Business Case Study: Multinational Retail Corporation

Context:

This business case focuses on the operations of an American multinational retail corporation that operates a chain of supercentres, discount departmental stores, and grocery stores from the United States, that has millions of customers worldwide. This case study aims to analyze the customer purchase behaviour (specifically, purchase amount) against the customer's gender and the various other factors using 5,00,00 transactions made at this store, provide data driven insights and actionable business recommendations to help the business target specific Customer base with more interesting products.

This case study report contains the solutions to the problem statements (as Python queries by employing data visualisation, descriptive Statistics & Probability), sample output of the queries, followed by insights and recommendations. As part of the confidentiality agreement, the name of the brand, the actual dataset and problem statements are not included in this report.

Google Colab Notebook-Python File - This Python project involves exploratory data analysis of a dataset from this retail corporation. The code is importing necessary libraries such as numpy, pandas, seaborn, scipy.stats and matplotlib.

****

Importing libraries

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

Loading Data

```
data = pd.read_csv("RetailCorporation.csv")
data
```

	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
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

550068 rows × 10 columns

Shape of the dataset and Column DataTypes

```
data.shape
data.info()
```

```
data.shape
```

```
(550068, 10)
```

```
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
```

Insights: There are a total of 550068 rows (data points) and 10 columns. Following columns have integer datatype – User ID, Occupation, Marital_Status, Product Category and Purchase. Remaining columns such as Product_ID, Gender, Age, City_category, Stay_in_Current_City_Years have object datatype.

Unique Value counts for each columns

```
data.nunique()
```

```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category                20
Purchase                     18105
dtype: int64
```

Insights: There are a total of **5,50,068** datapoints, out of which only **5891** User_IDs are unique. This means same users have bought multiple products or have performed multiple transactions.

Null/Missing Values and Duplicate Values Detection

```
data.isna().sum()
data.duplicated().sum()
```

```
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 or duplicate values in the dataset

Updating 'Marital_Status' column

```
data['Marital_Status'] = data['Marital_Status'].apply(lambda x: "Married" if x == 1 else "Single")
```

	Marital_Status
0	Single
1	Single
2	Single
3	Single
4	Single
...	...

550063	Married
550064	Single
550065	Married
550066	Single
550067	Married

550068 rows × 1 columns

Conversion of Column datatype to Category for Memory Optimisation

```
category_col = ["Gender", "Age", "City_Category", "Marital_Status",
"Stay_In_Current_City_Years"]

for i in category_col:
    data[i] = data[i].astype("category")

data.dtypes
```

```
User_ID                        int64
Product_ID                    object
Gender                      category
Age                         category
Occupation                     int64
City_Category               category
Stay_In_Current_City_Years  category
Marital_Status              category
Product_Category               int64
Purchase                       int64
dtype: object
```

Unique Values from each column

```
columns = ['Product_ID','Gender', 'Age', 'Occupation','City_Category',
'Marital_Status', 'Product_Category', 'Stay_In_Current_City_Years']
for i in columns:
  print(f"{i}   -> {len(data[i].unique())}, {data[i].unique()}")
```

```
Product_ID   -> 3631, ['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']
Gender   -> 2, ['F', 'M']
Categories (2, object): ['F', 'M']
Age   -> 7, ['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
Occupation   -> 21, [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]
City_Category   -> 3, ['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']
Marital_Status   -> 2, ['Single', 'Married']
Categories (2, object): ['Married', 'Single']
Product_Category   -> 20, [ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]
Stay_In_Current_City_Years   -> 5, ['2', '4+', '3', '1', '0']
Categories (5, object): ['0', '1', '2', '3', '4+']
```

Insights:

- There are 3631 unique products that customers purchased.
- There are 7 age-groups/categories
- There are 3 city categories
- There are 2 categories of Marital Status – Single and Married

Detect Outliers (using boxplot, "describe" method by checking the difference between mean and median)

```
data.describe()
```

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

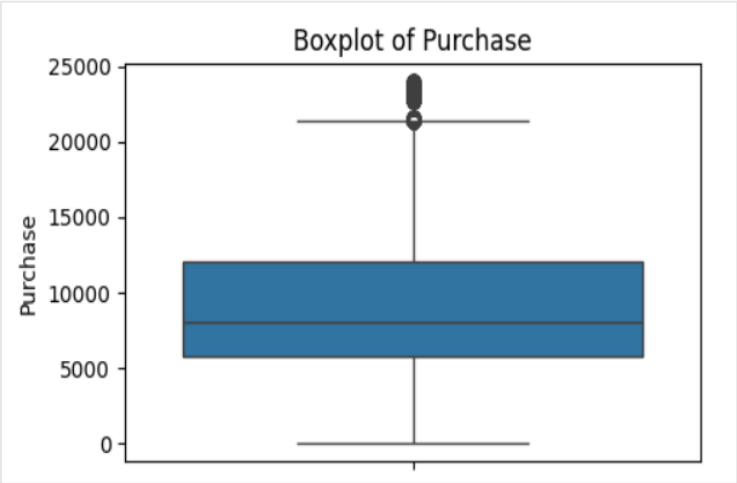
We will be focusing on Purchase column here for identifying outliers as this is a continuous variable. Get the difference between mean and median for purchase column to identify outliers:

```
data.describe().loc["mean", "Purchase"] - data.describe().loc["50%", "Purchase"]
```

Insights: Difference between mean and median of Purchase column is **1216.968**. There is significant difference in Mean & Median of Purchase Amounts indicating presence of outliers. The mean is higher than the median indicating right skewness (positive skew). IQR is 6231.

Identifying outliers using Boxplots

```
plt.figure(figsize=(5,3))
sns.boxplot(y=data["Purchase"])
plt.title(f'Boxplot of Purchase')
plt.tight_layout()
plt.show()
```



As shown on the plot, the Purchase column has some outliers above the upper_whisker value.

```
Q1 = data["Purchase"].quantile(0.25)
Q3 = data["Purchase"].quantile(0.75)
IQR = Q3 - Q1
lower_whisker = Q1 - 1.5 * IQR
upper_whisker = Q3 + 1.5 * IQR
```

Insights:

- First Quartile (q1) = 5823.0
- Third Quartile (q2) = 12054.0
- IQR = 6231.0
- The values above the upper whisker (Q3+ 1.5*IQR) and below the lower whisker (Q1 - 1.5*IQR) are considered outliers.

Now, clip the data between the 5 percentile and 95 percentile by retaining all rows. This allows to set lower and upper bounds for the values in the DataFrame. i.e. it sets the values that are below the 5th percentile to the 5th percentile value, and those above the 95th percentile to the 95th percentile value.

```
def clip_outliers(data):
  clipped_data = data.copy()
  lower_bound = round(data["Purchase" ].quantile(0.05))
  upper_bound = round(data["Purchase" ].quantile(0.95))
  clipped_data["Purchase" ]=data["Purchase" ].clip(lower =lower_bound,upper =
upper_bound)
  return clipped_data

clipped_data = clip_outliers(data)
print("Clipped DataFrame:")
clipped_data
```

Results:

Clipped DataFrame:

	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	1984
3	1000001	P00085442	F	0-17	10	A	2	Single	12	1984
4	1000002	P00285442	M	55+	16	C	4+	Single	8	7969
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	Married	20	1984
550064	1006035	P00375436	F	26-35	1	C	3	Single	20	1984
550065	1006036	P00375436	F	26-35	15	B	4+	Married	20	1984
550066	1006038	P00375436	F	55+	1	C	2	Single	20	1984
550067	1006039	P00371644	F	46-50	0	B	4+	Married	20	1984

550068 rows × 10 columns

Note: We will use this clipped_data for all further analysis

Statistical Analysis of Categorical/Object Columns

```
data.describe(include = ["object", "category"])
```

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

Insights:

- The most frequently occurred Product_ID is P00265242. This value occurs 1,880 times in the dataset.
- The most frequently occurred gender is Male. It appears 414,259 times in the dataset.
- The most frequently occurred Age group is 26-35.

Non-Graphical Analysis - Value_counts and Unique attributes

Distribution of female and male customers

```
clipped_data.groupby(["Gender"])["User_ID"].nunique()
```

	Gender	User_ID
0	F	1666
1	M	4225

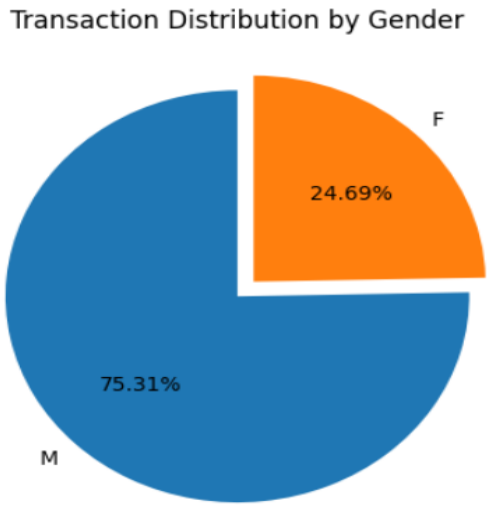
Insights: There are 5891 unique User_IDs in the dataset. Out of this, 1666 are females. This account for 28.28% of total. There are 4225 males and this account for 71.71% of total unique customers.

Total number of transactions made by each gender

```
np.round(clipped_data['Gender'].value_counts(normalize=True)*100,2).reset_index()
```

Gender	count
M	414259
F	135809

Insights: 75% of transactions are done by Male customers, which is 4,14,259/5,50,068 transactions. Females have done 24.6% of transactions.



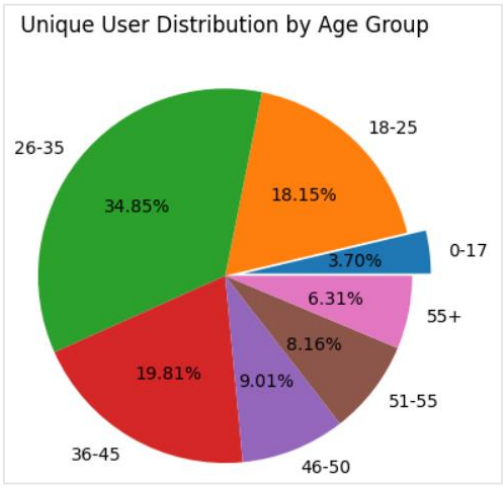
```
print('Average number of transactions made by each Male on Black Friday
is',round(414259/4225))
print('Average number of transactions made by each Female on Black Friday
is',round(135809/1666))
```

Insights: Average number of transactions made by each Male on Black Friday is 98.
Average number of transactions made by each Female on Black Friday is 82

Share of Unique Customers based on their age group

```
Unique_customer_by_age_group = pd.DataFrame(clipped_data.groupby(by=['Age'])
['User_ID'].nunique()).reset_index().rename(columns= {'User_ID':'Unique_Customer'})
Unique_customer_by_age_group
```

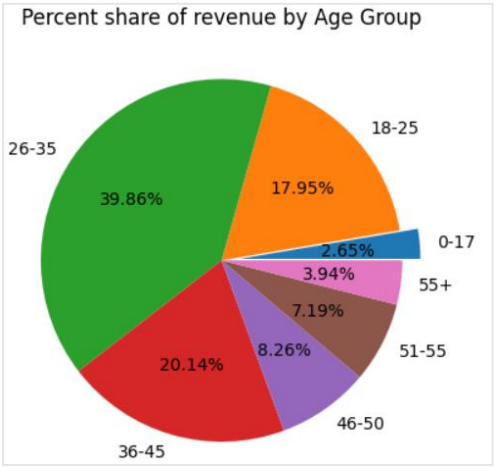
	Age	Unique_Customers
0	0-17	218
1	18-25	1069
2	26-35	2053
3	36-45	1167
4	46-50	531
5	51-55	481
6	55+	372

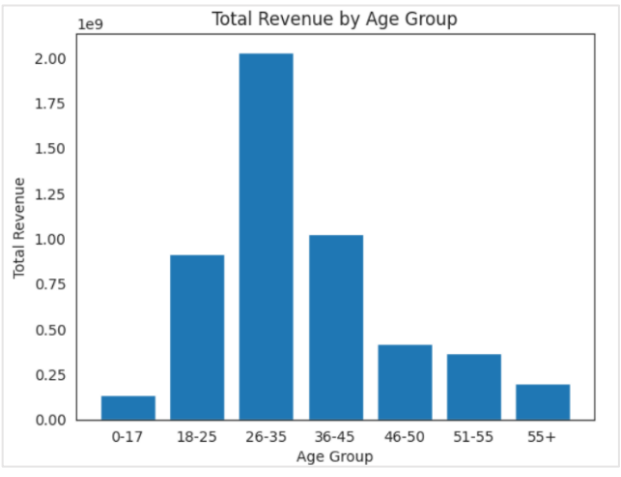


Insights: The highest proportion of customers are from 26-35 age-group. 72% of total customers are between 18 and 45 years of age.

Percent share of revenue generated from each age group

```
x = clipped_data.groupby('Age')['Purchase'].sum()
y = clipped_data.groupby('Age')['Purchase'].sum().index

plt.bar(x=y,height=x)
plt.title("Total Revenue by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Total Revenue")
plt.show()
```





Proportion of transactions done by customers from across different occupations

```
np.round(clipped_data['Occupation'].value_counts(normalize=True)*100,2).cumsum().reset_index()
```

	Occupation	proportion
0	4	13.15
1	0	25.81
2	7	36.56
3	1	45.18
4	17	52.46
5	20	58.56
6	12	64.23
7	14	69.19
8	2	74.02
9	16	78.63
10	6	82.33
11	3	85.54
12	10	87.89
13	5	90.10
14	15	92.31
15	11	94.42
16	19	95.96
17	13	97.36
18	18	98.56
19	9	99.70
20	8	99.98

Insights: 82.33 % of the total transactions are made by the customers belonging to 11 occupations. These include 4, 0, 7, 1, 17, 20, 12, 14, 2, 16, 6 (sorted in descending order of the total transactions' share.)

Proportion of transactions done by customers from across different 'Stay_In_Current_City_Years'

```
np.round(clipped_data['Stay_In_Current_City_Years'].value_counts(normalize = True) * 100, 2)
```

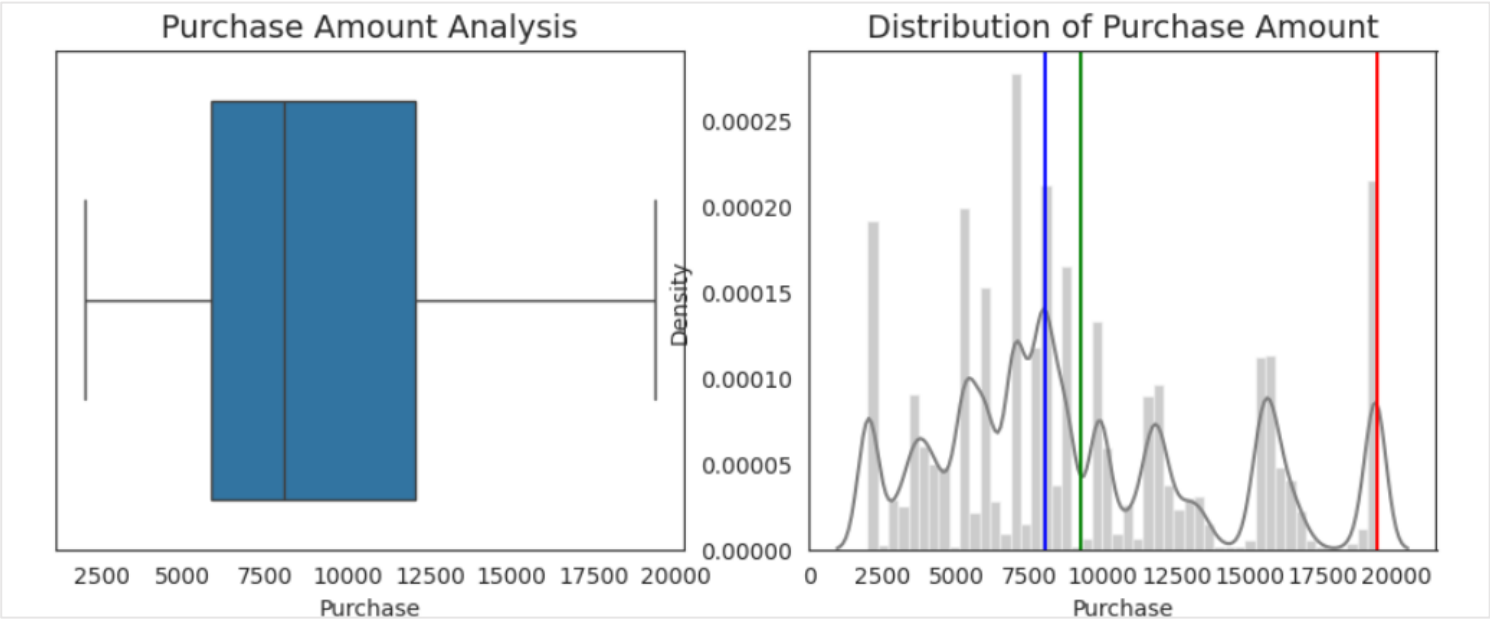
```
1     35.24
2     18.51
3     17.32
4+    15.40
0     13.53
Name: Stay_In_Current_City_Years, dtype: float64
```

Insights: Majority of the transactions (53.75 % of total transactions) are made by the customers having 1 or2 years of stay in the current city.

Graphical Analysis

Univariate Continous Variable Analysis

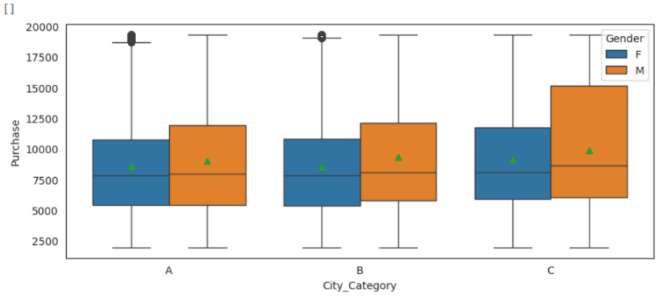
```
fig = plt.figure(figsize=(11,4))
sns.set_style("white")

plt.subplot(1, 2, 1)
sns.boxplot(data = clipped_data, x = "Purchase", orient = "h")

plt.title('Purchase Amount Analysis', fontsize = '14')

plt.subplot(1, 2, 2)
sns.distplot(a = clipped_data["Purchase"], color = 'gray')
plt.title("Distribution of Purchase Amount", fontsize = '14')
plt.axvline(clipped_data["Purchase"].mean(),color="g")
plt.axvline(clipped_data["Purchase"].median(),color="b")
plt.axvline(clipped_data["Purchase"].mode()[0],color="r")

plt.show()
```



Insights: Majority of customer purchase within 5,000 - 20,000 range. The distribution of purchase amount from density plot shows that the distribution is right skewed, which means majority of data is concentrated on left side.

Bivariate Analysis

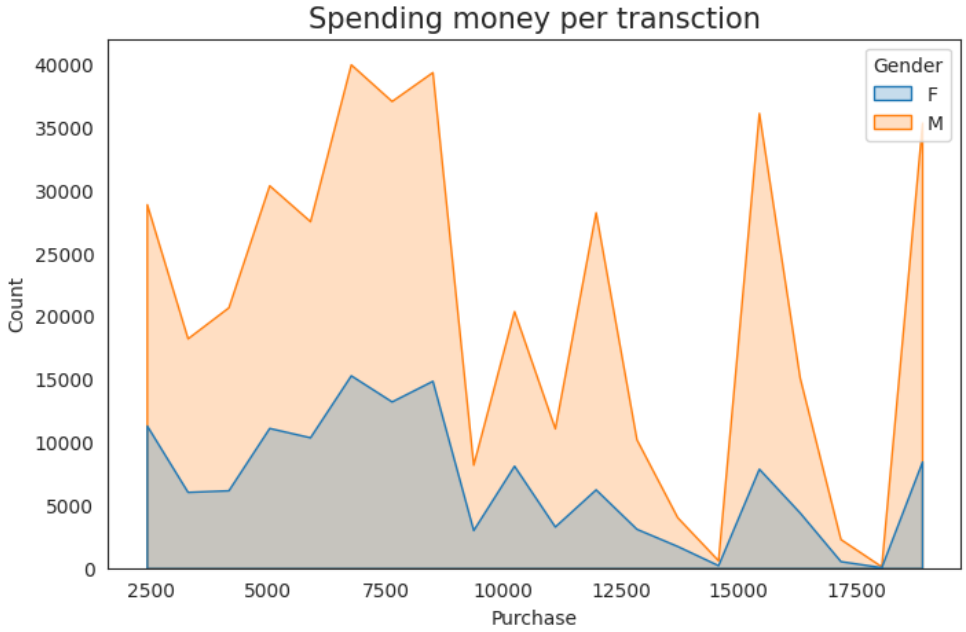
Purchasing behaviour across cities



Insights: In this box plot, we are analyzing the purchase patterns across three cities (A, B, and C) and comparing them between genders (male and female) for each city. There is greater variability in spending among the customers in City C as the middle 50% of purchase values for City C are more spread out. In City C, the box plot for male customers is positioned higher than that for female customers, suggesting that male customers tend to make higher purchases compared to female customers in this city.

Are women spending more money per transaction than men?

```
plt.title("Spending money per transction", fontsize = 15)
sns.histplot(data=clipped_data, x = "Purchase", bins=20, hue = "Gender",
element="poly")

plt.show()
```



Insights: The histplot shows that the amount of money spent by female customers per transaction is relatively less when compared to that of men. Some of the factors that contributed to this difference could be socio-economic status, salary range etc.

How does gender affect the amount spent?

Calculating CI of 95% using Central Limit Theorem (CLT) for Puchases made by male and female

Let's first filter purchases by gender

```
data_male_customers = clipped_data.loc[clipped_data["Gender"] == "M"][["Gender",
"User_ID","Purchase"]]
data_female_customers = clipped_data.loc[clipped_data["Gender"] == "F"][["Gender",
"User_ID","Purchase"]]
```

First, compute the 95% confidence interval for the entire dataset:

For Male Customers:

```
pop_mean = data_male_customers["Purchase"].mean()
pop_std = data_male_customers["Purchase"].std(ddof=1)
n = len(data_male_customers)
se = pop_std / np.sqrt(n)
z_score = 1.96  # Z-score for 95% confidence
margin_of_error = z_score * se

# Calculate the confidence interval bounds
lower_bound = pop_mean - margin_of_error
upper_bound = pop_mean + margin_of_error
print(f"Population Mean -> {pop_mean:.2f}")
print(f"Standard Error -> {se:.2f}")
print(f"total data size - > {n}")
print(f"95% CI -> ({lower_bound:.2f}, {upper_bound:.2f})")
```

Results:

```
Population Mean amount of Male Purchases -> 9427.24
Standard Error -> 7.65
total data size - > 414259
95% CI -> (9412.24, 9442.24)
```

For Female Customers:

```
pop_mean = data_female_customers["Purchase"].mean()
pop_std = data_female_customers["Purchase"].std(ddof=1)
n = len(data_female_customers)
se = pop_std / np.sqrt(n)
z_score = 1.96  # Z-score for 95% confidence
margin_of_error = z_score * se

# Calculate the confidence interval bounds
xa = pop_mean - margin_of_error
xb = pop_mean + margin_of_error
print(f"Population Mean amount of Female purchases-> {pop_mean:.2f}")
print(f"Standard Error -> {se:.2f}")
print(f"total data size - > {n}")
print(f"95% CI -> ({xa:.2f}, {xb:.2f})")
```

Results:

```
Population Mean amount of Female purchases-> 8736.54
Standard Error -> 12.47
total data size - > 135809
95% CI -> (8712.09, 8760.99)
```

Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?

Yes, the confidence interval computed using the entire dataset is wider for females.

Why is the Female CI Wider?

- Sample Size (n): The total number of female observations (n=135809) is significantly smaller than the number of male observations (n=414259). Since standard error (SE) is inversely proportional to the square root of the sample size, a smaller n leads to a larger SE. This means that for females, the standard error is larger, which directly impacts the width of the confidence interval.

$$SE = \sigma / \sqrt{n}$$

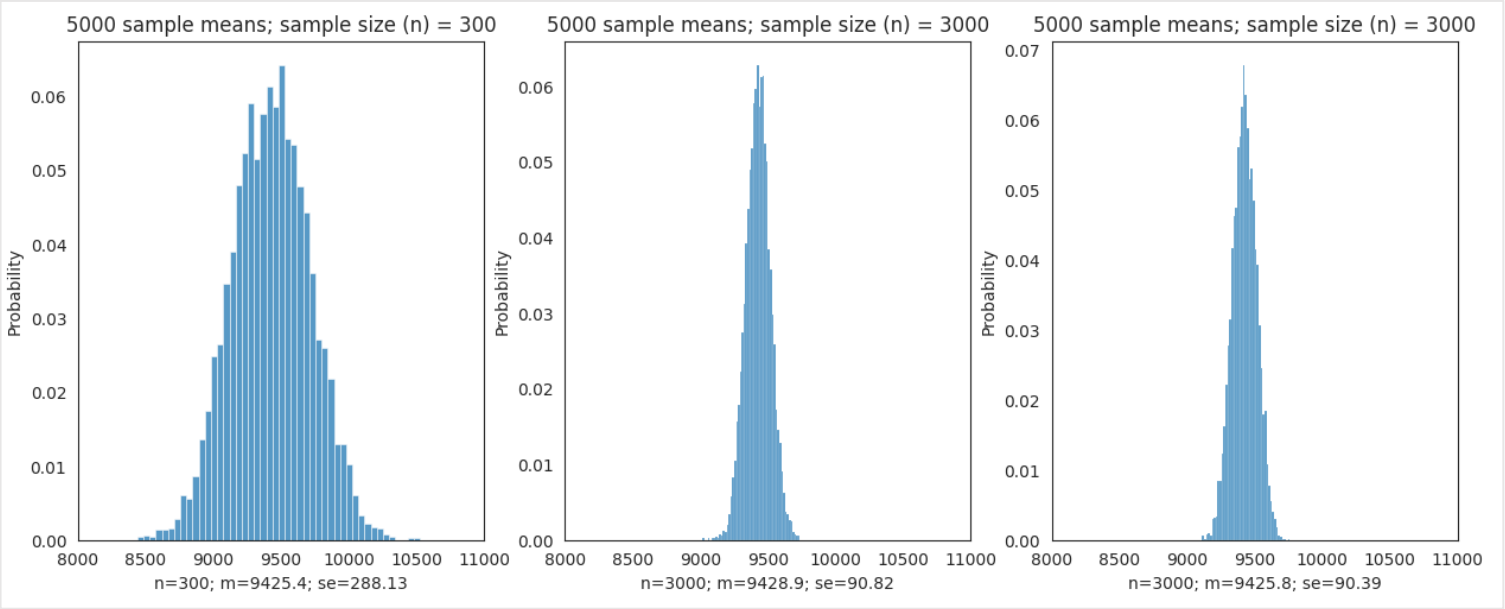
- Effect of Standard Error on CI:
  - Females: SE =12.47→ 95% CI = (8712.09, 8760.99) → Width ≈ 48.9
  - Males: SE =7.65 → 95% CI = (9412.24, 9442.24) → Width ≈ 30
  - A larger SE for females results in a wider CI. The larger width of the confidence interval for females reflects greater uncertainty about the true population mean due to the smaller sample size.

Now, compute the 95 % confidence interval for 3 smaller sample sizes - 300, 3000, and 30000.

Male Samples:

```
no_of_samples = 5000

plt.figure(figsize=(15,12))
subplot_no = 1
sample_sizes = [300, 3000, 3000]
for n in sample_sizes:
  new_sample = np.random.choice(a=data_male_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  male_sample_means = new_sample.mean(axis=1)

  plt.subplot(2,3,subplot_no)
  plt.title(f"{no_of_samples} sample means; sample size (n) = {n}")
  plt.xlim(8000,11000)
  #plt.xlabel(f"'Mean' of sample means={sample_means.mean():.1f}")
  plt.xlabel(f"n={n}; m={male_sample_means.mean():.1f};
se={male_sample_means.std(ddof=1):.2f}")
  sns.histplot(x=male_sample_means, stat="probability")

  subplot_no += 1
```

Results:




```
no_of_samples = 5000
```

```
sample_sizes = [300, 3000, 3000]
for n in sample_sizes:
  new_sample = np.random.choice(a=data_male_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  male_sample_means = new_sample.mean(axis=1)

  pop_mean = male_sample_means.mean()
  se = male_sample_means.std(ddof=1)    # SE is the std dev of the sample means
  z_score = 1.96  # For 95% confidence
  margin_of_error = z_score * se

  xa = pop_mean - margin_of_error
  xb = pop_mean + margin_of_error
  print(f"sample size - > {n}, Population Mean -> {pop_mean:.2f} SE -> {se:.2f},
95% CI -> ({xa:.2f}, {xb:.2f})")
```

Results:

```
sample size - > 300, Population Mean -> 9428.75, SE -> 284.56, 95% CI -> (8871.01, 9986.49)
sample size - > 3000, Population Mean -> 9426.13, SE -> 89.12, 95% CI -> (9251.44, 9600.81)
sample size - > 3000, Population Mean -> 9428.24, SE -> 88.85, 95% CI -> (9254.09, 9602.38)
```

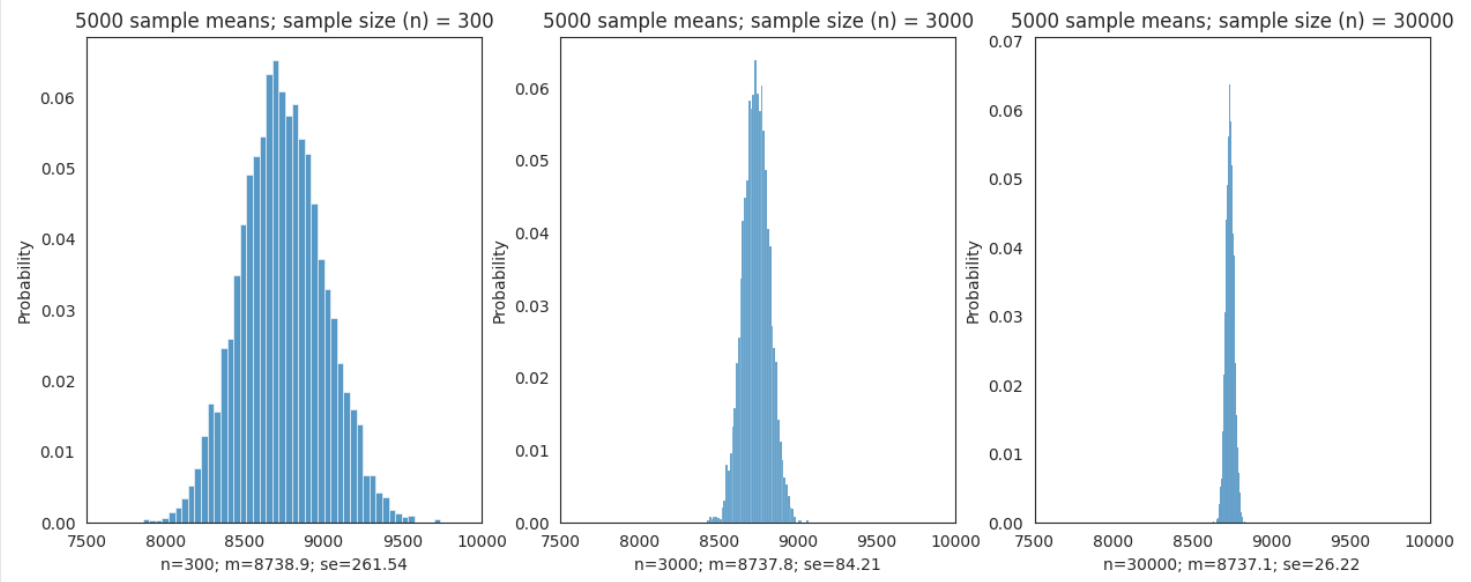
Insights:

Using the sample of male customers from n= 3000, we can say the interval within which the average spending of 50 million male customers lie is between 9254.09, and 9602.38.

Female Samples:

```
no_of_samples = 5000

plt.figure(figsize=(15,12))
subplot_no = 1
for n in [300, 3000, 30000]:
  new_sample = np.random.choice(a=data_female_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  female_sample_means = new_sample.mean(axis=1)

  plt.subplot(2,3,subplot_no)
  plt.title(f"{no_of_samples} sample means; sample size (n) = {n}")
  plt.xlim(7500,10000)
  #plt.xlabel(f"'Mean' of sample means={sample_means.mean():.1f}")
  plt.xlabel(f"n={n}; m={female_sample_means.mean():.1f};
se={female_sample_means.std(ddof=1):.2f}")
  sns.histplot(x=female_sample_means, stat="probability")

  subplot_no += 1
```

Results:




```
no_of_samples = 5000

sample_sizes = [300, 3000, 3000]
for n in sample_sizes:
  new_sample = np.random.choice(a=data_female_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  female_sample_means = new_sample.mean(axis=1)

  pop_mean = female_sample_means.mean()
  se = female_sample_means.std(ddof=1)    # SE is the std dev of the sample means
  z_score = 1.96  # For 95% confidence
  margin_of_error = z_score * se

  xa = pop_mean - margin_of_error
  xb = pop_mean + margin_of_error
  print(f"sample size - > {n}, Population Mean -> {pop_mean:.2f} SE -> {se:.2f},
95% CI -> ({xa:.2f}, {xb:.2f})")
```

Results:

```
sample size - > 300, Population Mean -> 8739.24 SE -> 264.63, 95% CI -> (8220.58, 9257.91)
sample size - > 3000, Population Mean -> 8739.22 SE -> 83.90, 95% CI -> (8574.78, 8903.65)
sample size - > 3000, Population Mean -> 8734.42 SE -> 83.17, 95% CI -> (8571.40, 8897.44)
```

Insights:

Using the sample of female customers from n= 3000, we can say the interval within which the average spending of 50 million female customers lie is between 8571.40, and 8897.44.

How is the width of the confidence interval affected by the sample size?

As the sample size increased from 300 to 3,000 to 30,000, the confidence interval becomes narrower. The **standard error (SE)** decreases as the sample size increases because Seis inversely proportional to sample size (n). A larger sample size provides a more reliable estimate of the population mean, reducing uncertainty.

Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes overlap. For females, the 95% CI for sample size of n = 300 (8220.58, 9257.91) overlaps with 95% CI for sample size of n = 3000 ((8574.78, 8903.65). A smaller sample size will have a wider confidence interval, which may overlap with the CI from a larger sample. Also, if the sample means are close enough, their confidence intervals might overlap even if the sample sizes are very different.

How does the sample size affect the shape of the distributions of the means?

Sample size is inversely proportional to the Standard Error which determines the thickness of the distribution. With larger sample size, the thickness of SE narrows and results in a thin curve of CLT.

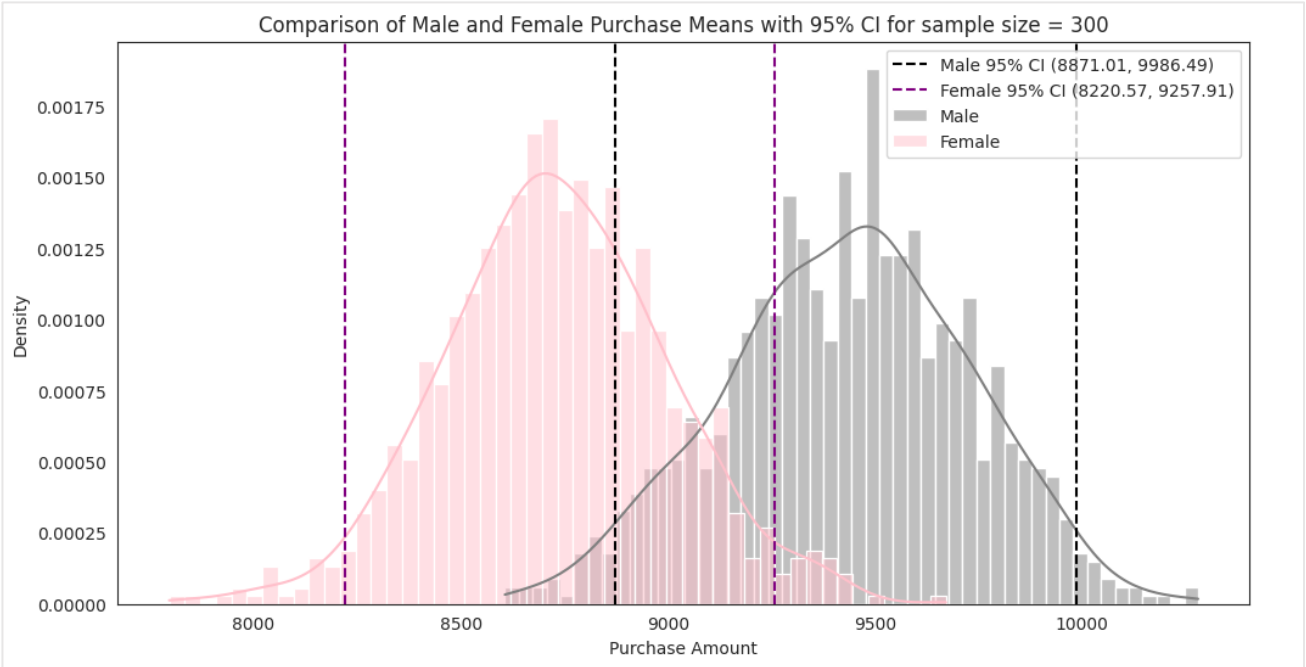
Do the confidence intervals for males and females overlap for any particular sample size?

Yes, the confidence intervals for males and females overlap for the sample size of n = 300.

```
# Define the sample means and standard errors for males and females
male_sample_mean = 9428.75
male_se = 284.56
female_sample_mean = 8739.24
female_se = 264.63

# Define the 95% CI (z-score for 95% confidence is 1.96)
z_score = 1.96
male_margin_of_error = z_score * male_se
female_margin_of_error = z_score * female_se

# Male CI
male_xa = male_sample_mean - male_margin_of_error
male_xb = male_sample_mean + male_margin_of_error


# Female CI
female_xa = female_sample_mean - female_margin_of_error
female_xb = female_sample_mean + female_margin_of_error
```

```
# Create a normal distribution for male and female samples
x = np.linspace(8000, 10500, 1000)
male_distribution = np.random.normal(male_sample_mean, male_se, 1000)
female_distribution = np.random.normal(female_sample_mean, female_se, 1000)

# Plotting the distributions
plt.figure(figsize=(12, 6))

# Plot male distribution
sns.histplot(male_distribution, color='grey', kde=True, stat="density",
label='Male', bins=50)

# Plot female distribution
sns.histplot(female_distribution, color='pink', kde=True, stat="density",
label='Female', bins=50)

# Plot vertical lines for CI
plt.axvline(male_xa, color='black', linestyle='--', label=f'Male 95% CI
({male_xa:.2f}, {male_xb:.2f})')
plt.axvline(male_xb, color='black', linestyle='--')

plt.axvline(female_xa, color='purple', linestyle='--', label=f'Female 95% CI
({female_xa:.2f}, {female_xb:.2f})')
plt.axvline(female_xb, color='purple', linestyle='--')

# Add labels and legend
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Comparison of Male and Female Purchase Means with 95% CI for sample size
= 300')
plt.legend()
plt.show()
```

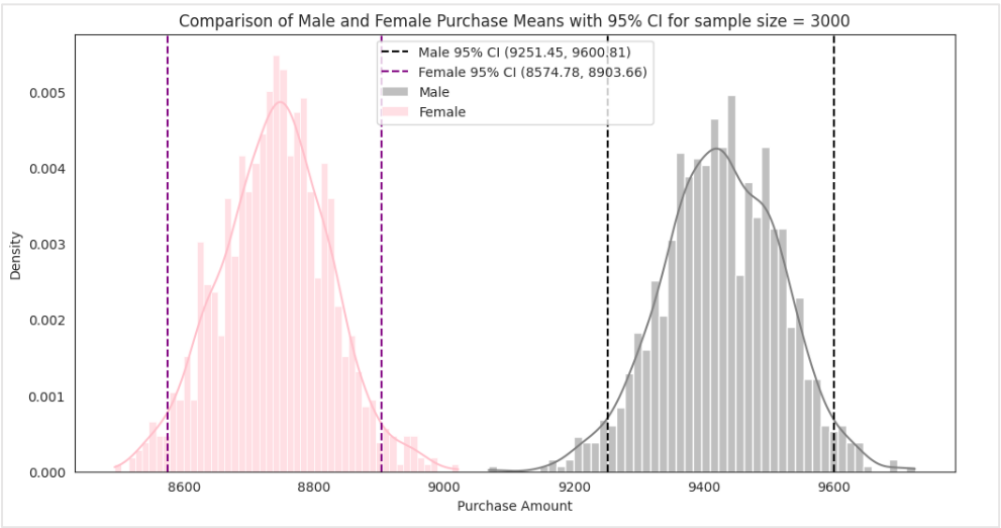
Results:



With smaller sample size, the standard error will be wider and so will be the upper and lower bounds of the confidence interval. This can be rectified with a larger sample size as illustrated below:

```
# Define the sample means and standard errors for males and females
```

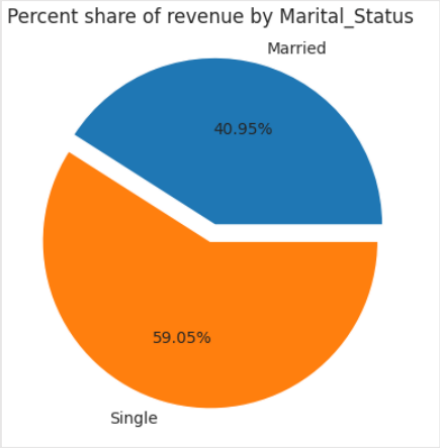
```
male_sample_mean = 9426.13
male_se = 89.12
female_sample_mean = 8739.22
female_se = 83.90

# Define the 95% CI (z-score for 95% confidence is 1.96)
z_score = 1.96
male_margin_of_error = z_score * male_se
female_margin_of_error = z_score * female_se

# Male CI
male_xa = male_sample_mean - male_margin_of_error
male_xb = male_sample_mean + male_margin_of_error

# Female CI
female_xa = female_sample_mean - female_margin_of_error
female_xb = female_sample_mean + female_margin_of_error

# Create a normal distribution for male and female samples
x = np.linspace(8000, 10500, 1000)
male_distribution = np.random.normal(male_sample_mean, male_se, 1000)
female_distribution = np.random.normal(female_sample_mean, female_se, 1000)

# Plotting the distributions
plt.figure(figsize=(12, 6))

# Plot male distribution
sns.histplot(male_distribution, color='grey', kde=True, stat="density",
label='Male', bins=50)

# Plot female distribution
sns.histplot(female_distribution, color='pink', kde=True, stat="density",
label='Female', bins=50)

# Plot vertical lines for CI
plt.axvline(male_xa, color='black', linestyle='--', label=f'Male 95% CI
({male_xa:.2f}, {male_xb:.2f})')
plt.axvline(male_xb, color='black', linestyle='--')

plt.axvline(female_xa, color='purple', linestyle='--', label=f'Female 95% CI
({female_xa:.2f}, {female_xb:.2f})')
plt.axvline(female_xb, color='purple', linestyle='--')

# Add labels and legend
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Comparison of Male and Female Purchase Means with 95% CI for sample size
= 3000')
plt.legend()
plt.show()
```



Recommendations: While the confidence intervals overlap at a sample size of 300, this changed with larger sample sizes. Hence, the brand should always **analyze gender spending trends** using larger datasets to see if more precise measurements reveal differences.

How does Marital_Status affect the amount spent?

```
plt.pie(x=clipped_data.groupby('Marital_Status')['Purchase'].sum(),labels=clipped_d
ata.groupby('Marital_Status')['Purchase'].sum().index,explode=(0.1,0),autopct='%.2f
%%')
plt.title("Percent share of revenue by Marital_Status")
plt.show()
```



Insights: 59.05 % of the total revenue is generated from the customers who are Single. Remaining is from Married customers

Now, lets look at the mean purchase amount by Married and single customers by calculating CI of 95% using Central Limit Theorem (CLT)

Let's first filter purchases by marital status

```
data_married_customers = clipped_data.loc[clipped_data["Marital_Status"] ==
"Married"][["Marital_Status", "User_ID","Purchase"]]
data_single_customers = clipped_data.loc[clipped_data["Marital_Status"] ==
"Single"][["Marital_Status", "User_ID","Purchase"]]
```

```
Population Mean amount of Single Customers' Purchases -> 9258.82
Standard Error -> 8.54
total data size - > 324731
95% CI -> (9242.09, 9275.55)
```

```
Population Mean amount of Married Customers' Purchases -> 9253.67
Standard Error -> 10.20
total data size - > 225337
95% CI -> (9233.67, 9273.67)
```

Single Customers' Samples

```
no_of_samples = 5000
sample_sizes = [300, 3000, 3000]
```

```
for n in sample_sizes:
  new_sample = np.random.choice(a=data_single_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  single_sample_means = new_sample.mean(axis=1)

  pop_mean = single_sample_means.mean()
  se = single_sample_means.std(ddof=1)    # SE is the std dev of the sample means
  z_score = 1.96  # For 95% confidence
  margin_of_error = z_score * se

  xa = pop_mean - margin_of_error
  xb = pop_mean + margin_of_error
  print(f"sample size - > {n}, Population Mean -> {pop_mean:.2f}, SE -> {se:.2f},
95% CI -> ({xa:.2f}, {xb:.2f})")
```

Results:

```
sample size - > 300, Population Mean -> 9260.92, SE -> 282.74, 95% CI -> (8706.76, 9815.09)
sample size - > 3000, Population Mean -> 9260.19, SE -> 89.16, 95% CI -> (9085.43, 9434.95)
sample size - > 3000, Population Mean -> 9259.22, SE -> 88.91, 95% CI -> (9084.97, 9433.47)
```

Insights: Using the sample of married customers from n= 3000, we can say the interval within which the average spending lie is between 9084 and 9433.

Married Customers' Samples

```
no_of_samples = 5000

sample_sizes = [300, 3000, 3000]
for n in sample_sizes:
  new_sample = np.random.choice(a=data_married_customers["Purchase"],
size=(no_of_samples,n)) #(rows,cols)
  married_sample_means = new_sample.mean(axis=1)

  pop_mean = married_sample_means.mean()
  se = married_sample_means.std(ddof=1)    # SE is the std dev of the sample means
  z_score = 1.96  # For 95% confidence
  margin_of_error = z_score * se

  xa = pop_mean - margin_of_error
  xb = pop_mean + margin_of_error
  print(f"sample size - > {n}, Population Mean -> {pop_mean:.2f}, SE -> {se:.2f},
95% CI -> ({xa:.2f}, {xb:.2f})")
```

Results:

```
sample size - > 300, Population Mean -> 9251.25, SE -> 277.09, 95% CI -> (8708.15, 9794.36)
sample size - > 3000, Population Mean -> 9255.48, SE -> 89.57, 95% CI -> (9079.93, 9431.03)
sample size - > 3000, Population Mean -> 9254.76, SE -> 89.75, 95% CI -> (9078.85, 9430.66)
```

Insights: Using the sample of married customers from n= 3000, we can say the interval within which the average spending lie is between 9078 and 9430.

Business Recommendations:

- **Targeted marketing**: Since 75% of transactions and a higher average purchase amount is made by males, it would be effective to tailor marketing strategies to cater to their preferences and needs. Specific promotions, product offerings, or advertising campaigns can be designed to attract customers from this gender.
- **Optimize revenue from specific age groups**: Since a significant number of transactions are made by customers between the ages of 26 and 45, it is important to drive more sales through this demographic. Use AI-driven product recommendations based on their past purchases. Customers in the 26-45 age range are often looking for convenience and relevance, so personalized suggestions can increase conversion rates.
- **Enhance Focus on Single Customers**: With 59.05% of total revenue coming from single customers, it's crucial to prioritize meeting their needs and preferences. By gaining insights into their motivations and tailoring personalized offers, you can significantly improve their shopping experience and boost loyalty. Create attractive product bundles that cater to individual needs, such as health, fitness, or entertainment, encouraging higher spending.
- **Engage with new residents**: As a major portion of transactions (53.75%) are done by customers who have recently moved to the current city (<=2 years), it offers an opportunity to engage with these new residents. Welcome offers, first order offers, and incentives for newcomers can help capture their attention.