean = 8734.563 and sigma = 67.80
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 8700 and 8800 approximately.

Confidence intervals of 50 million female customers average spending

```
*    90% CI -  (8732.9866, 8736.1412)
*    95% CI -  (8732.6844, 8736.4433)
*    99% CI -  (8732.0939, 8737.0339)
```

Let us try to compute the sample mean for purchase Column for Married and Unmarried customers

Married customers

In [63]:
```
walmart_married = walmart[walmart["Marital_Status"] == True]
walmart_married.head()
```

Out[63]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	
10	1000005	P00251242	M	26-35	20	A	1	

Fetching the samples of Married customers using Bootstrapping

In [64]:

```
n=5000
bootstrapped_walmart_married_purchase = []
for reps in range(10000):
    bootstrapped_samples_married = np.random.choice(walmart_married["Purchase"], s:
    bootstrapped_mean_married = np.mean(bootstrapped_samples_married)
    bootstrapped_walmart_married_purchase.append(bootstrapped_mean_married)
```

Changing the sample size to observe the distribution of the mean

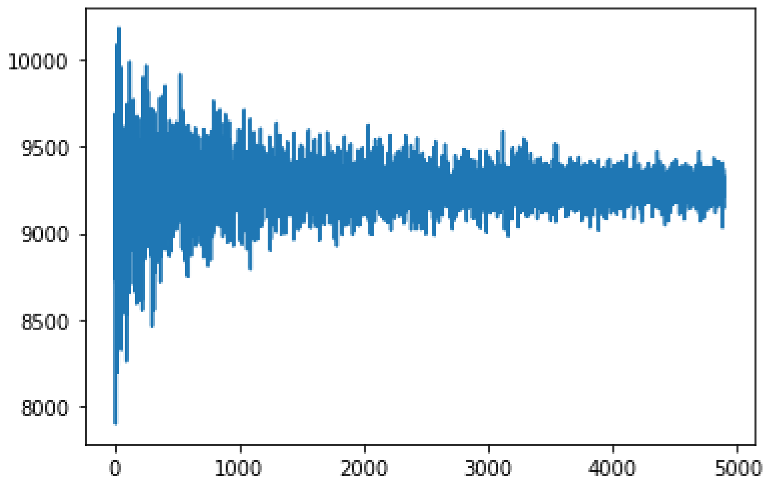
In [91]:

```
sample_mean_trend_married_purchase = []

for num_samples in range(100, 5000):
    sample_married_purchase = walmart_married["Purchase"].sample(num_samples)
    sample_mean_married = np.mean(sample_married_purchase)
    sample_mean_trend_married_purchase.append(sample_mean_married)

plt.plot(sample_mean_trend_married_purchase)
```

Out[91]: [<matplotlib.lines.Line2D at 0x7f2c95a2c3a0>]



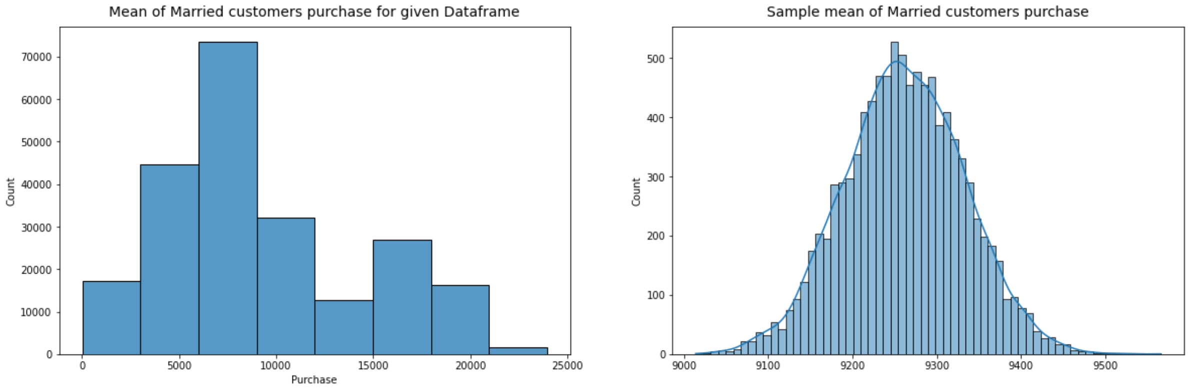
Comparsion of mean purchase through Histplot for Married customers

In [65]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_married['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_married_purchase,kde = True,ax = axs[1])

axs[0].set_title("Mean of Married customers purchase for given Dataframe", pad=10,
axs[1].set_title("Sample mean of Married customers purchase", pad=10, fontsize=14)

plt.show()
```



sample mean,standard deviation and standard error

In [66]:
```
sample_mean_married = np.mean(bootstrapped_walmart_married_purchase)
sample_stddev_married = np.std(bootstrapped_walmart_married_purchase)
se_married = sample_stddev_married / np.sqrt(n)
print("sample_mean_married = ",sample_mean_married)
print("sample_stddev_married =",sample_stddev_married)
print("se_married =",se_married)
```

```
sample_mean_married =  9261.81013716
sample_stddev_married = 71.29112599024837
se_married = 1.008208772522583
```

Confidence Interval for purchases made by Married customers

In [67]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_married, scale
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_married, scale
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_married, scale
```

```
90% CI -  (9260.151781303792, 9263.46849301621)
95% CI -  (9259.83408427696, 9263.786190043042)
99% CI -  (9259.213163459643, 9264.407110860358)
```

Un-Married customers

In [68]:
```
walmart_unmarried = walmart[walmart["Marital_Status"] == True]
walmart_unmarried.head()
```

Out[68]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	
10	1000005	P00251242	M	26-35	20	A	1	

Fetching the samples of unmarried customers using Bootstrapping

In [69]:

```
n=5000
bootstrapped_walmart_unmarried_purchase = []
for reps in range(10000):
    bootstrapped_samples_unmarried = np.random.choice(walmart_unmarried["Purchase"]
    bootstrapped_mean_unmarried = np.mean(bootstrapped_samples_unmarried)
    bootstrapped_walmart_unmarried_purchase.append(bootstrapped_mean_unmarried)
```

Changing the sample size to observe the distribution of the mean

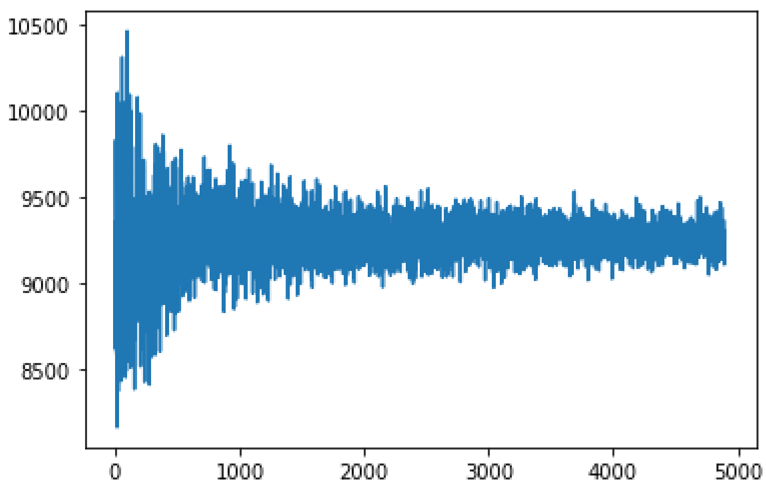
In [92]:

```
sample_mean_trend_unmarried_purchase = []

for num_samples in range(100, 5000):
    sample_unmarried_purchase = walmart_unmarried["Purchase"].sample(num_samples)
    sample_mean_unmarried = np.mean(sample_unmarried_purchase)
    sample_mean_trend_unmarried_purchase.append(sample_mean_unmarried)

plt.plot(sample_mean_trend_unmarried_purchase)
```

Out[92]: [<matplotlib.lines.Line2D at 0x7f2c959de1c0>]



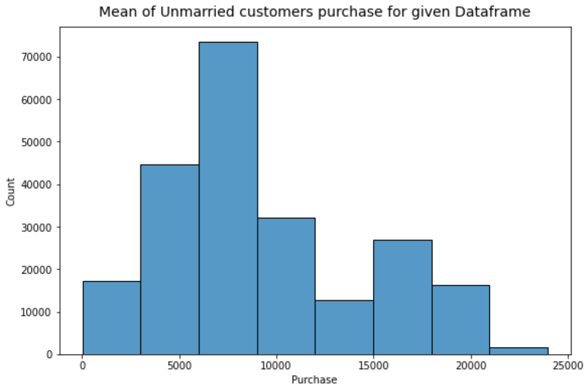
Comparsion of mean purchase through Histplot for unmarried customers

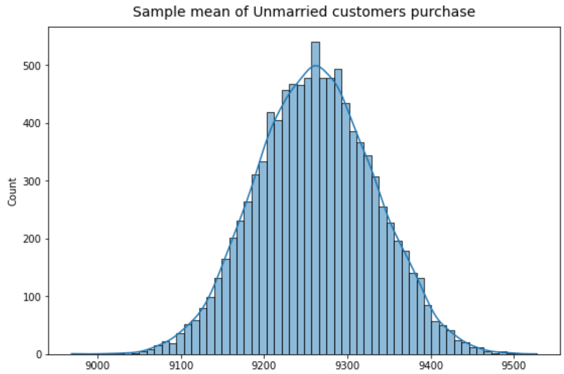
In [70]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_unmarried['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_unmarried_purchase,kde = True,ax = axs[1])

axs[0].set_title("Mean of Unmarried customers purchase for given Dataframe", pad=1(
axs[1].set_title("Sample mean of Unmarried customers purchase", pad=10, fontsize=1

plt.show()
```





sample mean,standard deviation and standard error

In [71]:
```
sample_mean_unmarried = np.mean(bootstrapped_walmart_unmarried_purchase)
sample_stddev_unmarried = np.std(bootstrapped_walmart_unmarried_purchase)
se_unmarried = sample_stddev_unmarried / np.sqrt(n)
print("sample_mean_unmarried = ",sample_mean_unmarried)
print("sample_stddev_unmarried =",sample_stddev_unmarried)
print("se_unmarried =",se_unmarried)
```

```
sample_mean_unmarried =  9261.685469060001
sample_stddev_unmarried = 70.73732974153934
se_unmarried = 1.0003769108654263
```

Confidence Interval for purchases made by unmarried customers

In [72]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_unmarried, sca
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_unmarried, sca
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_unmarried, sca
```

```
90% CI -  (9260.039995469846, 9263.330942650156)
95% CI -  (9259.72476634374, 9263.646171776263)
99% CI -  (9259.1086688984, 9264.262269221603)
```

Observations

Married customers

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Looking at the graph we can see that it forms normal distribution with mean = 9261.810 and sigma = 71.29
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9200 and 9300 approximately.

Confidence intervals of 50 million Married customers average spending

```
*  90% CI -  (9260.1517, 9263.4684)
*  95% CI -  (9259.8340, 9263.7861)
*  99% CI -  (9259.2131, 9264.4071)
```

Unmarried customers

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Looking at the graph we can see that it forms normal distribution with mean = 9261.685 and sigma = 70.73
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9200 and 9300 approximately.

Confidence intervals of 50 million Unmarried customers average spending

```
*  90% CI -  (9260.0399, 9263.3309)
*  95% CI -  (9259.7247, 9263.6461)
*  99% CI -  (9259.1086, 9264.2622)
```

Let us try to compute the sample mean for purchase Column for Different Age groups

In [73]:
```
walmart["Age"].value_counts()
```

Out[73]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

Lets focus on Age groups - (18-25),(26-35),(36-45)

AgeGroup - (18-25)

In [74]:
```
walmart_age1825 = walmart[walmart["Age"] == '18-25']
walmart_age1825.head()
```

Out[74]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
70	1000018	P00366542	F	18-25	3	B	3	
71	1000018	P00190742	F	18-25	3	B	3	
72	1000018	P00151842	F	18-25	3	B	3	
73	1000018	P00112642	F	18-25	3	B	3	
74	1000018	P00118442	F	18-25	3	B	3	

Fetching the samples of customers falling under age group - (18-25) using Bootstrapping

In [75]:

```
n=5000
bootstrapped_walmart_age1825 = []
for reps in range(10000):
    bootstrapped_samples_age1825 = np.random.choice(walmart_age1825["Purchase"], s
    bootstrapped_mean_age1825 = np.mean(bootstrapped_samples_age1825)
    bootstrapped_walmart_age1825.append(bootstrapped_mean_age1825)
```

Changing the sample size to observe the distribution of the mean

In [93]:

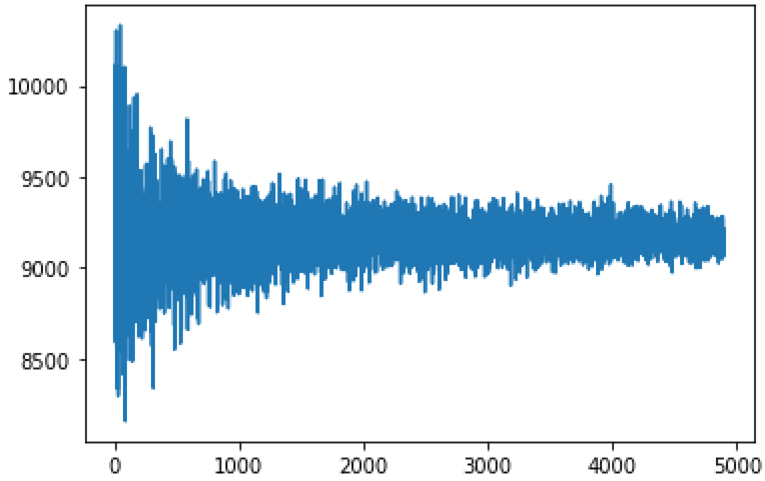
```
sample_mean_trend_age1825 = []

for num_samples in range(100, 5000):
    sample_age1825 = walmart_age1825["Purchase"].sample(num_samples)
    sample_mean_age1825 = np.mean(sample_age1825)
    sample_mean_trend_age1825.append(sample_mean_age1825)

plt.plot(sample_mean_trend_age1825)
```

Out[93]:

```
[<matplotlib.lines.Line2D at 0x7f2c958b5220>]
```



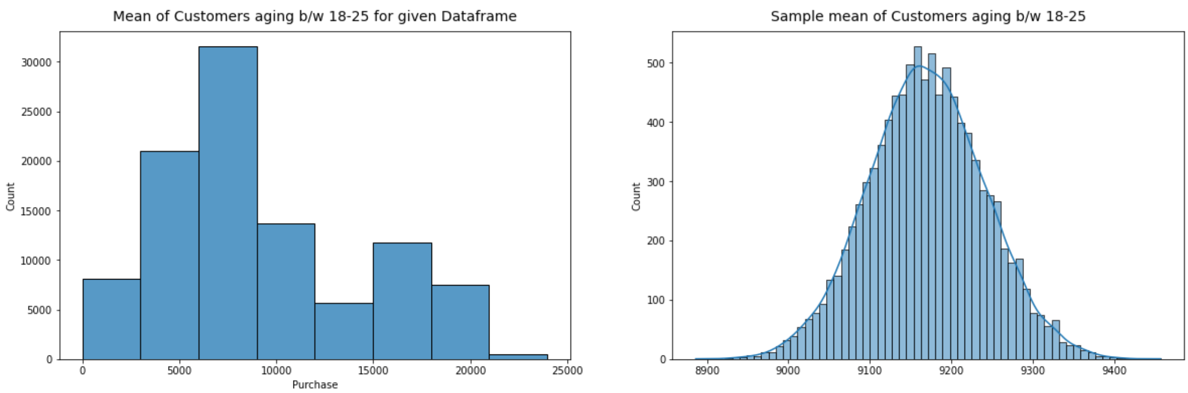
Comparsion of mean through Histplot for 18-25 aged customers

In [76]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_age1825['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_age1825,kde = True,ax = axs[1])

axs[0].set_title("Mean of Customers aging b/w 18-25 for given Dataframe", pad=10,
axs[1].set_title("Sample mean of Customers aging b/w 18-25", pad=10, fontsize=14)

plt.show()
```



sample mean,standard deviation and standard error

In [77]:
```
sample_mean_age1825 = np.mean(bootstrapped_walmart_age1825)
sample_stddev_age1825 = np.std(bootstrapped_walmart_age1825)
se_age1825 = sample_stddev_age1825 / np.sqrt(n)
print("sample_mean_age1825 = ",sample_mean_age1825)
print("sample_stddev_age1825 =",sample_stddev_age1825)
print("se_age1825 =",se_age1825)
```

```
sample_mean_age1825 =  9169.71988158
sample_stddev_age1825 = 71.81323136921232
se_age1825 = 1.0155924576017705
```

Confidence Interval of customers falling under age group - (18-25)

In [78]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_age1825, scale
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_age1825, scale
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_age1825, scale
```

```
90% CI -  (9168.04938064261, 9171.390382517391)
95% CI -  (9167.72935694013, 9171.71040621987)
99% CI -  (9167.103888767246, 9172.335874392755)
```

AgeGroup - (26-35)

In [79]:
```
walmart_age2635 = walmart[walmart["Age"] == '26-35']
walmart_age2635.head()
```

Out[79]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
5	1000003	P00193542	M	26-35	15	A	3	
9	1000005	P00274942	M	26-35	20	A	1	
10	1000005	P00251242	M	26-35	20	A	1	
11	1000005	P00014542	M	26-35	20	A	1	
12	1000005	P00031342	M	26-35	20	A	1	

Fetching the samples of customers falling under age group - (26-35) using Bootstrapping

In [80]:
```
n=5000
bootstrapped_walmart_age2635 = []
for reps in range(10000):
    bootstrapped_samples_age2635 = np.random.choice(walmart_age2635["Purchase"], s
    bootstrapped_mean_age2635 = np.mean(bootstrapped_samples_age2635)
    bootstrapped_walmart_age2635.append(bootstrapped_mean_age2635)
```

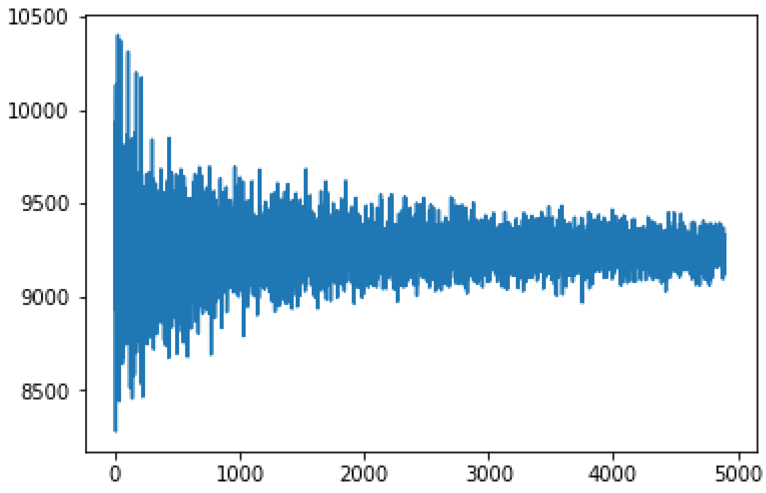
Changing the sample size to observe the distribution of the mean

In [94]:
```
sample_mean_trend_age2635 = []

for num_samples in range(100, 5000):
    sample_age2635 = walmart_age2635["Purchase"].sample(num_samples)
    sample_mean_age2635 = np.mean(sample_age2635)
    sample_mean_trend_age2635.append(sample_mean_age2635)

plt.plot(sample_mean_trend_age2635)
```

Out[94]:
```
[<matplotlib.lines.Line2D at 0x7f2c9583bcd0>]
```



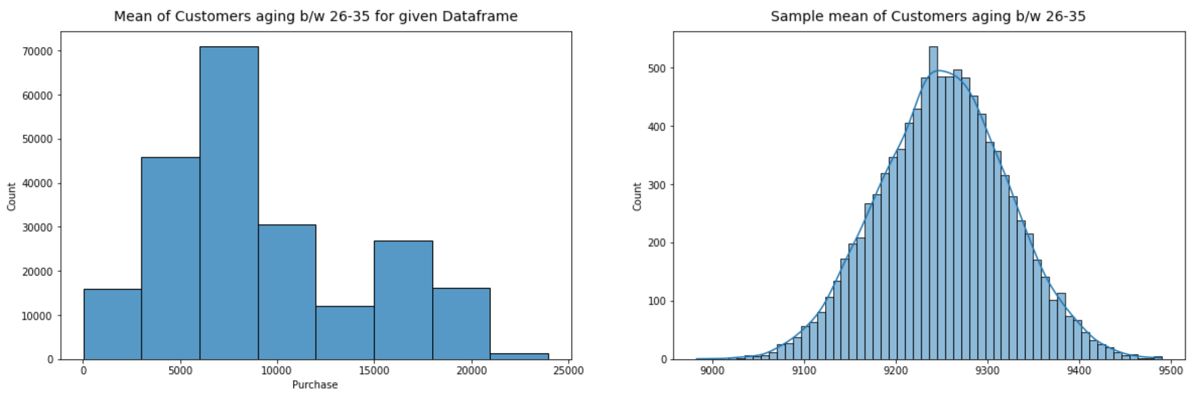
Comparsion of mean through Histplot for 26-35 aged customers

In [81]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_age2635['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_age2635,kde = True,ax = axs[1])

axs[0].set_title("Mean of Customers aging b/w 26-35 for given Dataframe", pad=10, 
axs[1].set_title("Sample mean of Customers aging b/w 26-35", pad=10, fontsize=14)

plt.show()
```



sample mean,standard deviation and standard error

In [82]:
```
sample_mean_age2635 = np.mean(bootstrapped_walmart_age2635)
sample_stddev_age2635 = np.std(bootstrapped_walmart_age2635)
se_age2635 = sample_stddev_age2635 / np.sqrt(n)
print("sample_mean_age2635 = ",sample_mean_age2635)
print("sample_stddev_age2635 =",sample_stddev_age2635)
print("se_age2635 =",se_age2635)
```

```
sample_mean_age2635 =  9252.49582042
sample_stddev_age2635 = 70.42227148660999
se_age2635 = 0.9959213142948394
```

Confidence Interval of customers falling under age group - (26-35)

In [83]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_age2635, scale
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_age2635, scale
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_age2635, scale
```

```
90% CI -  (9250.857675634023, 9254.133965205976)
95% CI -  (9250.543850512546, 9254.447790327453)
99% CI -  (9249.93049711461, 9255.061143725388)
```

AgeGroup - (36-45)

In [84]:
```
walmart_age3645 = walmart[walmart["Age"] == '36-45']
walmart_age3645.head()
```

Out[84]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
18	1000007	P00036842	M	36-45	1	B	1	
29	1000010	P00085942	F	36-45	1	B	4+	
30	1000010	P00118742	F	36-45	1	B	4+	
31	1000010	P00297942	F	36-45	1	B	4+	
32	1000010	P00266842	F	36-45	1	B	4+	

Fetching the samples of customers falling under age group - (36-45) using Bootstrapping

In [85]:
```
n=5000
bootstrapped_walmart_age3645 = []
for reps in range(10000):
    bootstrapped_samples_age3645 = np.random.choice(walmart_age3645["Purchase"], s
    bootstrapped_mean_age3645 = np.mean(bootstrapped_samples_age3645)
    bootstrapped_walmart_age3645.append(bootstrapped_mean_age3645)
```

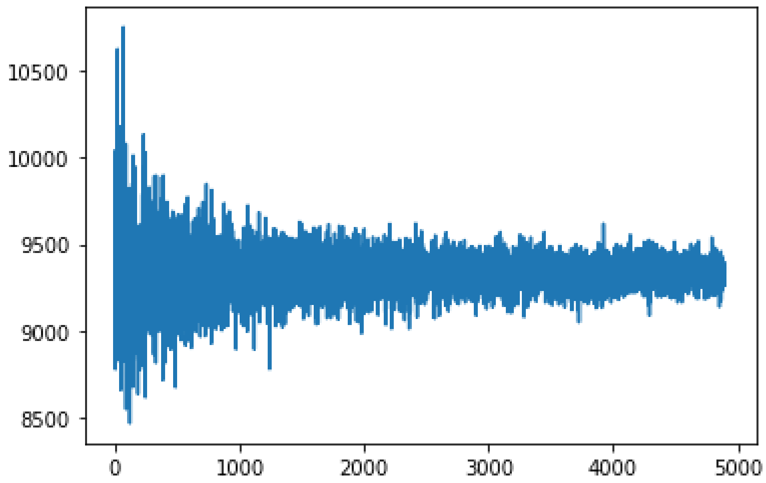
Changing the sample size to observe the distribution of the mean

In [95]:
```
sample_mean_trend_age3645 = []

for num_samples in range(100, 5000):
    sample_age3645 = walmart_age3645["Purchase"].sample(num_samples)
    sample_mean_age3645 = np.mean(sample_age3645)
    sample_mean_trend_age3645.append(sample_mean_age3645)

plt.plot(sample_mean_trend_age3645)
```

Out[95]:
```
[<matplotlib.lines.Line2D at 0x7f2c9575ab80>]
```



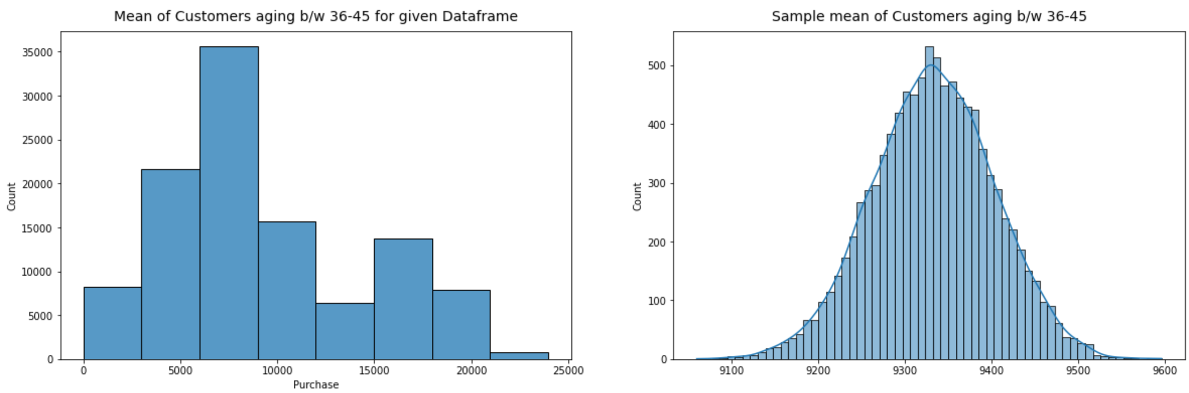
Comparsion of mean through Histplot for 36-45 aged customers

In [86]:
```
fig, axs = plt.subplots(nrows=1, ncols = 2, figsize=(20,6))

sns.histplot(walmart_age3645['Purchase'],bins= 8, ax = axs[0])
sns.histplot(bootstrapped_walmart_age3645,kde = True,ax = axs[1])

axs[0].set_title("Mean of Customers aging b/w 36-45 for given Dataframe", pad=10, 
axs[1].set_title("Sample mean of Customers aging b/w 36-45", pad=10, fontsize=14)

plt.show()
```



sample mean,standard deviation and standard error

In [87]:
```
sample_mean_age3645 = np.mean(bootstrapped_walmart_age3645)
sample_stddev_age3645 = np.std(bootstrapped_walmart_age3645)
se_age3645 = sample_stddev_age3645 / np.sqrt(n)
print("sample_mean_age3645 = ",sample_mean_age3645)
print("sample_stddev_age3645 =",sample_stddev_age3645)
print("se_age3645 =",se_age3645)
```

```
sample_mean_age3645 =  9333.015967899999
sample_stddev_age3645 = 70.48137839809665
se_age3645 = 0.9967572122533837
```

Confidence Interval of customers falling under age group - (36-45)

In [88]:
```
print("90% CI - ",st.norm.interval(confidence=0.90, loc=sample_mean_age3645, scale
print("95% CI - ",st.norm.interval(confidence=0.95, loc=sample_mean_age3645, scale
print("99% CI - ",st.norm.interval(confidence=0.99, loc=sample_mean_age3645, scale
```

```
90% CI -  (9331.376448184234, 9334.655487615764)
95% CI -  (9331.062359662652, 9334.969576137346)
99% CI -  (9330.448491464153, 9335.583444335845)
```

Observations

Agegroup - (18-25)

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9100 and 9200 approximately.
- Looking at the graph we can see that it forms normal distribution with mean = 9169.71 and sigma = 71.81

Confidence intervals of 50 million customers falling under age group - (18-25) average spending

```
*   90% CI -  (9168.0493, 9171.3903)
*   95% CI -  (9167.7293, 9171.7104)
*   99% CI -  (9167.1038, 9172.3358)
```

==

Agegroup -(26-35)

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9200 and 9300 approximately.
- Looking at the graph we can see that it forms normal distribution with mean = 9252.495 and sigma = 70.42

Confidence intervals of 50 million customers falling under age group - (26-35) average spending

```
*  90% CI -  (9250.8576, 9254.1339)
*  95% CI -  (9250.5438, 9254.4477)
*  99% CI -  (9249.9304, 9255.0611)
```

==

Agegroup - (36-45)

- A 10,000 samples of size 5k each has been taken using bootstrapping to calculate the sample mean.
- Changing the sample size from 500 to 5000, distribution of mean has been observed resulting mean between 9300 and 9400 approximately.
- Looking at the graph we can see that it forms normal distribution with mean = 9333.015 and sigma = 70.48

Confidence intervals of 50 million customers falling under age group - (36-45) average spending

```
*   90% CI -  (9331.3764, 9334.6554)
*   95% CI -  (9331.0623, 9334.9695)
*   99% CI -  (9330.4484, 9335.5834)
```

Business Insights / Observations:

There are no missing values in the data

- There are 3 Types of category presents in the data such as A,B & C. 50% of users purchase amount is around 8500 range.
- Out of 550068 data points, there are 5891 unique customers purchased the products on Black friday sale.
- Peoples are purchasing more who are stayed one and two years. More than 4+ years who lived in city purchasing very less.
- Women are not spending more money than male customers. May be products might be less or no attractive offers
- There are outliers when comparing category type to purchase. Gender and Occupation correlate the data.
- Age group from 26-35 followed by 36-45 and 18-25 have made more spending

Are women spending more money per transaction than men.....? No. Men are spending more money than women.

Recommendations:-

- We need to observe the customers and provide some benifits who stayed long in the city and we need to do more analysis on them.
- There are more customers choosing maximum value of transaction is around 9000. Put some special benifits or gifts to attract the customers.
- Customers who are staying from long time in city are spending less. Company should focus on this aspect.
- Married and unmarried customers are spending money equally and Confidence intervals of married and unmarried customers spending is overlapping. Company should focus on acquisition of married customers.
- The tier-2 city called B has the highest population, management should open more outlets in tier-1 and tier-2 cities like A and C inoder to increase business.

Confidence intervals of male and female spending is not overlapping because female customer have spent less amount compared to males. Management should focus on Women products and should release attractive offers.

Confidence intervals of different age group customers spending is not overlapping as only 18-25 age group customers have made more spending. Company should try to get more products which can be buyed from other age group customers.