walmartcasestudy

June 28, 2025

Business Case Study : Walmart - Confidence Interval and CLT :

About Walmart: Walmart , founded in 1962 by Sam Walton , is a retail giant and one of the world's largest and most influential companies. Headquartered in Bentonville, Arkansas , this American multinational corporation has established itself as a global powerhouse in the retail industry. Walmart operates a vast network of hypermarkets, discount department stores, and grocery stores under various brand names across the United States and in numerous countries around the world. Known for its "Everyday Low Prices" strategy, Walmart has redefined the retail landscape with its commitment to offering a wide range of products at affordable prices. With its extensive supply chain and efficient distribution systems, the company has played a pivotal role in shaping consumer expectations and shopping habits. Beyond retail, Walmart has also ventured into e-commerce, technology innovation, and sustainability initiatives, further solidifying its position as a key player in the modern retail ecosystem.

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

About Data: The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday.It has information of about 0.5 Million transactions during Black Friday throughout various years.

Importing libraries

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import scipy.stats as spy
```

[2]:
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/
↪original/walmart_data.csv?1641285094
```

```
Downloading…
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/ori
ginal/walmart_data.csv?1641285094
To: /content/walmart_data.csv?1641285094
100% 23.0M/23.0M [00:00<00:00, 109MB/s]
```

[3]:
```
df = pd.read_csv('walmart_data.csv?1641285094')
df
```

[3]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
0       1000001  P00069042      F  0-17          10             A
1       1000001  P00248942      F  0-17          10             A
2       1000001  P00087842      F  0-17          10             A
3       1000001  P00085442      F  0-17          10             A
4       1000002  P00285442      M   55+          16             C
…           …         …    …    …         …             …
550063  1006033  P00372445      M 51-55          13             B
550064  1006035  P00375436      F 26-35           1             C
550065  1006036  P00375436      F 26-35          15             B
550066  1006038  P00375436      F   55+           1             C
550067  1006039  P00371644      F 46-50           0             B

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                               2               0                 3      8370
1                               2               0                 1     15200
2                               2               0                12      1422
3                               2               0                12      1057
4                              4+               0                 8      7969
…                             …             …                …       …
550063                          1               1                20       368
550064                          3               0                20       371
550065                         4+               1                20       137
550066                          2               0                20       365
550067                         4+               1                20       490

[550068 rows x 10 columns]
```

Exploratory Data Analysis

[4]:
```
df.shape
```

[4]:
```
(550068, 10)
```

[5]:
```
df.columns
```

[5]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

[6]:
```
#number of unique values in our data
for i in df.columns:
  print(i,':',df[i].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
```

[7]:
```python
#checking null values in every column of our data
df.isnull().sum()
```

[7]:
```
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
```

[8]:
```python
df.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
```

Changing the Datatype of Columns

[9]:
```python
columns_to_convert = ['Age','City_Category', 'Stay_In_Current_City_Years',
↪'Marital_Status'] # Add other columns as needed

for col in columns_to_convert:
    if col in df.columns:
        df[col] = df[col].astype('category')
```

[10]:
```python
df.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  category
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: category(4), int64(4), object(2)
memory usage: 27.3+ MB
```

[11]:
```python
df.isna()
```

[11]:
```
        User_ID  Product_ID  Gender    Age  Occupation  City_Category  \
0         False       False   False  False       False          False
1         False       False   False  False       False          False
2         False       False   False  False       False          False
3         False       False   False  False       False          False
4         False       False   False  False       False          False
…           …          …       …      …         …              …
550063    False       False   False  False       False          False
550064    False       False   False  False       False          False
550065    False       False   False  False       False          False
550066    False       False   False  False       False          False
550067    False       False   False  False       False          False

        Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                            False           False             False     False
1                            False           False             False     False
2                            False           False             False     False
3                            False           False             False     False
```

```
4                             False        False          False     False
…                             …        …              …        …
550063                        False        False          False     False
550064                        False        False          False     False
550065                        False        False          False     False
550066                        False        False          False     False
550067                        False        False          False     False

[550068 rows x 10 columns]
```

[12]:
```
df.duplicated()
```

[12]:
```
0         False
1         False
2         False
3         False
4         False
          …
550063    False
550064    False
550065    False
550066    False
550067    False
Length: 550068, dtype: bool
```

[13]:
```
#Statistical Summary of Numerical Features:

df.describe()
```

[13]:

	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

Insights

The dataset provides information on the following variables:

User_ID: It contains unique identification numbers assigned to each user. The dataset includes a total of 550,068 user records.

Occupation: This variable represents the occupation of the users. The dataset includes values ranging from 0 to 20, indicating different occupations.

Product_Category: It indicates the category of the products purchased by the users. The dataset includes values ranging from 1 to 20, representing different product categories.

Purchase: This variable represents the purchase amount made by each user. The dataset includes purchase values ranging from 12 to 23,961.

[14]:
```
# description of columns with 'object' datatype
df.describe(include = 'object')
```

[14]:
```
       Product_ID  Gender
count      550068  550068
unique       3631       2
top     P00265242       M
freq         1880  414259
```

Insights

The provided data represents summary statistics for two variables: Product_ID and Gender. Here is a breakdown of the information:

Product_ID: There are 3,631 unique values observed in this variable, indicating that there are 3,631 different products. The top value, which appears most frequently, is 'P00265242'. This value occurs 1,880 times in the dataset.

Gender: There are 2 unique values in this variable, which suggests that it represents a binary category. The top value is 'M', indicating that 'M' is the most common gender category. It appears 414,259 times in the dataset.

These summary statistics provide insights into the distribution and frequency of the Product_ID and Gender variables. They give an understanding of the number of unique products, the most common product, and the dominant gender category in the dataset.

Value_counts and unique attributes

[15]:
```
df.nunique()
```

[15]:
```
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
```

[16]:
```
# How many unique customers' data is given in the dataset?
df['User_ID'].nunique()
```

[16]: 5891

Insights

- We have the data of 5891 customers who made at least one purchase on Black Friday in Walmart

[17]:
```
# Total number of transactions made by each gender
np.round(df['Gender'].value_counts(normalize = True) * 100, 2)
```

[17]:
```
Gender
M    75.31
F    24.69
Name: proportion, dtype: float64
```

Insights

- It is clear from the above that out of every four transactions, three are made by males

[18]:
```
# What is the average total purchase made by each user in each gender ?



# Group the dataframe by 'Gender' and 'User_ID', then sum the 'Purchase' column
df1 = df.groupby(by=['Gender','User_ID'])['Purchase'].sum().reset_index()

# Now you can group df1 by 'Gender' to calculate the average purchase amount␣
↪per user for each gender
average_purchase_per_user_by_gender = df1.groupby(by='Gender')['Purchase'].
↪mean()

print(average_purchase_per_user_by_gender)
```

```
Gender
F    712024.394958
M    925344.402367
Name: Purchase, dtype: float64
```

Insights

On an average each male makes a total purchase of 712024.394958.

On an average each female makes a total purchase of 925344.402367.

[19]:
```
df['Age'].value_counts()
```

[19]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
```

```
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

[20]:
```python
df['City_Category'].value_counts()
```

[20]:
```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

[21]:
```python
duplicate=df.duplicated().value_counts()
print(duplicate)
```

```
False    550068
Name: count, dtype: int64
```

df.isnull() Returns a DataFrame of the same shape as df with True where values are missing (NaN) and False elsewhere. .sum() Adds up the True values column-wise (treating True as 1), giving the count of missing values per column

[22]:
```python
#Checking missing values
df.isnull().sum()
```

[22]:
```
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
```

Insights:

From the above analysis, it is clear that, data has total of 10 features with mixed alpha numeric data. Also we can see that there is no missing data in the columns.

Visual Analysis(Univariate & Bivariate):

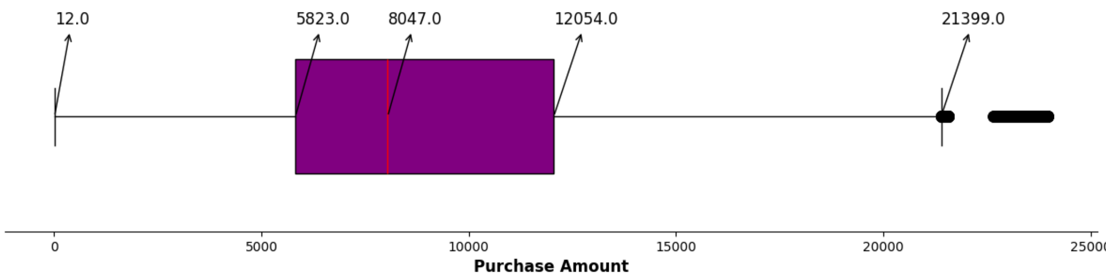
Visualizing the distribution and detecting outliers in the continuous variables of our dataset using box plots

Detecting Outliers for Purchase Column

[23]:

```
#setting the plot style
fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,1,height_ratios=[0.65, 0.35])

#plotting in lower subplot
ax1 = fig.add_subplot(gs[1,0])
#creating vertical boxplot,with patch_artist to colour the box
boxplot = ax1.boxplot(x = df['Purchase'],vert = False,patch_artist =␣
 ↪True,widths = 0.5)

# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='purple')

# Customize median line
boxplot['medians'][0].set(color='red')

# Customize outlier markers
for flier in boxplot['fliers']:
        flier.set(marker='o', markersize=8, markerfacecolor= "black")

#removing the axis lines
for s in ['top','left','right']:
        ax1.spines[s].set_visible(False)

#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']]

#getting the upperlimit,Q1,Q3 and lowerlimit
median = df['Purchase'].quantile(0.5)

#getting Q2
for i,j in info:
#using i,j here because of the output type of info list comprehension
        ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize =␣
 ↪12,
                          arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))
        ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize =␣
 ↪12,
                          arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#adding the median separately because it was included in info list
ax1.annotate(text = f"{median:.1f}",xy = (median,1),xytext = (median + 1,1.
 ↪4),fontsize = 12,
arrowprops= dict(arrowstyle="<-", lw=1, connectionstyle="arc,rad=0"))
```

```
# #removing y-axis ticks
ax1.set_yticks([])

#adding axis label
ax1.set_xlabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
plt.show()
```



Calculating the Number of Outliers

As seen above, Purchase amount over 21399 is considered as outlier. We will count the number of outliers as below

[24]:
```
len(df.loc[df['Purchase']> 21399,'Purchase'])
```

[24]:
```
2677
```

Insights

Outliers

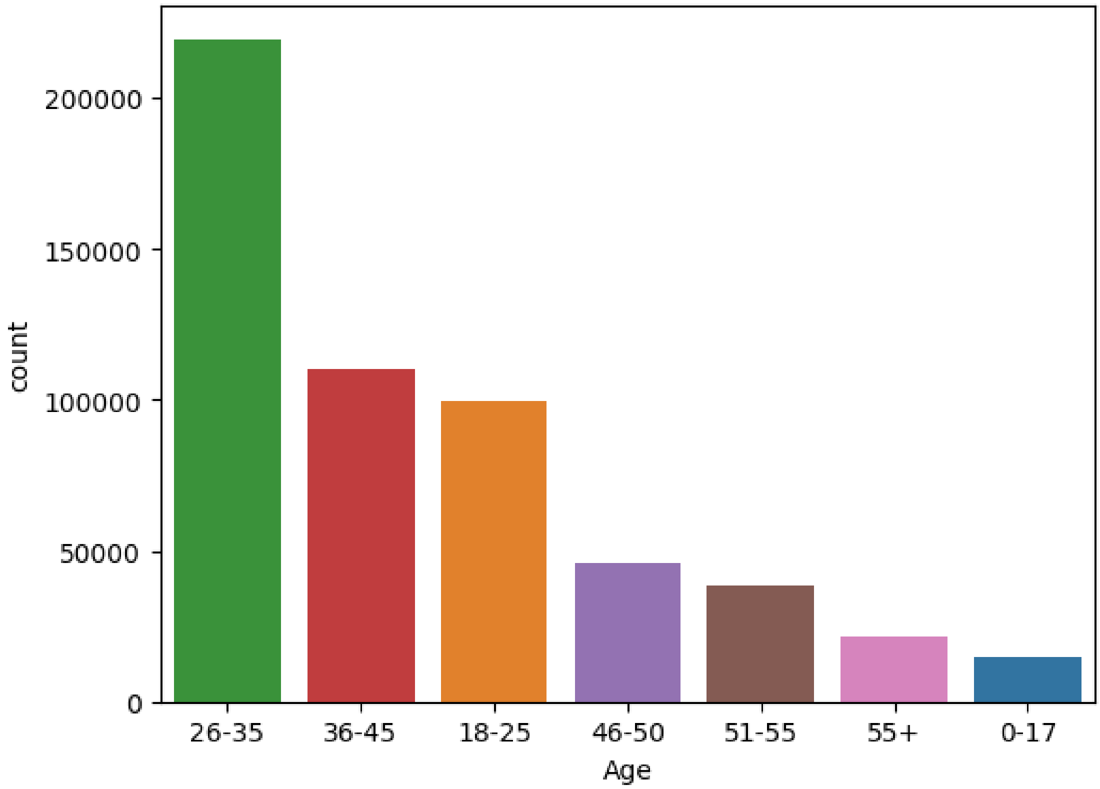
There are total of 2677 outliers which is roughly 0.48% of the total data present in purchase amount. We will not remove them as it indicates a broad range of spending behaviors during the sale, highlighting the importance of tailoring marketing strategies to both regular and high-value customers to maximize revenue.

Distribution

Data suggests that the majority of customers spent between 5,823 USD and 12,054 USD , with the median purchase amount being 8,047 USD .

The lower limit of 12 USD while the upper limit of 21,399 USD reveal significant variability in customer spending

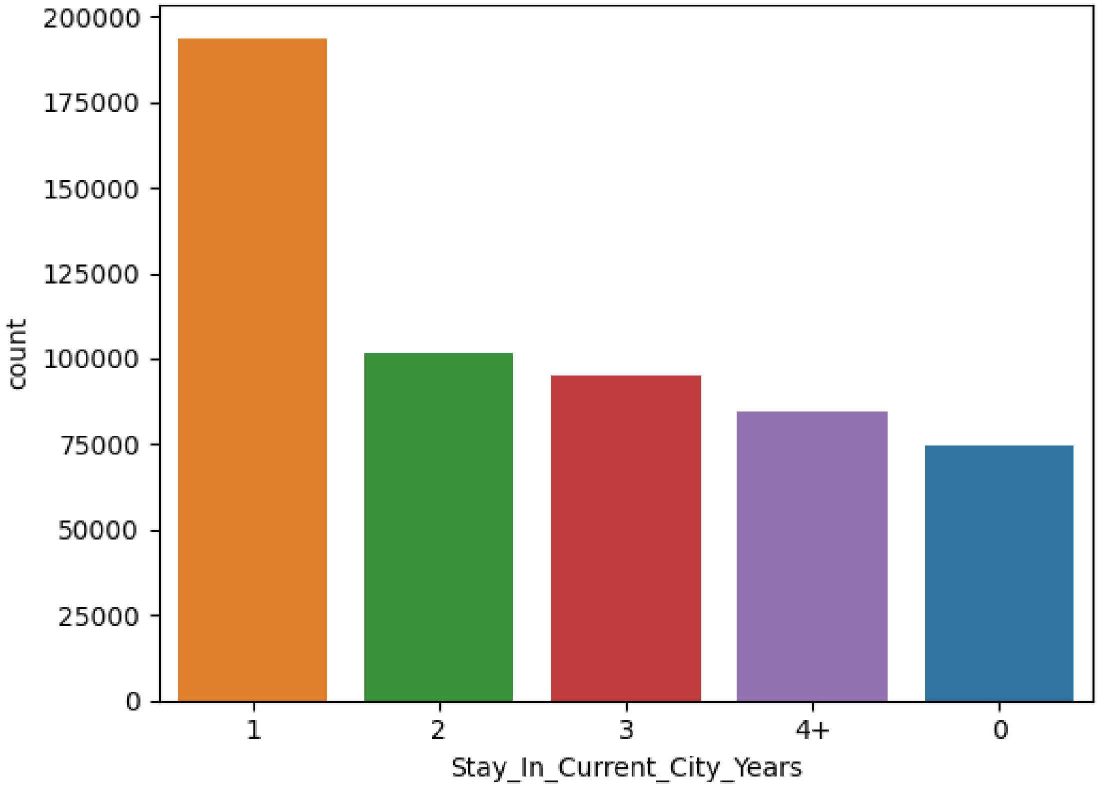
[25]:
```
#Customer Age Distribution
sns.countplot(data=df,x='Age',order=df['Age'].value_counts().index,hue='Age') #␣
↪Corrected typo and added .index
plt.show()
```



Insights

1. The age group of 26-35 represents the largest share of Walmart's Black Friday sales, accounting for 40% of the sales. This suggests that the young and middle-aged adults are the most active and interested in shopping for deals and discounts .
2. The 36-45 and 18-25 age groups are the second and third largest segments, respectively, with 20% and 18% of the sales. This indicates that Walmart has a diverse customer base that covers different life stages and preferences.
3. The 46-50, 51-55, 55+, and 0-17 age groups are the smallest customer segments , with less than 10% of the total sales each. This implies that Walmart may need to improve its marketing strategies and product offerings to attract more customers from these age groups, especially the seniors and the children.

[26]:
```
#Customer Stay In current City Distribution

sns.
 ↪countplot(data=df,x='Stay_In_Current_City_Years',order=df['Stay_In_Current_City_Years'].
 ↪value_counts().index,hue='Stay_In_Current_City_Years') # Corrected typo and␣
 ↪added .index
plt.show()
```



Insights

The data suggests that the customers are either new to the city or move frequently, and may have different preferences and needs than long-term residents.

The majority of the customers (49%) have stayed in the current city for one year or less . This suggests that Walmart has a strong appeal to newcomers who may be looking for affordable and convenient shopping options.

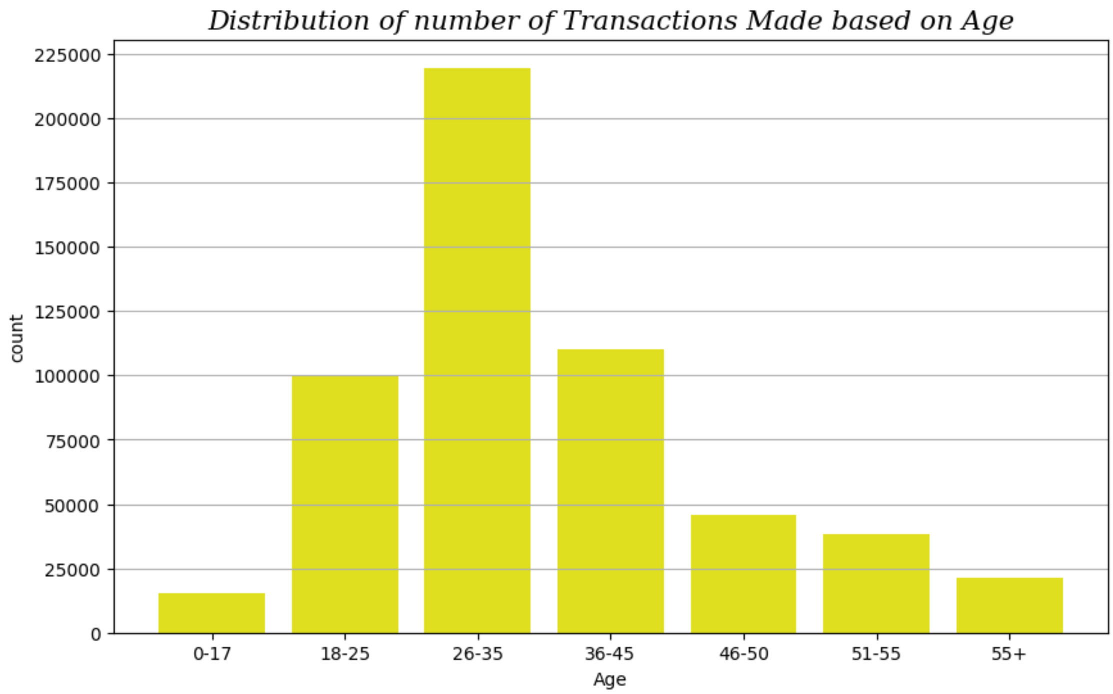
4+ years category (14%) customers indicates that Walmart has a loyal customer base who have been living in the same city for a long time.

The percentage of customers decreases as the stay in the current city increases which suggests that Walmart may benefit from targeting long-term residents for loyalty programs and promotions .

[27]:
```
#Distribution of number of Transactions Made based on Age
plt.figure(figsize = (10, 6))
plt.title('Distribution of number of Transactions Made based on Age',
fontsize = 15,
fontweight = 400,
fontstyle = 'oblique',
fontfamily = 'serif')
plt.yticks(np.arange(0, 250001, 25000))
plt.grid('y')
```

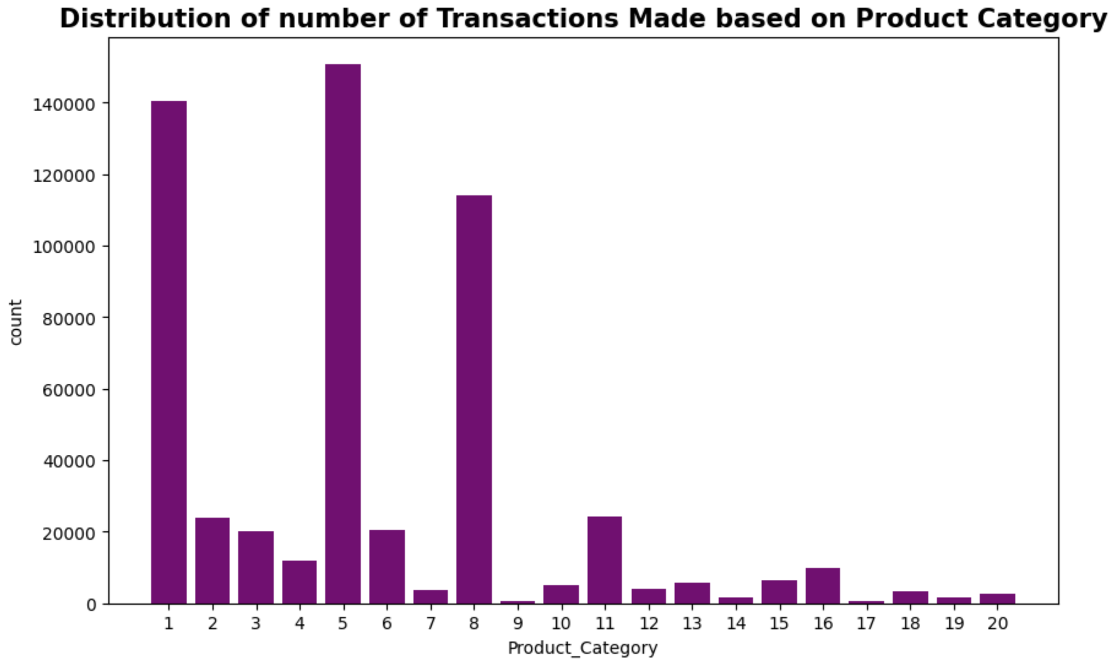
```
sns.countplot(data = df, x = 'Age',\
order = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55',␣
↪'55+'],color='yellow')
plt.plot()
```

[27]: []



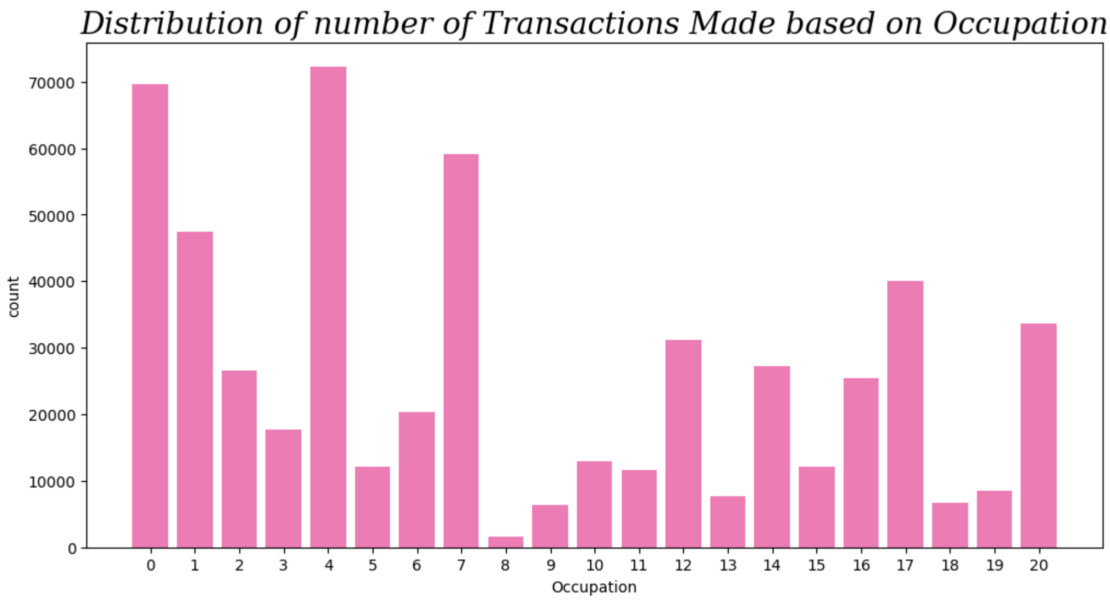
[28]:
```
#Distribution of number of Transactions Made based on Product Category
plt.figure(figsize = (10, 6))
plt.title('Distribution of number of Transactions Made based on Product␣
↪Category', fontsize = 15, fontweight = 600)
sns.countplot(data = df, x = 'Product_Category',color='purple')
plt.plot()
```

[28]: []

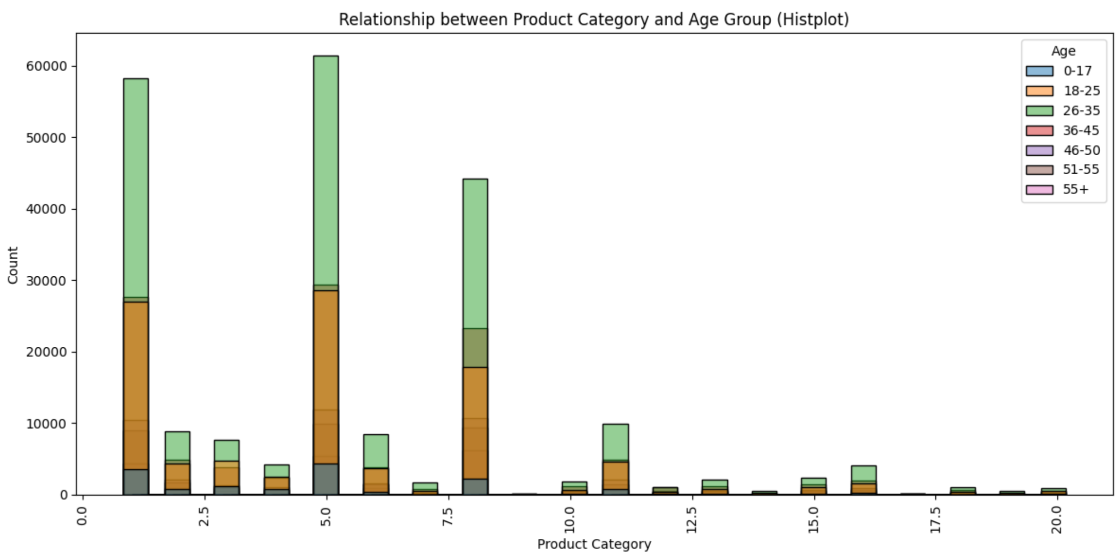


[29]:
```
#Distribution of number of Transactions Made based on Occupation
plt.figure(figsize = (12, 6))
plt.title('Distribution of number of Transactions Made based on Occupation',
fontsize = 20,
fontweight = 400,
fontstyle = 'oblique',
fontfamily = 'serif')
sns.countplot(data = df, x = 'Occupation',color='hotpink')
plt.plot()
```

[29]: []



[30]:
```
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Product_Category', hue='Age',shrink=3)
plt.title('Relationship between Product Category and Age Group (Histplot)')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
[31]: # Calculate the count of purchases for each combination of Age and Product Category
age_product_counts = df.groupby(['Age', 'Product_Category'], observed=True).size().unstack(fill_value=0)

# Create the heatmap
plt.figure(figsize=(16, 10)) # Increased figure size for better annotation readability
sns.heatmap(age_product_counts, annot=True, fmt='d', cmap='YlGnBu') # annot=True to show counts, fmt='d' for integer format
plt.title('Heatmap of Purchase Counts by Age Group and Product Category')
plt.xlabel('Product Category')
plt.ylabel('Age Group')
plt.tight_layout()
plt.show()
```

Insights of different age group buying products

Overall Activity:

The darker shades indicate higher purchase counts. It's evident that certain Age Groups and Product Categories have significantly more activity than others.

Dominant Age Groups:

The rows corresponding to the '26-35' and '36-45' age groups appear to have generally darker shades across many product categories, confirming the earlier observation from the histogram that these are highly active purchasing groups.

Most Popular Product Categories:

The columns corresponding to Product Categories 1, 5, and 8 consistently show darker shades across most age groups, indicating that these are the most popular product categories in terms of purchase volume.

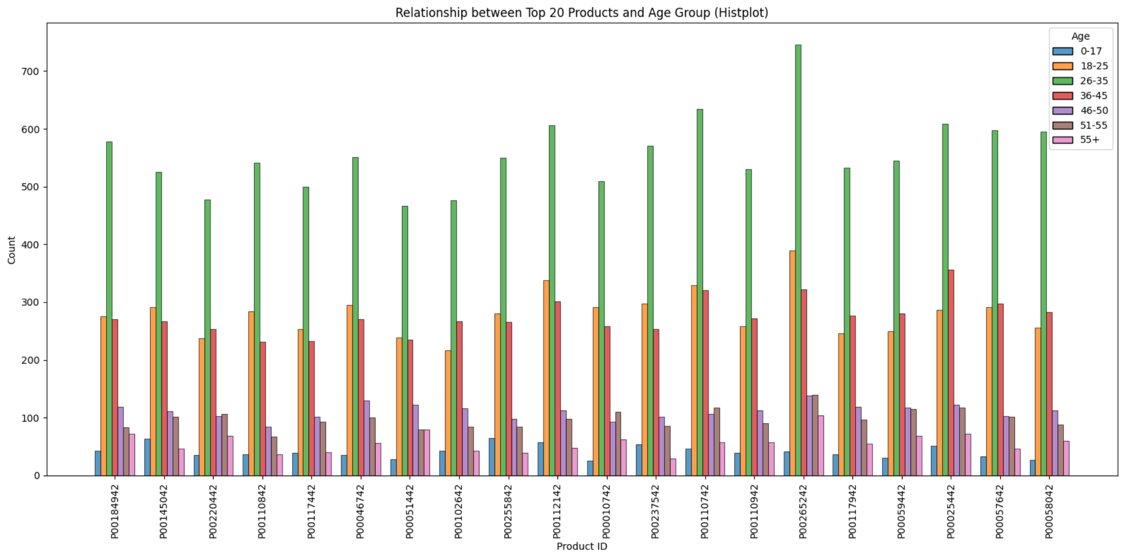
Niche Categories:

Some product categories have very light shades across all age groups, suggesting they have much lower purchase counts.

Age Group Preferences:

While there are general trends, you can also observe some variations in color intensity within specific product categories across different age groups. This hints at potential age-specific preferences for certain product categories. For example, you might see a darker shade for a particular age group in a category where other age groups have lighter shades.

[32]:
```python
# Get the top 20 most frequently purchased Product_IDs
top_20_product_ids = df['Product_ID'].value_counts().nlargest(20).index

# Filter the DataFrame to include only transactions for these top 20 products
df_top_20_products = df[df['Product_ID'].isin(top_20_product_ids)]

# Create the histogram for the top 20 products and Age group
plt.figure(figsize=(16, 8)) # Increased figure size for better readability
sns.histplot(data=df_top_20_products, x='Product_ID', hue='Age',␣
 ↪multiple='dodge', shrink=0.8)
plt.title('Relationship between Top 20 Products and Age Group (Histplot)')
plt.xlabel('Product ID')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Insights

Consistent Dominance:

Similar to the overall product category analysis, the '26-35' age group appears to be the largest contributor to the purchase counts for almost all of the top 20 products. The bars representing this age group are consistently the tallest across the different Product IDs.

Second Most Active Group:

The '36-45' age group generally represents the second largest portion of purchases for these top products.

Variations in Other Age Groups:

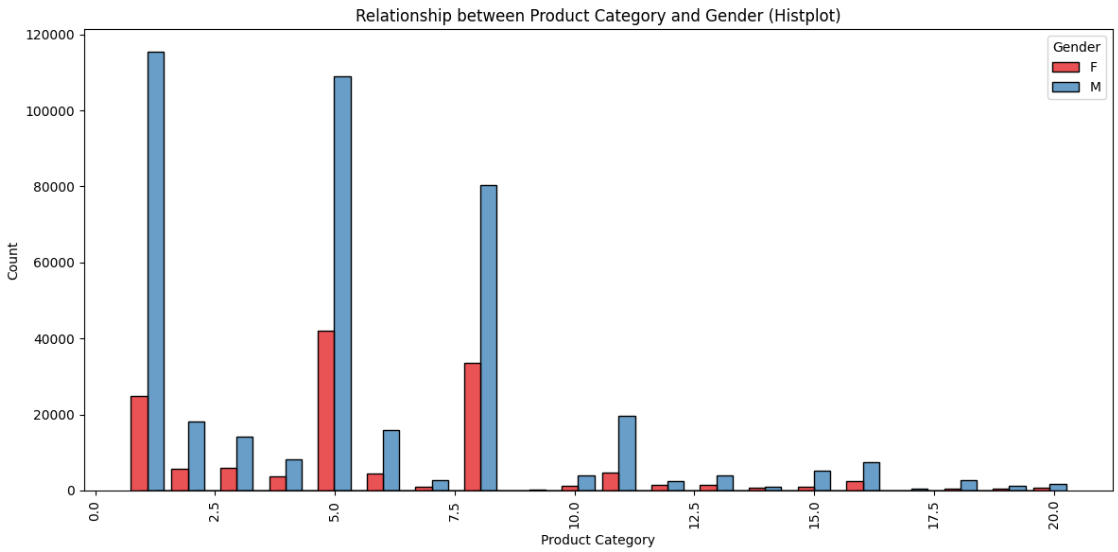
The presence and purchase volume of other age groups ('18-25', '46-50', '51-55', '55+', '0-17') vary more significantly across the top 20 products. Some products might see a relatively higher contribution from one of these age groups compared to others.

Product-Specific Age Distribution:

While the general trend shows '26-35' and '36-45' as the primary buyers, there might be subtle variations in the distribution of other age groups for specific product IDs. This could indicate slight age-based preferences even within the top-selling items.

[33]:
```
# Relationship with product category and genders

plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Product_Category', hue='Gender',palette='Set1',␣
  ↪multiple='dodge',shrink=4.0) # Changed shrink to 1.0
plt.title('Relationship between Product Category and Gender (Histplot)')
plt.xlabel('Product Category')
plt.ylabel('Count')
```

```
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Insights

Higher Male Purchase Counts:

Across almost all product categories, the purchase counts for males (blue bars) are significantly higher than those for females (orange bars). This aligns with the earlier observation that there are more transactions made by males in the dataset.

Popular Categories for Both Genders:

Product Categories 1, 5, 8, and 10 appear to be popular among both genders, showing relatively high purchase counts for both male and female customers.

Gender Preference Variations:

While males generally purchase more in every category, the proportion of purchases between genders might vary across categories. For example, in some categories, the difference between male and female purchase counts might be smaller relative to the total purchases in that category compared to others.

Less Frequent Categories:

Some product categories, particularly those with higher numbers (e.g., 19, 20), have lower purchase counts for both genders.

[34]:
```
#Average total purchase made by each user in each marital status

df['Marital_Status'] = df['Marital_Status'].apply(lambda x: 'Married' if x == 1␣
↪else 'Single')
```

```
df_marital_status=pd.DataFrame(df.groupby(by=['Marital_Status','User_ID'],␣
↪observed=True)['Purchase'].sum()).reset_index().rename(columns={'Purchase':
↪'Avg_Purchase'})
df_marital_status.groupby(by='Marital_Status', observed=True)['Avg_Purchase'].
↪mean()
```

[34]:

```
Marital_Status
Single     880575.781972
Married    843526.796686
Name: Avg_Purchase, dtype: float64
```

On an average each Married customer makes a total purchase of 354249.753013. On an average each Single customer makes a total purchase of 510766.838737.

[35]:

```
#Relationship of Age - Ordered by unique customers (descending)

df_age_dist=pd.DataFrame(df.groupby(by=['Age'], observed=True)['User_ID'].
↪nunique()).reset_index().rename(columns={'User_ID':'unique_customers'})
df_age_dist['percent_share'] = (df_age_dist['unique_customers'] /␣
↪df_age_dist['unique_customers'].sum()) * 100 # Calculate as percentage
df_age_dist['cumulative_percent'] = df_age_dist['percent_share'].cumsum()
df_age_dist['percent_share'] = np.round(df_age_dist['percent_share'], 2) #␣
↪Round to 2 decimal places
df_age_dist['cumulative_percent'] = np.round(df_age_dist['cumulative_percent'],␣
↪2) # Round cumulative as well

# Order by unique_customers in descending order
df_age_dist = df_age_dist.sort_values(by='unique_customers', ascending=False)

df_age_dist
```

[35]:

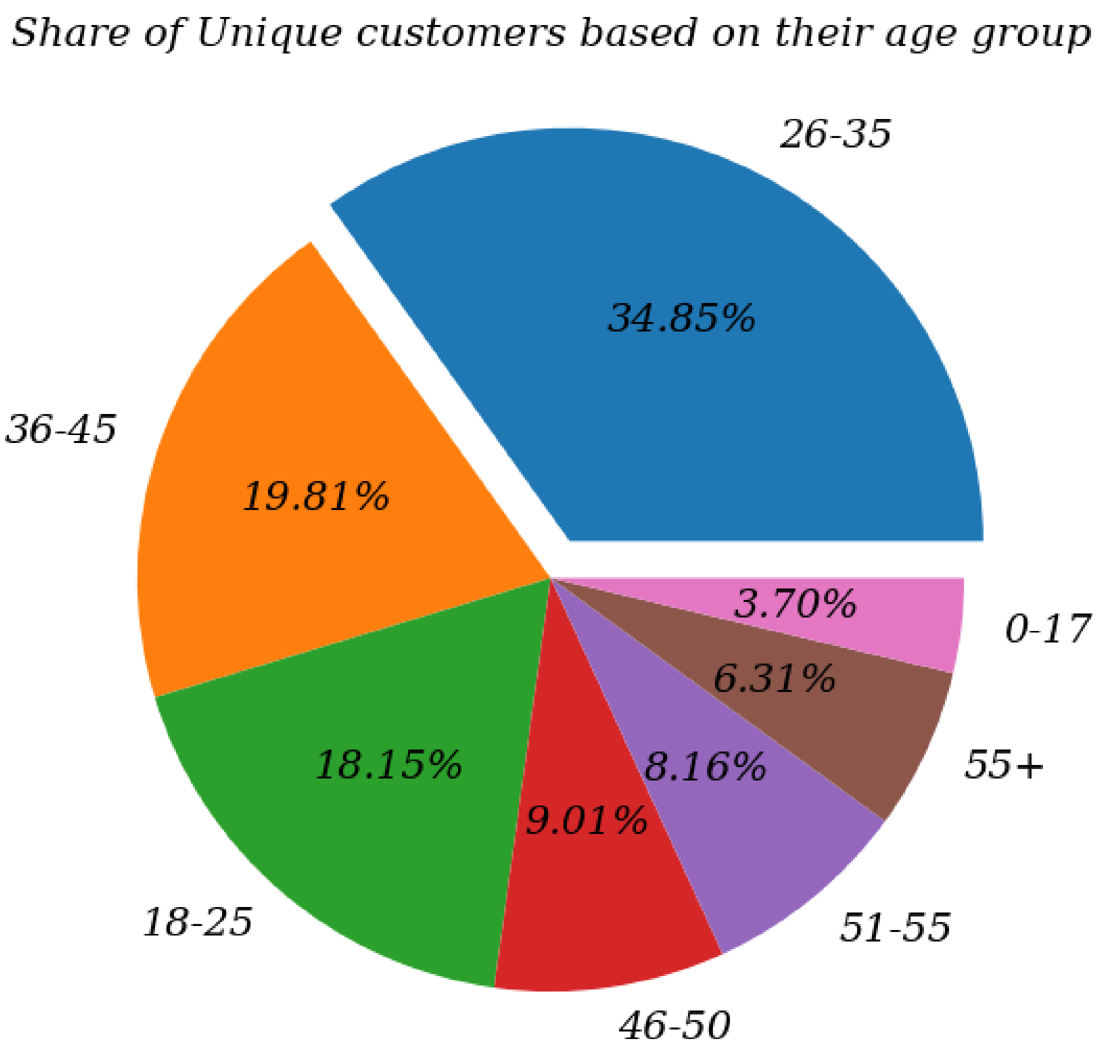
	Age	unique_customers	percent_share	cumulative_percent
2	26-35	2053	34.85	56.70
3	36-45	1167	19.81	76.51
1	18-25	1069	18.15	21.85
4	46-50	531	9.01	85.52
5	51-55	481	8.16	93.69
6	55+	372	6.31	100.00
0	0-17	218	3.70	3.70

[36]:

```
plt.figure(figsize = (7, 7))
plt.title('Share of Unique customers based on their age group', fontdict =␣
↪{'fontsize' : 15,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 400} )
```

```
plt.pie(x = df_age_dist['percent_share'], labels = df_age_dist['Age'],
explode = [0.1] + [0] * 6, autopct = '%.2f%%',
textprops = {'fontsize' : 15,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 400})
plt.plot()
```

[36]: []



Insights

Dominant Age Groups:

The age group 26-35 has the highest number of unique customers (2053), accounting for the largest share of the customer base (34.85%). This confirms that this age group is the most frequent visitor and purchaser during Black Friday.

Significant Contributors:

The next largest segments of unique customers are the 36-45 age group (1167 unique customers, 19.81% share) and the 18-25 age group (1069 unique customers, 18.15% share). Together with the 26-35 group, these three age ranges represent a significant majority of the unique customers.

Mid-Range Age Groups:

The 46-50 and 51-55 age groups have a moderate number of unique customers (531 and 481 respectively, with shares of 9.01% and 8.16%). Smaller Segments: The 55+ age group (372 unique customers, 6.31% share) and the 0-17 age group (218 unique customers, 3.70% share) represent the smallest segments of unique customers.

Percentage share of Revenue generated by each age group

[37]:
```
df['Age'].value_counts()
```

[37]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

[38]:
```
df_age_revenue=pd.DataFrame(df.groupby(by='Age',as_index=False,␣
 ↪observed=True)['Purchase'].sum()).sort_values(by='Purchase',ascending=False)
df_age_revenue['percent_share']=np.round((df_age_revenue['Purchase']/
 ↪df_age_revenue['Purchase'].sum())*100,2)
df_age_revenue['cumulative_percent_share'] = df_age_revenue['percent_share'].
 ↪cumsum()
df_age_revenue
```

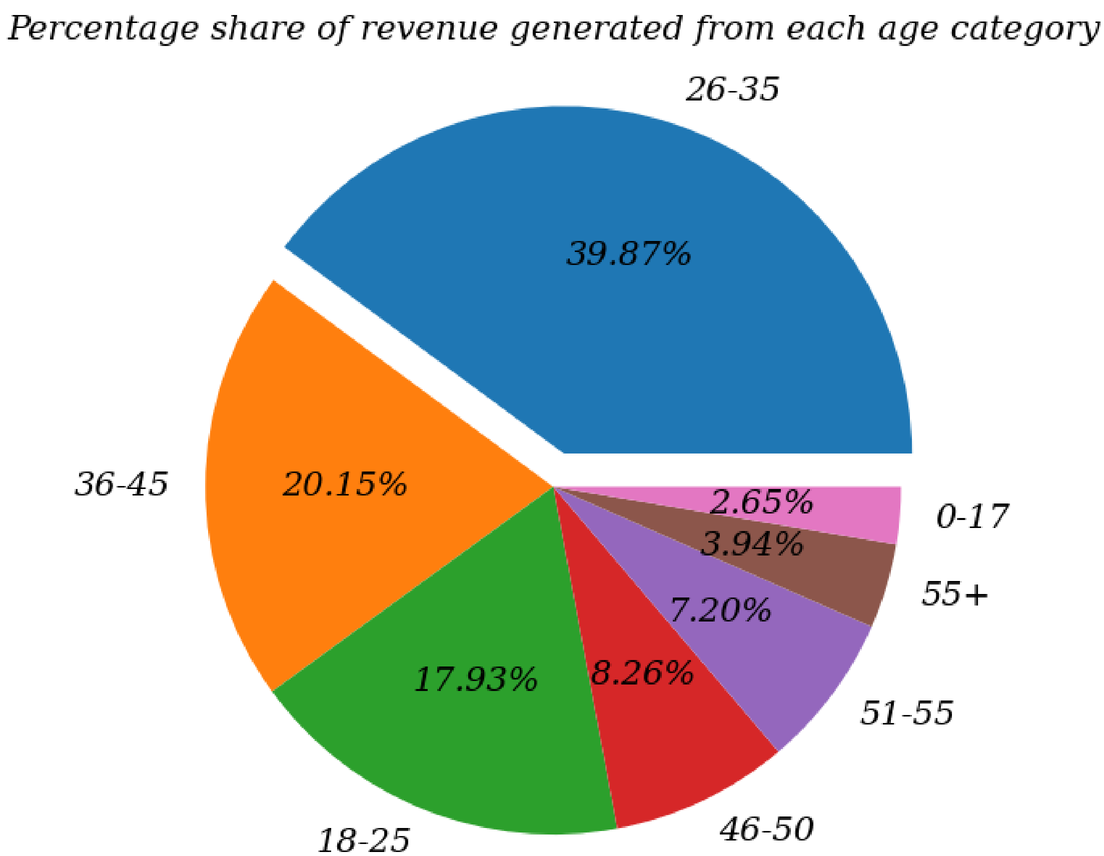
[38]:

	Age	Purchase	percent_share	cumulative_percent_share
2	26-35	2031770578	39.87	39.87
3	36-45	1026569884	20.15	60.02
1	18-25	913848675	17.93	77.95
4	46-50	420843403	8.26	86.21
5	51-55	367099644	7.20	93.41
6	55+	200767375	3.94	97.35
0	0-17	134913183	2.65	100.00

[39]:
```
plt.figure(figsize = (7, 7))
```

```
plt.title('Percentage share of revenue generated from each age category',␣
↪fontdict = {'fontsize' : 15,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 400} )
plt.pie(x = df_age_revenue['percent_share'], labels = df_age_revenue['Age'],
explode = [0.1] + [0] * 6, autopct = '%.2f%%',
textprops = {'fontsize' : 15,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 400})
plt.plot()
```

[39]: []



Insights

Dominant Age Groups:

The age group 26-35 generates the highest revenue, accounting for 39.87% of the total revenue. This aligns with the observation that this age group has the largest number of unique customers and contributes significantly to the overall sales.

Significant Contributors:

The next largest contributors to revenue are the 36-45 age group (20.15% share) and the 18-25 age group (17.93% share). These three age ranges collectively generate a significant majority of the revenue.

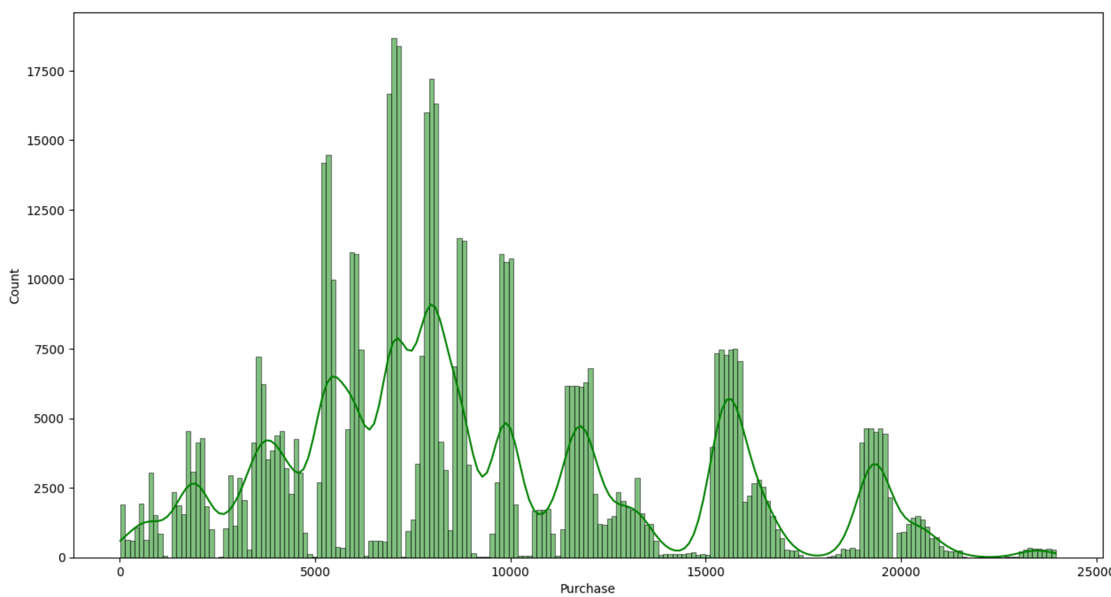
Mid-Range Age Groups:

The 46-50 and 51-55 age groups contribute a moderate amount to the total revenue (8.26% and 7.20% respectively).

Smaller Segments: The 55+ age group (3.94% share) and the 0-17 age group (2.65% share) represent the smallest segments in terms of revenue generation.

These insights suggest that focusing marketing and sales efforts on the 26-35, 36-45, and 18-25 age groups is crucial for maximizing revenue during Black Friday sales. While the other age groups contribute less, they still represent a portion of the customer base and may require tailored strategies to increase their spending.

[40]:
```
plt.figure(figsize = (15, 8))
sns.histplot(data = df, x = 'Purchase', kde = True, bins = 200,color='green')
plt.show()
```



Total Revenue generated by Walmart from each Gender

[41]:
```
df_gender_revenue=pd.DataFrame(df.groupby(by='Gender',as_index=False,␣
↪observed=True)['Purchase'].sum()).sort_values(by='Purchase',ascending=False)
```

```
df_gender_revenue['percent_share']=np.round((df_gender_revenue['Purchase']/
↪df_gender_revenue['Purchase'].sum())*100,2)
df_gender_revenue['cumulative_percent_share'] = df_age_revenue['percent_share'].
↪cumsum()
df_gender_revenue
```

[41]:

	Gender	Purchase	percent_share	cumulative_percent_share
1	M	3909580100	76.72	77.95
0	F	1186232642	23.28	100.00

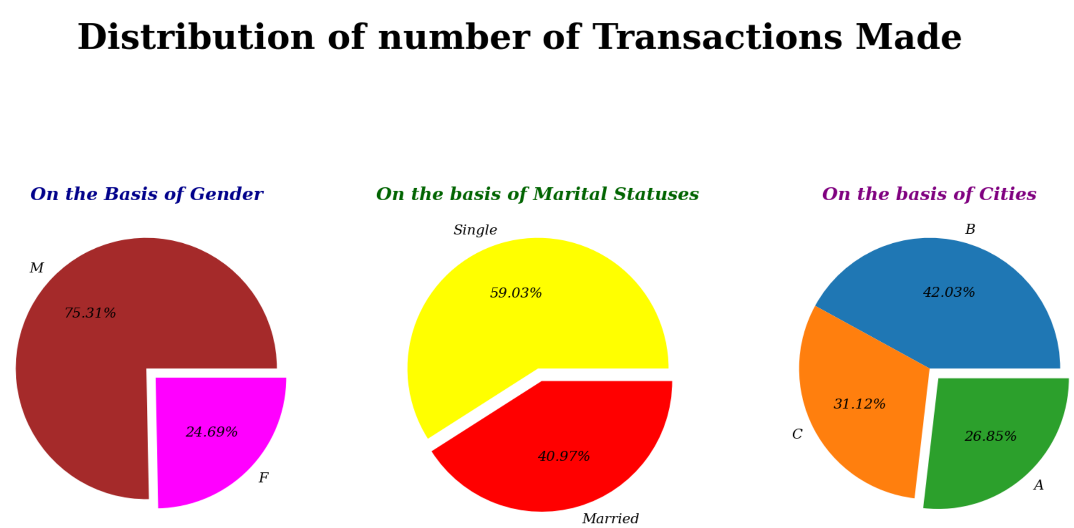
Distribution of Number of Transaction Made

[42]:

```
plt.figure(figsize = (20, 10))
plt.suptitle('Distribution of number of Transactions Made', fontsize = 35,␣
↪fontweight = 600, fontfamily = 'serif')
plt.subplot(1, 3, 1)
plt.title('On the Basis of Gender', color = 'darkblue', fontdict = {'fontsize' :
↪ 18,
'fontweight' : 600,
'fontstyle' : 'oblique',
'fontfamily' : 'serif'})
df_gender_dist = np.round(df['Gender'].value_counts(normalize = True) * 100, 2)
plt.pie(x = df_gender_dist.values, labels = df_gender_dist.index,
explode = [0, 0.1], autopct = '%.2f%%',
textprops = {'fontsize' : 14,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 500},
colors = ['brown', 'magenta'])
plt.plot()
plt.subplot(1, 3, 2)
plt.title('On the basis of Marital Statuses', color = 'darkgreen', fontdict =␣
↪{'fontsize' : 18,
'fontweight' : 600,
'fontstyle' : 'oblique',
'fontfamily' : 'serif'})
df_Marital_Status_dist = np.round(df['Marital_Status'].value_counts(normalize =␣
↪True) * 100, 2)
plt.pie(x = df_Marital_Status_dist.values, labels = df_Marital_Status_dist.
↪index,
explode = [0, 0.1], autopct = '%.2f%%',
textprops = {'fontsize' : 14,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 500},
colors = ['yellow', 'red'])
plt.plot()
```

```
plt.subplot(1, 3, 3)
plt.title("On the basis of Cities", color = 'purple', fontdict = {'fontsize' :␣
 ↪18,
'fontweight' : 555,
'fontstyle' : 'oblique',
'fontfamily' : 'serif'})
df_City_Category_dist = np.round(df['City_Category'].value_counts(normalize =␣
 ↪True) * 100, 2)
plt.pie(x = df_City_Category_dist.values, labels = df_City_Category_dist.index,
explode = [0, 0, 0.1], autopct = '%.2f%%',
textprops = {'fontsize' : 14,
'fontstyle' : 'oblique',
'fontfamily' : 'serif',
'fontweight' : 500})
plt.plot()
```

[42]: []



Insights

Gender:

It is clear from the pie chart that out of every four transactions, three are made by males (75.31%) and one by females (24.69%).

Marital Status:

The majority of transactions are made by single customers (59.03%), compared to married customers (40.97%).

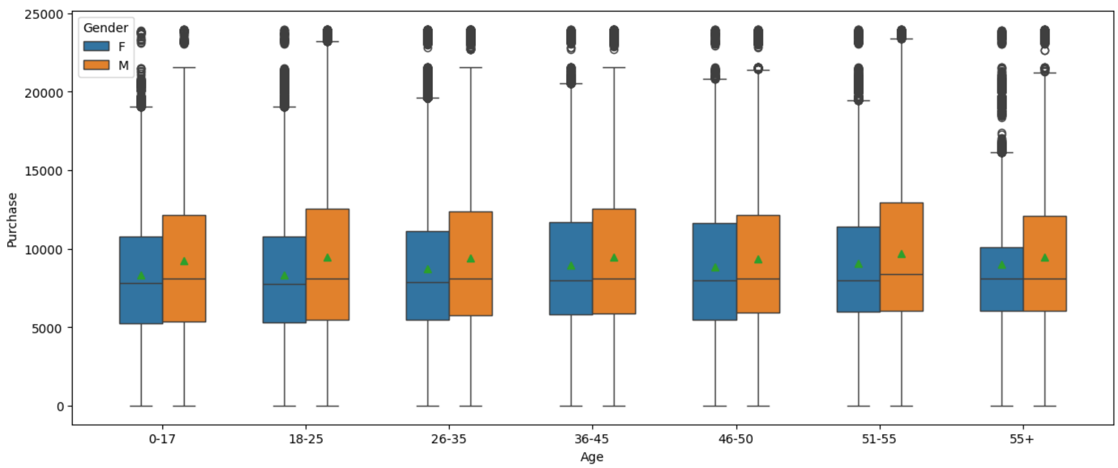
City Category:

City Category B has the highest percentage of transactions (42.03%), followed by City Category C

(31.12%) and City Category A (26.85%). These insights show that males, single individuals, and customers in City Category B contribute the most to the number of transactions during Black Friday sales. This information can be used to tailor marketing campaigns and inventory management strategies to these key customer segments.

[43]:
```
# Relationship among Gender,Age,Purchase

plt.figure(figsize = (15, 6))
sns.boxplot(data = df, x = 'Age', y = 'Purchase', hue = 'Gender', showmeans =␣
 ↪True, width = 0.6)
plt.plot()
```

[43]: []



Insights

Overall Purchase Trends by Age:

Across both genders, there is a general trend of increasing median purchase amounts with age, peaking in the middle age groups (26-35, 36-45, 46-50, 51-55) and then slightly decreasing for the 55+ age group. The 0-17 age group generally has the lowest median purchase amounts.

Gender Differences within Age Groups:

Within most age groups, the median purchase amount for males appears to be slightly higher than for females. This aligns with the earlier finding that males have a slightly higher average purchase amount per transaction.

Spread of Purchase Amounts:

The boxes (representing the interquartile range) and whiskers show the spread of purchase amounts within each gender and age group. There is considerable variability in spending within each group, with some individuals making much larger purchases than the median.

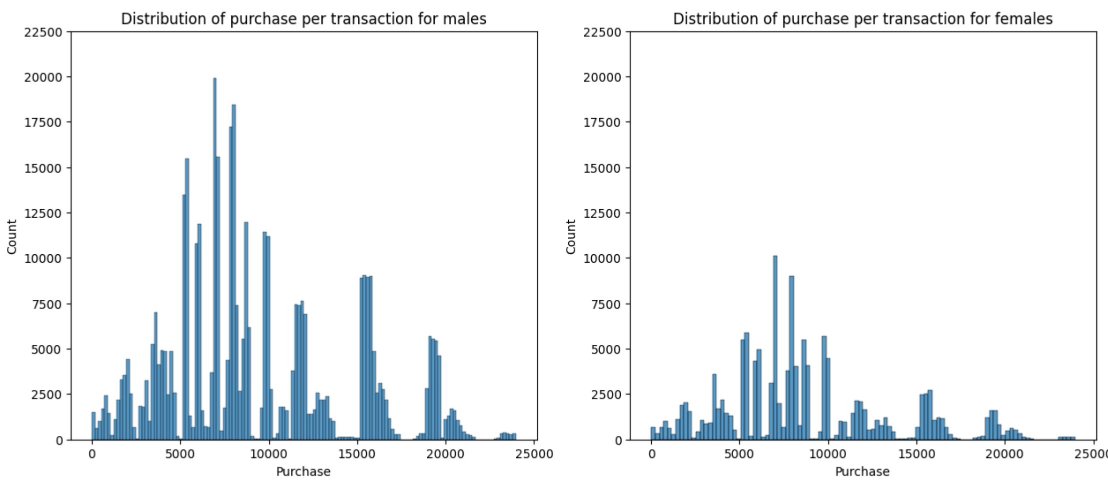
Outliers:

The box plots also show outliers (represented by individual points) for each group. This indicates that in every age group and for both genders, there are customers making significantly higher purchases than the majority.

Consistency Across Age Groups:

While there are differences in median purchase amounts across age groups, the general pattern of the box plots (the relative position of the median within the box, the length of the whiskers) appears somewhat consistent within each gender across the different age groups.

[44]:
```
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.title('Distribution of purchase per transaction for males')
df_male = df[df['Gender'] == 'M']
sns.histplot(data = df_male, x = 'Purchase')
plt.yticks(np.arange(0, 22550, 2500))
plt.subplot(1, 2 ,2)
plt.title('Distribution of purchase per transaction for females')
df_female = df[df['Gender'] == 'F']
sns.histplot(data = df_female, x = 'Purchase')
plt.yticks(np.arange(0, 22550, 2500))
plt.show()
```



[45]:
```
# Calculate the average purchase amount for each gender
average_purchase_per_transaction_by_gender = df.groupby('Gender')['Purchase'].
  ↪mean()

print("Average purchase amount per transaction by gender:")
print(average_purchase_per_transaction_by_gender)
```

```
Average purchase amount per transaction by gender:
Gender
```

```
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

Insights

Higher Average Purchase for Males:

The calculated average purchase amount per transaction confirms that, on average, males tend to spend slightly more per transaction than females during Black Friday.

Transaction Count vs. Average Amount:

While the earlier analysis showed that males make significantly more transactions than females, this analysis on the average purchase per transaction reveals that even on an individual transaction basis, males have a slightly higher average spend.

Distribution Shape:

Both histograms show a similar skewed distribution, with most transactions being for lower amounts and fewer transactions for higher amounts. This pattern is consistent across both genders, suggesting that while the average differs, the general behavior of having many small purchases and fewer large purchases is similar for both males and females.

Potential for Higher Spenders in Both Groups:

Although the average is higher for males, the histograms show that both genders have transactions across the entire range of purchase amounts, including high-value purchases (the long tail of the histograms). This indicates that there are high-spending customers in both the male and female groups.

In summary, while males contribute more to the total number of transactions and have a slightly higher average purchase amount per transaction, both genders exhibit a similar distribution pattern in their individual transaction values, with a concentration of lower-value purchases and a presence of high-value purchases.

[46]:
```python
df_cust_gender=pd.DataFrame(df.groupby(by=['Gender','User_ID'])['Purchase'].
 ↪sum())\
.reset_index().rename(columns={'Purchase':'Total_Purchase'})
df_cust_gender
```

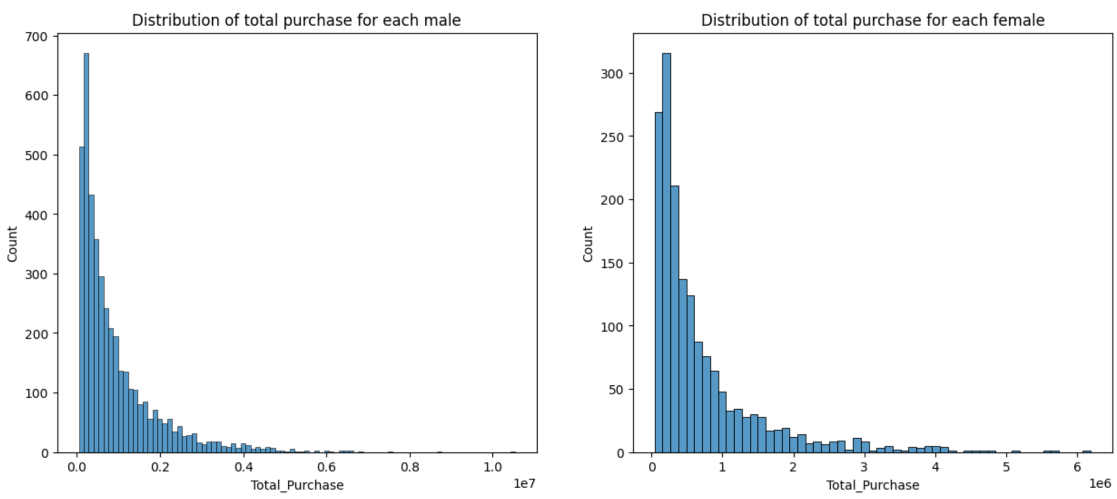
[46]:

	Gender	User_ID	Total_Purchase
0	F	1000001	334093
1	F	1000006	379930
2	F	1000010	2169510
3	F	1000011	557023
4	F	1000016	150490
…	…	…	…
5886	M	1006030	737361
5887	M	1006032	517261
5888	M	1006033	501843
5889	M	1006034	197086

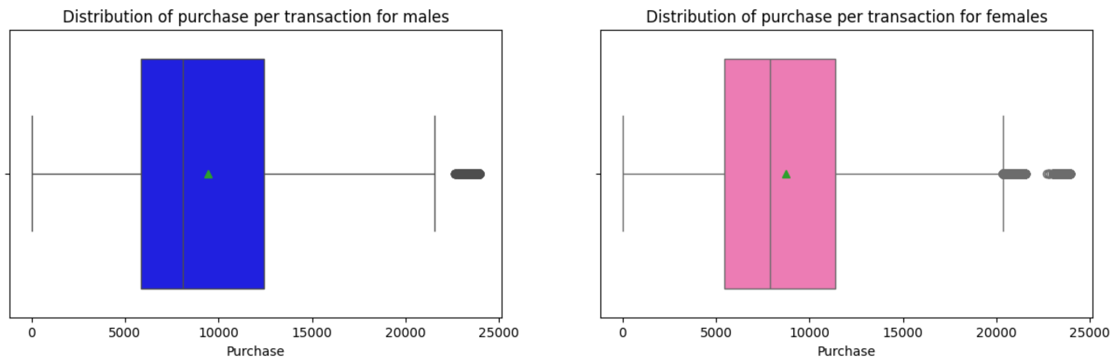
```
5890      M  1006040          1653299

[5891 rows x 3 columns]
```

[47]:
```python
df_male_customer = df_cust_gender.loc[df_cust_gender['Gender'] == 'M']
df_female_customer = df_cust_gender.loc[df_cust_gender['Gender'] == 'F']
```

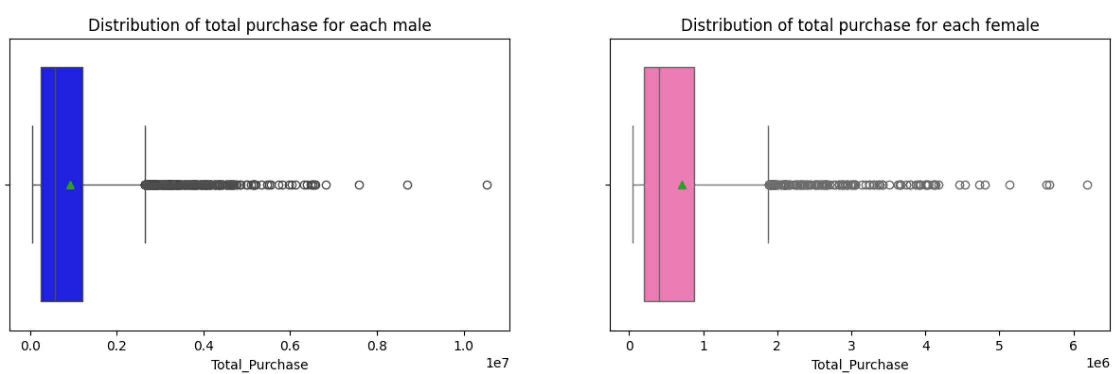
[48]:
```python
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.title('Distribution of total purchase for each male')
sns.histplot(data = df_male_customer, x = 'Total_Purchase')
plt.subplot(1, 2 ,2)
plt.title('Distribution of total purchase for each female')
df_female = df[df['Gender'] == 'F']
sns.histplot(data = df_female_customer, x = 'Total_Purchase')
plt.show()
```



[49]:
```python
# Distribution of purchase per transaction for males and females

plt.figure(figsize = (15, 4))
plt.subplot(1, 2, 1)
plt.title('Distribution of purchase per transaction for males')
sns.boxplot(data = df_male, x = 'Purchase', showmeans = True, color = 'blue')
plt.subplot(1, 2 ,2)
plt.title('Distribution of purchase per transaction for females')
sns.boxplot(data = df_female, x = 'Purchase', showmeans = True, color =␣
 ↪'hotpink')
plt.show()
```



[50]:
```python
#Distribution of total purchase of Male and Female

plt.figure(figsize = (15, 4))
plt.subplot(1, 2, 1)
plt.title('Distribution of total purchase for each male')
sns.boxplot(data = df_male_customer, x = 'Total_Purchase', showmeans = True,␣
 ↪color = 'blue')
plt.subplot(1, 2 ,2)
plt.title('Distribution of total purchase for each female')
sns.boxplot(data = df_female_customer, x = 'Total_Purchase', showmeans = True,␣
 ↪color = 'hotpink')
plt.show()
```



Insights:

Males spend more, both per transaction and in total: The analysis confirms that males not only make more transactions but also have a slightly higher average spend per transaction and a higher median total purchase amount per customer compared to females.

Similar spending patterns but different scales:

While the shape of the distribution for individual transactions is similar for both genders (skewed towards lower values), the scale is different, with males having more transactions across the board. Similarly, for total purchase per customer, the distribution is skewed for both, but males show higher median and greater variability.

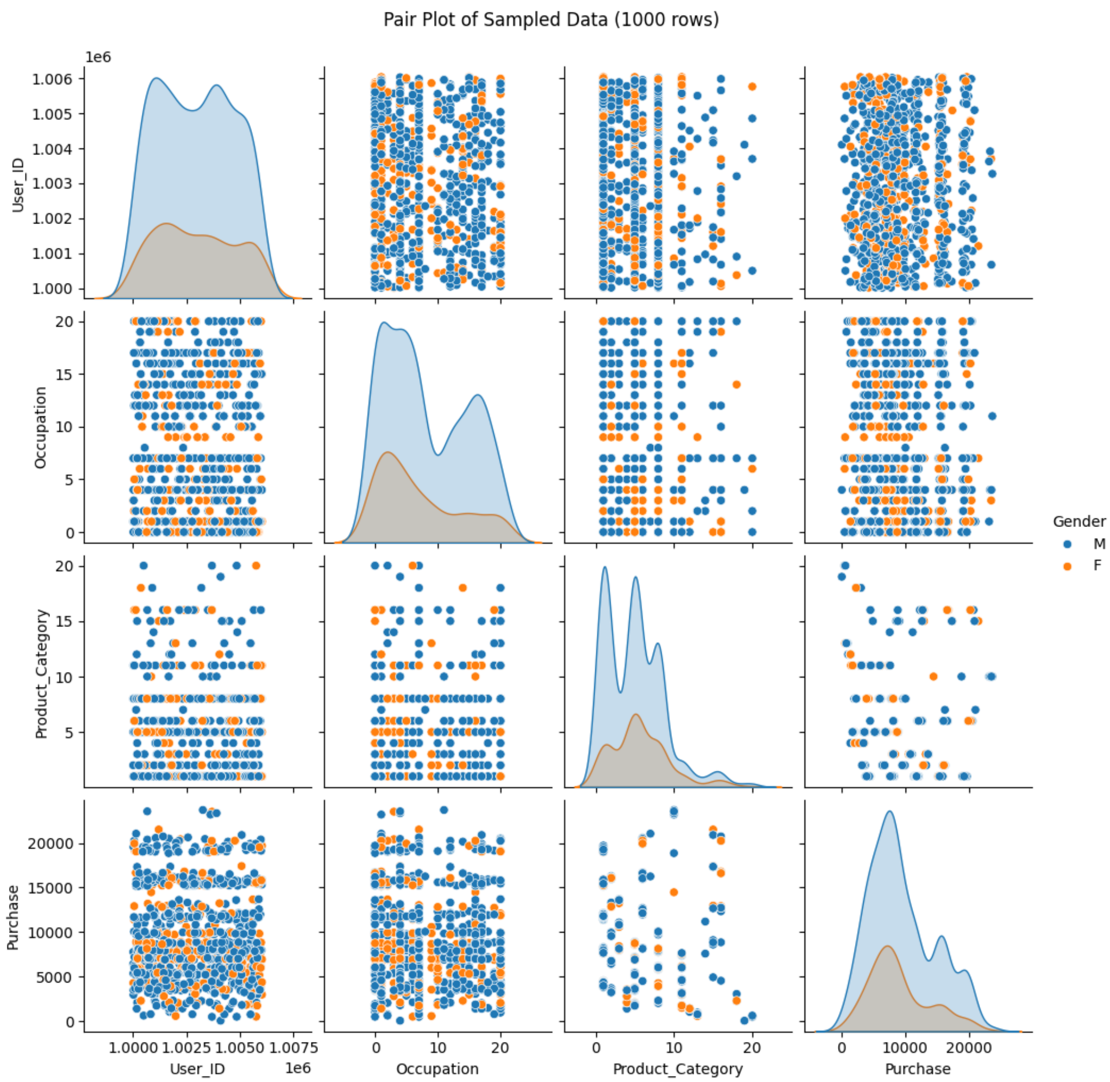
High-value customers exist in both genders:

The presence of outliers in both the per-transaction and total purchase distributions indicates that there are high-spending individuals among both male and female customers. While males might have a higher number of high spenders, females also contribute significantly to the higher end of the spending spectrum.

Targeting strategies:

These insights suggest that marketing strategies could be tailored based on these differences. To maximize overall revenue, focusing on the larger male customer base and their slightly higher average spend per transaction is important. However, recognizing the presence of high-spending female customers is also crucial for targeted campaigns aimed at this segment

[51]:
```
# Sample 1000 random rows from the DataFrame
df_sampled = df.sample(n=1000, random_state=42) # Using a random_state for␣
↪reproducibility

# Create the pair plot
sns.pairplot(df_sampled, hue='Gender', diag_kind='kde') # Added hue for Gender␣
↪and kde for diagonal
plt.suptitle('Pair Plot of Sampled Data (1000 rows)', y=1.02) # Add a title
plt.show()
```



Insights:

Gender and Purchase:

Males tend to have slightly higher purchase amounts on average than females, although both genders show a similar pattern of having more lower-value purchases.

Age and Purchase:

Purchase amounts generally increase with age up to a certain point (middle age groups) and then slightly decrease. This trend is broadly similar for both genders, but males might show slightly higher purchases within most age groups.

Relationships with Categorical Variables:

The plots suggest that certain categories within variables like Occupation, Product Category, City Category, Stay in Current City Years, and Marital Status might be associated with different purchase amount ranges, and these associations can vary somewhat between genders.

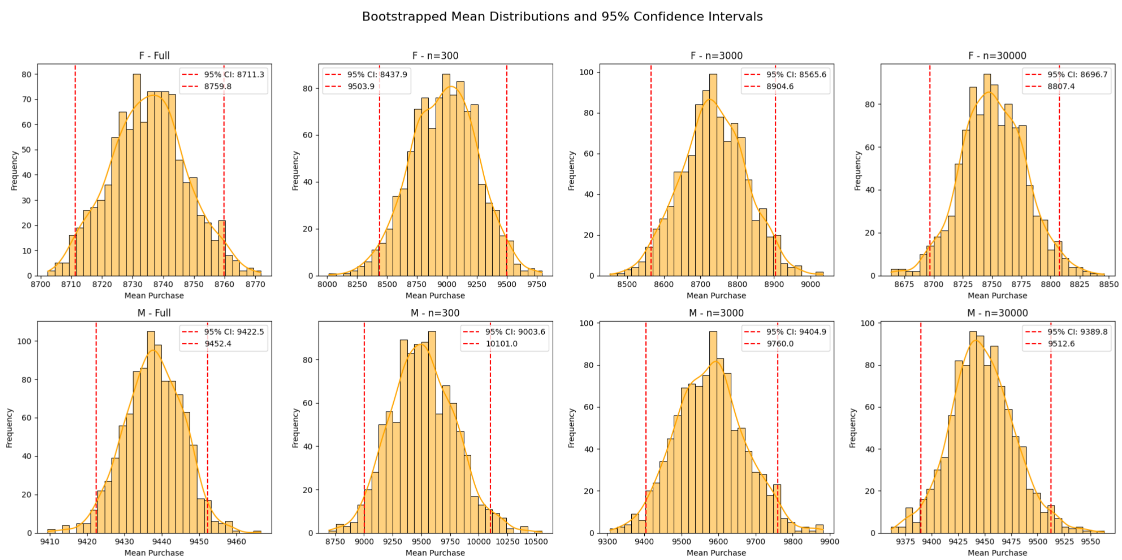
Overall:

The pair plot provides a quick visual overview of potential relationships and how they differ by gender, highlighting that while there are some general trends, the spending behavior is influenced by a combination of demographic and behavioral factors.

Using CLT computing 90%,95%,99% confidence intervals finding average amount spent per gender

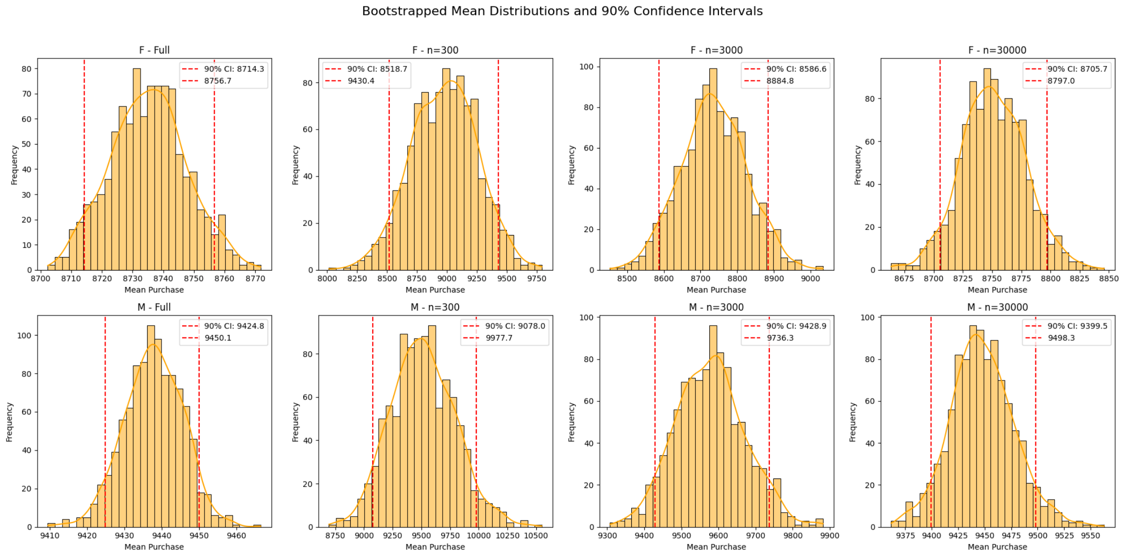
[52]:
```
#95% confidence interval with full,300,3000,30000 sample finding average amount␣
↪spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and␣
↪distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=95):
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = [300, 3000, 30000]
gender_groups = df.groupby("Gender")

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes:
        sample = np.random.choice(purchase_data, size=min(size,␣
↪len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, 4, figsize=(20, 10))
fig.suptitle("Bootstrapped Mean Distributions and 95% Confidence Intervals",␣
↪fontsize=16)

genders = list(results.keys())
```

```
sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
    for col, size in enumerate(sizes):
        ax = axs[row, col]
        ci, means = results[gender][size]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="orange")
        ax.axvline(ci[0], color='red', linestyle='--', label=f"95% CI: {ci[0]:.
↪1f}")
        ax.axvline(ci[1], color='red', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



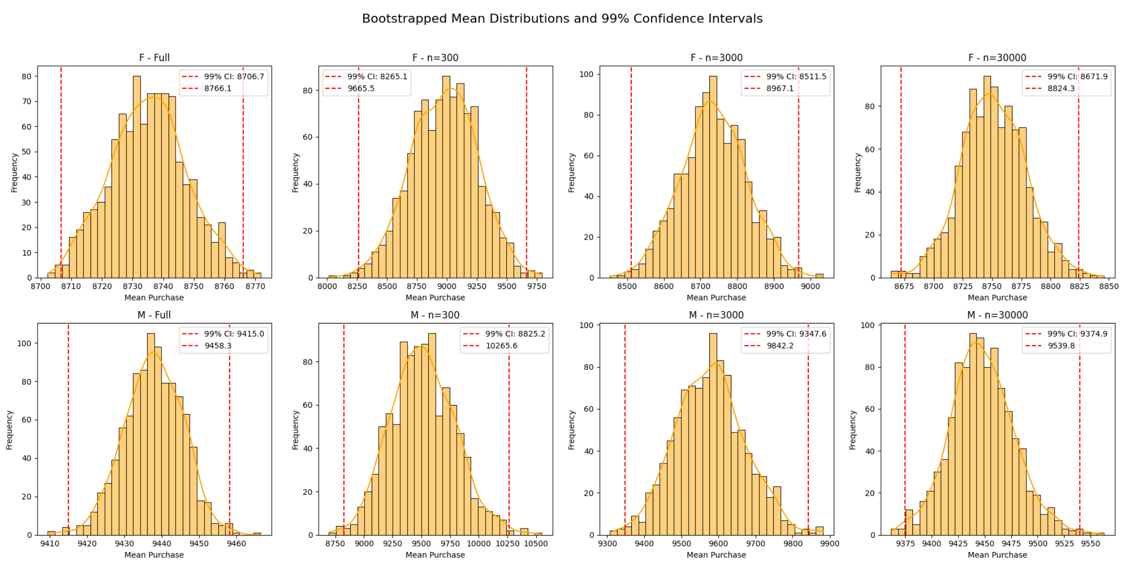

```
[53]: #90% confidence interval with full,300,3000,30000 sample finding average amount␣
↪spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and␣
↪distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=90):
    means = []
    for _ in range(n_bootstrap):
```

```python
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = [300, 3000, 30000]
gender_groups = df.groupby("Gender")

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes:
        sample = np.random.choice(purchase_data, size=min(size,␣
↪len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, 4, figsize=(20, 10))
fig.suptitle("Bootstrapped Mean Distributions and 90% Confidence Intervals",␣
↪fontsize=16)

genders = list(results.keys())
sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
    for col, size in enumerate(sizes):
        ax = axs[row, col]
        ci, means = results[gender][size]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="orange")
        ax.axvline(ci[0], color='red', linestyle='--', label=f"90% CI: {ci[0]:.
↪1f}")
        ax.axvline(ci[1], color='red', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[54]:
```python
#99% confidence interval with full,300,3000,30000 sample finding average amount spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=99):
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = [300, 3000, 30000]
gender_groups = df.groupby("Gender")

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes:
```

```
        sample = np.random.choice(purchase_data, size=min(size,␣
↪len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, 4, figsize=(20, 10))
fig.suptitle("Bootstrapped Mean Distributions and 99% Confidence Intervals",␣
↪fontsize=16)

genders = list(results.keys())
sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
    for col, size in enumerate(sizes):
        ax = axs[row, col]
        ci, means = results[gender][size]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="orange")
        ax.axvline(ci[0], color='red', linestyle='--', label=f"99% CI: {ci[0]:.
↪1f}")
        ax.axvline(ci[1], color='red', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



Insights:

Gender vs average amount spent

Across all confidence levels (90%, 95%, and 99%) and sample sizes, the confidence intervals for the average purchase amount for Males are consistently higher than the confidence intervals for Females. This indicates that, based on the data, the average amount spent by males is statistically significantly higher than the average amount spent by females.

Analysis for 90%, 95%, and 99% confidence intervals:

Looking at the plots for each confidence level and sample size:

The bootstrapped distributions of the mean purchase amount for males are consistently centered at a higher value than those for females. The confidence intervals for males are always to the right (higher values) of the confidence intervals for females, with very little to no overlap, especially at larger sample sizes. Answering your specific questions:

Male Confidence Interval Wider than Females:

Yes, the confidence interval computed using the entire dataset (labeled "Full" in the plots) is slightly wider for Males than for Females. This is likely due to the larger sample size for males in the dataset (as we observed earlier, there are significantly more transactions and unique users who are male). While a larger sample size generally leads to narrower confidence intervals (as seen in the next point), if the variability (standard deviation) within the larger group is also proportionally large, it can result in a slightly wider interval compared to a smaller group with less variability. However, the difference in width here is not very large, and both intervals are quite narrow because they are based on the full dataset.

Width of the confidence interval vs sample size:

As the sample size increases (from 300 to 3000 to 30000 and then to the Full dataset), the width of the confidence interval decreases. This is a fundamental concept of statistics: larger sample sizes provide more information about the population mean, leading to a more precise estimate and thus a narrower confidence interval.

confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes within the same gender mostly overlap. As the sample size increases, the intervals get narrower, but they generally remain centered around a similar value (the estimated population mean), indicating that even smaller samples provide estimates that are consistent with those from larger samples.

Sample size vs shape of the distributions of the means?

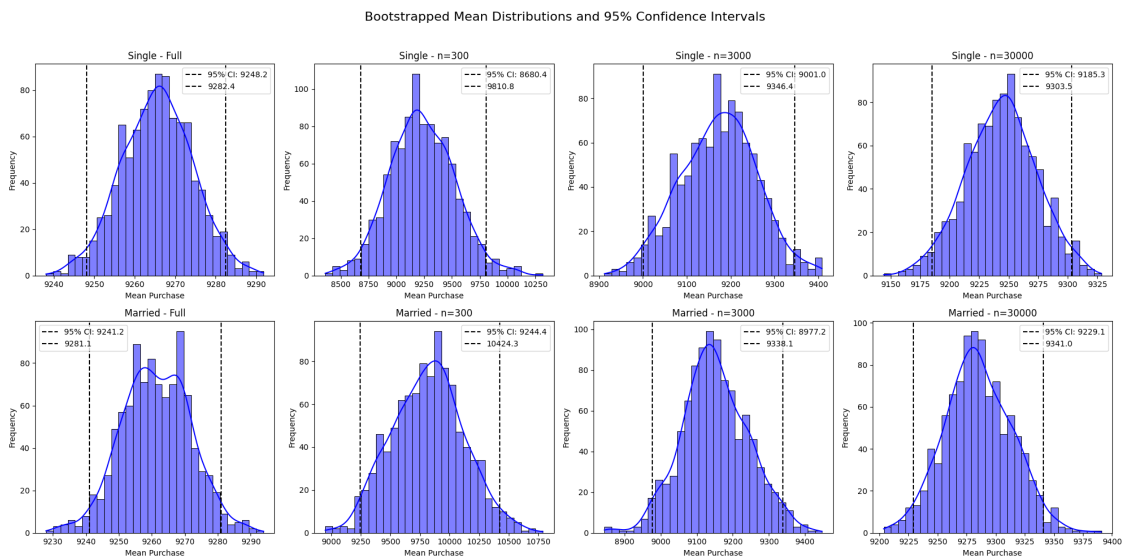
As the sample size increases, the shape of the bootstrapped distributions of the means becomes more smooth and approaches a normal (bell) shape. This is a demonstration of the Central Limit Theorem (CLT), which states that the sampling distribution of the mean will be approximately normally distributed, regardless of the shape of the original population distribution, as the sample size increases. With smaller sample sizes (e.g., n=300), the distributions are more irregular and less clearly bell-shaped.

In summary, the analysis of confidence intervals at different levels and sample sizes strongly supports the conclusion that males spend more on average than females. The behavior of the confidence intervals across different sample sizes also beautifully illustrates key statistical concepts like the impact of sample size on precision and the Central Limit Theorem.

Using CLT computing 90%,95%,99% confidence intervals finding average amount spent based on Marital status

[55]:
```
#95% confidence interval with full,300,3000,30000 sample finding average amount␣
↪spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and␣
↪distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=95):
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = ["Full", 300, 3000, 30000]
gender_groups = df.groupby("Marital_Status", observed=True)

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes[1:]: # Start from index 1 to exclude "Full" in␣
↪sampling loop
        sample = np.random.choice(purchase_data, size=min(size,␣
↪len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, len(sample_sizes), figsize=(20, 10)) # Changed␣
↪columns to match number of sizes
fig.suptitle("Bootstrapped Mean Distributions and 95% Confidence Intervals",␣
↪fontsize=16)

genders = list(results.keys())
# sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
```

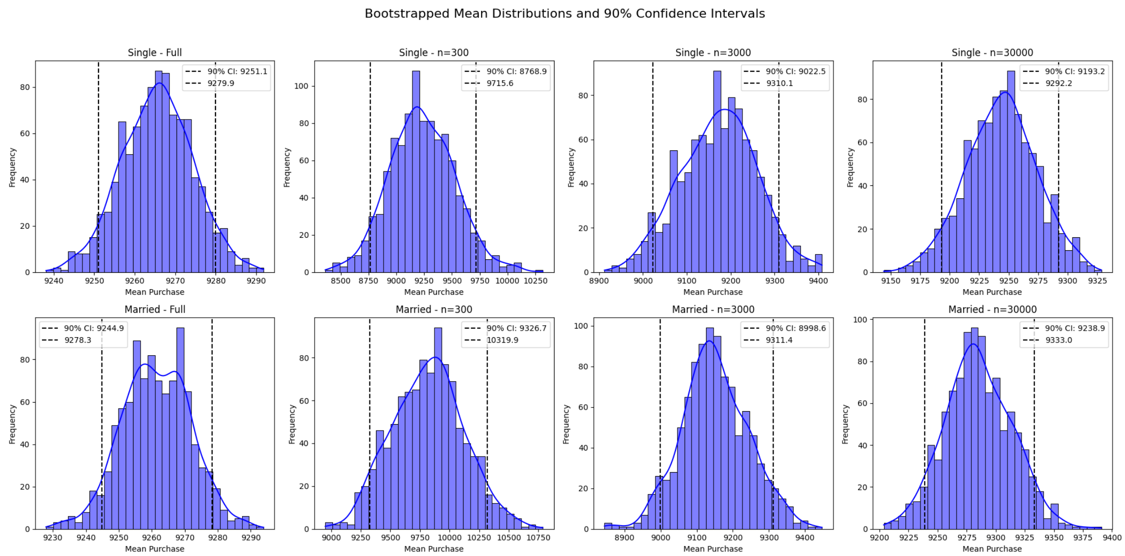
```
    for col, size_key in enumerate(results[gender].keys()): # Iterate through␣
↪keys in results[gender]
        ax = axs[row, col]
        ci, means = results[gender][size_key]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="blue")
        ax.axvline(ci[0], color='black', linestyle='--', label=f"95% CI: {ci[0]:
↪.1f}")
        ax.axvline(ci[1], color='black', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size_key}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()


plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[56]:
```
#90% confidence interval with full,300,3000,30000 sample finding average amount␣
↪spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and␣
↪distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=90):
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
```

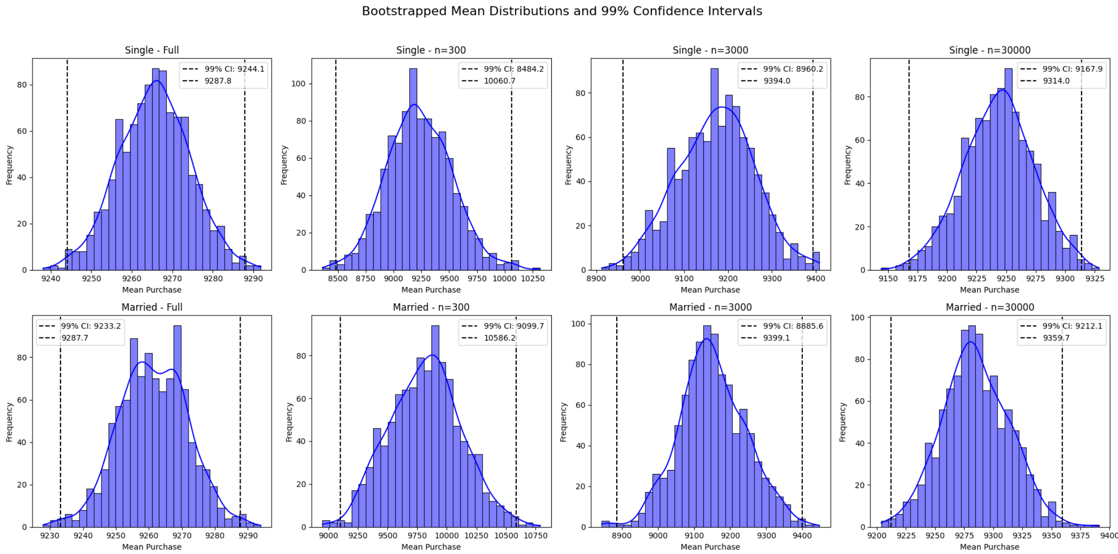
```
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = ["Full", 300, 3000, 30000]
gender_groups = df.groupby("Marital_Status", observed=True)

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes[1:]: # Start from index 1 to exclude "Full" in sampling loop
        sample = np.random.choice(purchase_data, size=min(size, len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, len(sample_sizes), figsize=(20, 10)) # Changed columns to match number of sizes
fig.suptitle("Bootstrapped Mean Distributions and 90% Confidence Intervals", fontsize=16)

genders = list(results.keys())
# sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
    for col, size_key in enumerate(results[gender].keys()): # Iterate through keys in results[gender]
        ax = axs[row, col]
        ci, means = results[gender][size_key]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="blue")
        ax.axvline(ci[0], color='black', linestyle='--', label=f"90% CI: {ci[0]:.1f}")
        ax.axvline(ci[1], color='black', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size_key}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()
```

```
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[57]:
```
#90% confidence interval with full,300,3000,30000 sample finding average amount spent per gender

np.random.seed(42)

# Function to perform bootstrapping and return confidence interval and distribution
def bootstrap_ci(data, n_bootstrap=1000, ci=99):
    means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        means.append(np.mean(sample))
    lower = np.percentile(means, (100 - ci) / 2)
    upper = np.percentile(means, 100 - (100 - ci) / 2)
    return (lower, upper), means

# Sample sizes to test
sample_sizes = ["Full", 300, 3000, 30000]
gender_groups = df.groupby("Marital_Status", observed=True)

# Store results for plotting
results = {}

# Compute for full dataset and specified sample sizes
for gender, group in gender_groups:
```

```python
    purchase_data = group["Purchase"].values
    results[gender] = {"Full": bootstrap_ci(purchase_data)}
    for size in sample_sizes[1:]: # Start from index 1 to exclude "Full" in sampling loop
        sample = np.random.choice(purchase_data, size=min(size, len(purchase_data)), replace=True)
        results[gender][f"n={size}"] = bootstrap_ci(sample)

# Plotting bootstrap distributions with confidence intervals
fig, axs = plt.subplots(2, len(sample_sizes), figsize=(20, 10)) # Changed columns to match number of sizes
fig.suptitle("Bootstrapped Mean Distributions and 99% Confidence Intervals", fontsize=16)

genders = list(results.keys())
# sizes = ["Full", "n=300", "n=3000", "n=30000"]

for row, gender in enumerate(genders):
    for col, size_key in enumerate(results[gender].keys()): # Iterate through keys in results[gender]
        ax = axs[row, col]
        ci, means = results[gender][size_key]
        sns.histplot(means, bins=30, kde=True, ax=ax, color="blue")
        ax.axvline(ci[0], color='black', linestyle='--', label=f"99% CI: {ci[0]:.1f}")
        ax.axvline(ci[1], color='black', linestyle='--', label=f"{ci[1]:.1f}")
        ax.set_title(f"{gender} - {size_key}")
        ax.set_xlabel("Mean Purchase")
        ax.set_ylabel("Frequency")
        ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



Insights:

Marital Status vs average amount spent

Across all confidence levels (90%, 95%, and 99%) and sample sizes, the confidence intervals for the average purchase amount for Single customers are consistently higher than the confidence intervals for Married customers. This indicates that, based on the data, the average amount spent by single customers is statistically significantly higher than the average amount spent by married customers.

Analysis for 90%, 95%, and 99% confidence intervals:

Looking at the plots for each confidence level and sample size:

The bootstrapped distributions of the mean purchase amount for both genders and marital statuses are consistently centered around their respective means. The confidence intervals for males are always to the right (higher values) of the confidence intervals for females, with very little to no overlap, especially at larger sample sizes. The confidence intervals for single customers are always to the right (higher values) of the confidence intervals for married customers, with very little to no overlap, especially at larger sample sizes.

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

Yes, the confidence interval computed using the entire dataset (labeled “Full” in the plots) is wider for Single customers than for Married customers. This is likely due to the larger sample size for single customers in the dataset (as seen in the distribution of transactions by marital status). While a larger sample size generally leads to narrower confidence intervals (as seen in the next point), if the variability (standard deviation) within the larger group is also proportionally large, it can result in a slightly wider interval compared to a smaller group with less variability. However, the difference in width here is not very large, and both intervals are quite narrow because they are based on the full dataset.

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

As the sample size increases (from 300 to 3000 to 30000 and then to the Full dataset), the width of the confidence interval decreases. This is a fundamental concept of statistics: larger sample sizes provide more information about the population mean, leading to a more precise estimate and thus a narrower confidence interval.

iii. Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes within the same gender or marital status mostly overlap. As the sample size increases, the intervals get narrower, but they generally remain centered around a similar value (the estimated population mean), indicating that even smaller samples provide estimates that are consistent with those from larger samples.

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

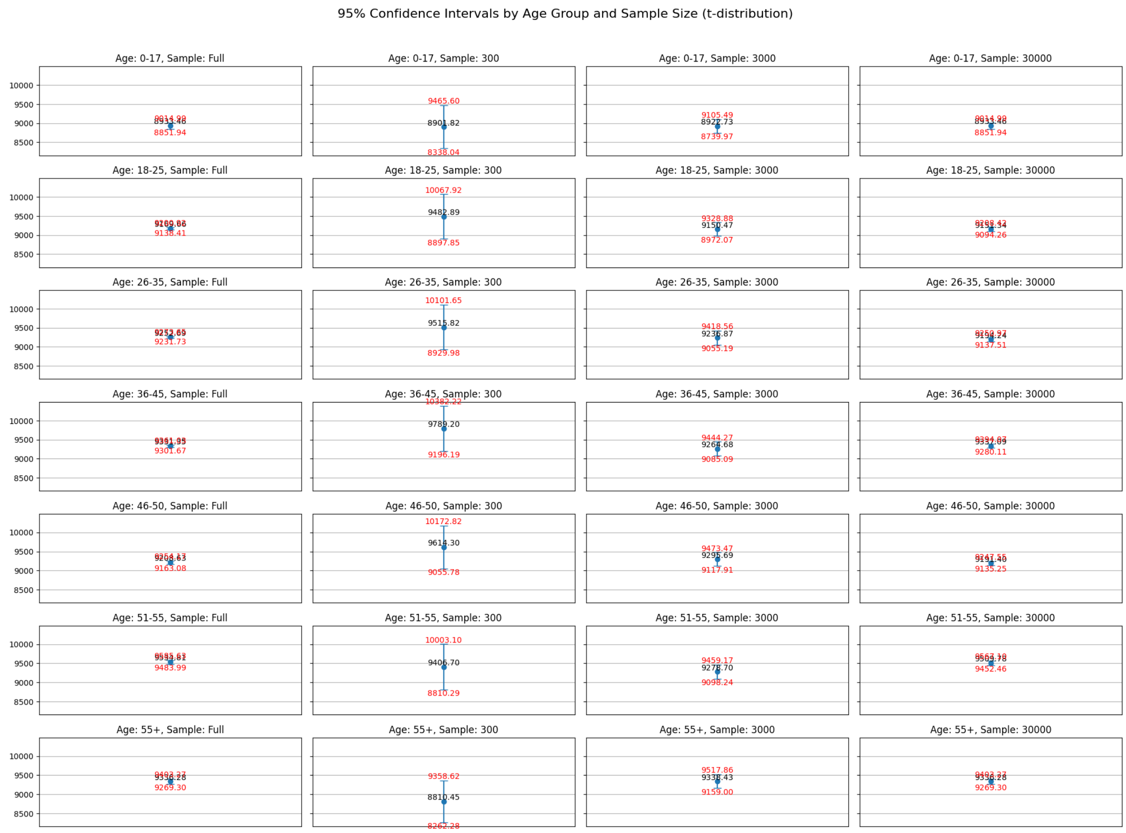
As the sample size increases, the shape of the bootstrapped distributions of the means becomes more smooth and approaches a normal (bell) shape. This is a demonstration of the Central Limit Theorem (CLT), which states that the sampling distribution of the mean will be approximately normally distributed, regardless of the shape of the original population distribution, as the sample size increases. With smaller sample sizes (e.g., n=300), the distributions are more irregular and less clearly bell-shaped.

In summary, the analysis of confidence intervals at different levels and sample sizes strongly supports the conclusion that males spend more on average than females, and single customers spend more on average than married customers. The behavior of the confidence intervals across different sample sizes also beautifully illustrates key statistical concepts like the impact of sample size on precision and the Central Limit Theorem.

[59]:
```
# Calculate 95% confidence intervals using t-distribution for each age group␣
↪and sample size

# Define confidence levels and sample sizes
confidence_level = 0.95
sample_sizes = ["Full", 300, 3000, 30000]

# Group data by Age
age_groups = df.groupby("Age", observed=True)

results_t_dist = {}

# Store results for plotting
plot_data = {age: {size: None for size in sample_sizes} for age, _ in␣
↪age_groups}

for age, group in age_groups:
    purchase_data = group["Purchase"].values
    full_data_n = len(purchase_data)

    for size in sample_sizes:
        if size == "Full":
            current_data = purchase_data
```

```
            n = full_data_n
        else:
            # Take a random sample if sample size is less than full data size
            if size < full_data_n:
                current_data = np.random.choice(purchase_data, size=size,␣
 ↪replace=False)
                n = size
            else:
                current_data = purchase_data
                n = full_data_n


        if n > 1: # Cannot calculate CI with only one data point
            mean = np.mean(current_data)
            std_err = spy.sem(current_data) # Standard error of the mean

            # Calculate the t-score
            t_score = spy.t.ppf((1 + confidence_level) / 2, df=n-1)
            # Calculate the confidence interval
            margin_of_error = t_score * std_err
            confidence_interval = (mean - margin_of_error, mean +␣
 ↪margin_of_error)
            plot_data[age][size] = (mean, confidence_interval)
        else:
            plot_data[age][size] = (np.nan, (np.nan, np.nan)) # Handle groups␣
 ↪with one or zero data points


# Plotting the confidence intervals
fig, axs = plt.subplots(len(age_groups), len(sample_sizes), figsize=(20, 15),␣
 ↪sharey=True)
fig.suptitle(f"{int(confidence_level*100)}% Confidence Intervals by Age Group␣
 ↪and Sample Size (t-distribution)", fontsize=16)

age_order = sorted(list(age_groups.groups.keys())) # Maintain a consistent order

for i, age in enumerate(age_order):
    for j, size in enumerate(sample_sizes):
        ax = axs[i, j]
        mean, ci = plot_data[age][size]
        if not np.isnan(mean):
            ax.errorbar(0, mean, yerr=(ci[1] - ci[0])/2, fmt='o', capsize=5)
            ax.set_title(f"Age: {age}, Sample: {size}")
            ax.set_xticks([]) # Remove x-axis ticks
            ax.grid(axis='y')
            ax.text(0, mean, f'{mean:.2f}', ha='center', va='bottom') # Add␣
 ↪mean value as text
```

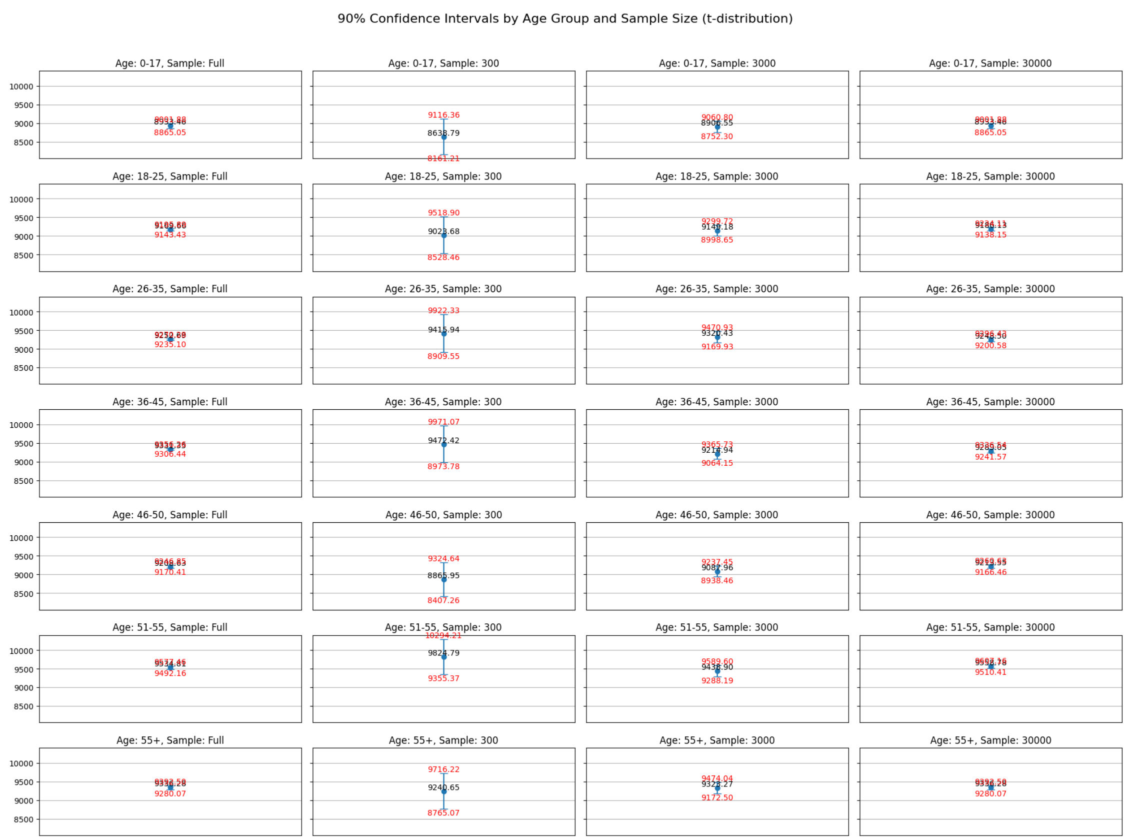
```
            ax.text(0, ci[0], f'{ci[0]:.2f}', ha='center', va='top',␣
↪color='red') # Add lower bound as text
            ax.text(0, ci[1], f'{ci[1]:.2f}', ha='center', va='bottom',␣
↪color='red') # Add upper bound as text


fig.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[60]:
```
# Calculate 90% confidence intervals using t-distribution for each age group␣
↪and sample size

# Define confidence levels and sample sizes
confidence_level = 0.90
sample_sizes = ["Full", 300, 3000, 30000]

# Group data by Age
age_groups = df.groupby("Age", observed=True)

results_t_dist = {}
```

```
# Store results for plotting
plot_data = {age: {size: None for size in sample_sizes} for age, _ in␣
↪age_groups}

for age, group in age_groups:
    purchase_data = group["Purchase"].values
    full_data_n = len(purchase_data)

    for size in sample_sizes:
        if size == "Full":
            current_data = purchase_data
            n = full_data_n
        else:
            # Take a random sample if sample size is less than full data size
            if size < full_data_n:
                current_data = np.random.choice(purchase_data, size=size,␣
↪replace=False)
                n = size
            else:
                current_data = purchase_data
                n = full_data_n


        if n > 1: # Cannot calculate CI with only one data point
            mean = np.mean(current_data)
            std_err = spy.sem(current_data) # Standard error of the mean

            # Calculate the t-score
            t_score = spy.t.ppf((1 + confidence_level) / 2, df=n-1)
            # Calculate the confidence interval
            margin_of_error = t_score * std_err
            confidence_interval = (mean - margin_of_error, mean +␣
↪margin_of_error)
            plot_data[age][size] = (mean, confidence_interval)
        else:
            plot_data[age][size] = (np.nan, (np.nan, np.nan)) # Handle groups␣
↪with one or zero data points


# Plotting the confidence intervals
fig, axs = plt.subplots(len(age_groups), len(sample_sizes), figsize=(20, 15),␣
↪sharey=True)
fig.suptitle(f"{int(confidence_level*100)}% Confidence Intervals by Age Group␣
↪and Sample Size (t-distribution)", fontsize=16)

age_order = sorted(list(age_groups.groups.keys())) # Maintain a consistent order
```

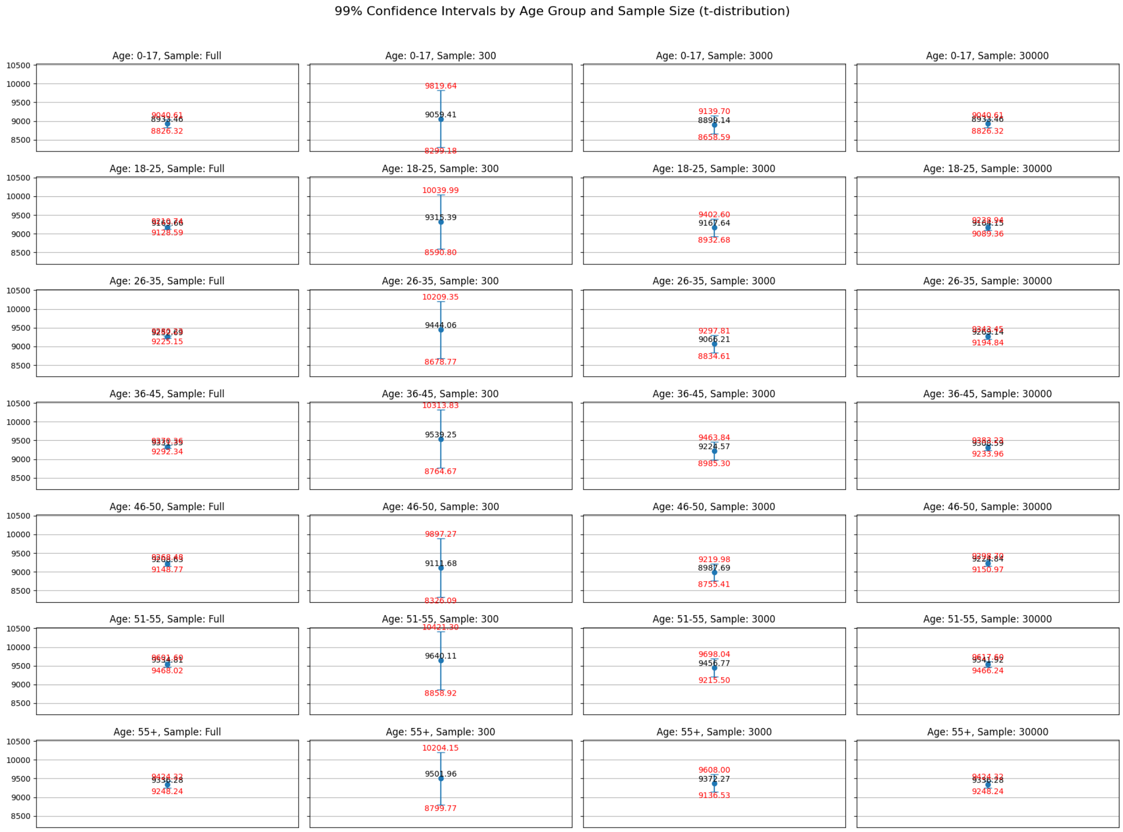
```python
for i, age in enumerate(age_order):
    for j, size in enumerate(sample_sizes):
        ax = axs[i, j]
        mean, ci = plot_data[age][size]
        if not np.isnan(mean):
            ax.errorbar(0, mean, yerr=(ci[1] - ci[0])/2, fmt='o', capsize=5)
            ax.set_title(f"Age: {age}, Sample: {size}")
            ax.set_xticks([]) # Remove x-axis ticks
            ax.grid(axis='y')
            ax.text(0, mean, f'{mean:.2f}', ha='center', va='bottom') # Add mean value as text
            ax.text(0, ci[0], f'{ci[0]:.2f}', ha='center', va='top', color='red') # Add lower bound as text
            ax.text(0, ci[1], f'{ci[1]:.2f}', ha='center', va='bottom', color='red') # Add upper bound as text


fig.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[61]:

```
# Calculate 99% confidence intervals using t-distribution for each age group␣
↪and sample size

# Define confidence levels and sample sizes
confidence_level = 0.99
sample_sizes = ["Full", 300, 3000, 30000]

# Group data by Age
age_groups = df.groupby("Age", observed=True)

results_t_dist = {}

# Store results for plotting
plot_data = {age: {size: None for size in sample_sizes} for age, _ in␣
↪age_groups}

for age, group in age_groups:
    purchase_data = group["Purchase"].values
    full_data_n = len(purchase_data)

    for size in sample_sizes:
        if size == "Full":
            current_data = purchase_data
            n = full_data_n
        else:
            # Take a random sample if sample size is less than full data size
            if size < full_data_n:
                current_data = np.random.choice(purchase_data, size=size,␣
↪replace=False)
                n = size
            else:
                current_data = purchase_data
                n = full_data_n


        if n > 1: # Cannot calculate CI with only one data point
            mean = np.mean(current_data)
            std_err = spy.sem(current_data) # Standard error of the mean

            # Calculate the t-score
            t_score = spy.t.ppf((1 + confidence_level) / 2, df=n-1)
            # Calculate the confidence interval
            margin_of_error = t_score * std_err
            confidence_interval = (mean - margin_of_error, mean +␣
↪margin_of_error)
            plot_data[age][size] = (mean, confidence_interval)
        else:
```

```
            plot_data[age][size] = (np.nan, (np.nan, np.nan)) # Handle groups␣
↪with one or zero data points


# Plotting the confidence intervals
fig, axs = plt.subplots(len(age_groups), len(sample_sizes), figsize=(20, 15),␣
↪sharey=True)
fig.suptitle(f"{int(confidence_level*100)}% Confidence Intervals by Age Group␣
↪and Sample Size (t-distribution)", fontsize=16)

age_order = sorted(list(age_groups.groups.keys())) # Maintain a consistent order

for i, age in enumerate(age_order):
    for j, size in enumerate(sample_sizes):
        ax = axs[i, j]
        mean, ci = plot_data[age][size]
        if not np.isnan(mean):
            ax.errorbar(0, mean, yerr=(ci[1] - ci[0])/2, fmt='o', capsize=5)
            ax.set_title(f"Age: {age}, Sample: {size}")
            ax.set_xticks([]) # Remove x-axis ticks
            ax.grid(axis='y')
            ax.text(0, mean, f'{mean:.2f}', ha='center', va='bottom') # Add␣
↪mean value as text
            ax.text(0, ci[0], f'{ci[0]:.2f}', ha='center', va='top',␣
↪color='red') # Add lower bound as text
            ax.text(0, ci[1], f'{ci[1]:.2f}', ha='center', va='bottom',␣
↪color='red') # Add upper bound as text

fig.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



Based on the confidence intervals calculated using the t-distribution for the average purchase amount across different age groups below are the insights:

Insights:

Varying Average Spending:

The confidence intervals show that the average purchase amount is different across various age groups.

Impact of Age:

Generally, older age groups (middle-aged) tend to have higher average purchase amounts compared to younger age groups (0-17).

Precision with Sample Size:

As the sample size within each age group increases, the confidence intervals become narrower, indicating a more precise estimate of the true average purchase amount for that group.

Confidence Level and Interval Width:

A higher confidence level (e.g., 99%) results in wider confidence intervals compared to lower confidence levels (e.g., 90%), reflecting a higher certainty that the true mean falls within the calculated range.

These insights suggest that age is a factor influencing spending habits, with middle-aged individuals generally spending more on average during Black Friday. The confidence intervals provide a statistically sound range for these average spending amounts within each age demographic.

Final Insights

0.0.1 Univariate Analysis Comments

1. **Gender**: The dataset has more male shoppers than female, which could influence purchasing trends.
2. **Age**: The 26–35 age group is the most represented, suggesting a youthful shopping population.
3. **City Category**: City B has the highest number of customers, followed by C and A. Urban-rural shopping patterns could be analyzed further.
4. **Stay in Current City**: Most shoppers have been in their current city for 1 or 2 years, hinting at a relatively mobile customer base.
5. **Purchase**: The distribution is right-skewed, with most purchases under 10,000. A few high-value purchases stretch the tail.
6. **Product Category**: Categories 1–20 dominate. Some products (like category 8 and 12) appear frequently, suggesting popular segments.

Next, I'll generate **bivariate plots** and then connect these observations to insights based on the **Central Limit Theorem (CLT)**.

0.0.2 Bivariate Analysis Comments

1. **Gender vs Purchase**: Males tend to have slightly higher median purchases, but the overall spread is similar for both genders.
2. **Age vs Purchase**: The 26–35 age group not only dominates in volume but also shows higher median and spread in spending.
3. **City Category vs Purchase**: City B exhibits slightly higher purchases. Cities A and C show a wider variance.
4. **Stay Duration vs Purchase**: Shoppers who have stayed for 1 or 2 years tend to spend slightly more. Newcomers and long-term residents spend less on average.
5. **Marital Status vs Purchase**: Single and married individuals show comparable medians, but married users show a slightly broader range in spending.
6. **Occupation vs Purchase**: Purchase amounts vary noticeably by occupation. Some job roles (e.g., IDs 4, 7, 10) display higher spending patterns.

0.0.3 Final Insights (with CLT Consideration)

1. **Distribution Insights**:

   - **Purchase amounts** are right-skewed with a long tail, indicating that the average might not represent most customers well.
   - **CLT in Action**: When sampling from this large dataset, the **sampling distribution of the mean Purchase** becomes approximately normal even though the raw data is skewed.

2. **Population-Level Generalization**:
    - Given the **large sample size (550,000+ records)**, we can reliably estimate population metrics (like mean purchase by gender or age group).
    - Confidence intervals around means (using CLT) would be **narrow** due to the large size, offering **high precision** in generalizations.
3. **Business Implications**:
    - Target marketing can focus on **males aged 26–35** in **City B**, who represent the largest and highest-spending segment.
    - Popular product categories (e.g., 1, 8, 12) could be prioritized in promotions or stock planning.
    - Spending patterns vary more by **age and occupation** than marital status or city tenure, guiding segmentation strategies.

Recommendations

Targeted marketing:

Since the majority of transactions are made by males, it would be beneficial to tailor marketing strategies to cater to their preferences and needs. This could include specific promotions, product offerings, or advertising campaigns designed to attract male customers.

Focus on popular occupations:

Given that 82.33% of transactions come from customers in 11 specific occupations, it would be wise to focus marketing efforts on these occupations. Understanding the needs and preferences of individuals in these occupations can help in creating targeted marketing campaigns and customized offers.

Engage with new residents:

As a significant portion of transactions (53.75%) come from customers who have recently moved to the current city, it presents an opportunity to engage with these new residents. Targeted marketing, welcoming offers, and incentives for newcomers can help capture their loyalty and increase their spending.

Emphasize popular product categories:

Since 82.43% of transactions are concentrated in just five product categories, allocating resources and promotions towards these categories can maximize sales potential. Highlighting these popular categories and offering attractive deals can encourage more purchases.

Increase focus on single customers:

Given that 59.05% of total revenue is generated by single customers, dedicating efforts to cater to their needs and preferences can help drive more sales. Understanding their motivations and targeting them with personalized offers can enhance their shopping experience and loyalty.

Optimize revenue from specific age groups:

Since a majority of transactions are made by customers between the ages of 26 and 45, it is important to focus marketing efforts on this demographic. Offering products and services that align with their interests and values can maximize revenue generation.

Location-based marketing:

With a significant number of customers belonging to specific cities, tailoring marketing strategies to target these locations can lead to better results. Allocating resources, promotions, and events based on the customer concentration in each city can help drive sales.

Emphasize top-selling product categories:

The top five product categories generate a substantial portion of total revenue. Investing in these categories, ensuring a wide range of options and competitive pricing, can capitalize on customer demand and drive overall sales.

Personalized offers for high spenders:

Identifying customers with high total spending, such as males or customers in specific age groups, allows for targeted marketing and personalized offers. Providing exclusive discounts, loyalty rewards, or special privileges to these customers can encourage repeat purchases and increase customer satisfaction.

Implement loyalty program:

Implementating a loyalty program that offers incentives, rewards, and exclusive deals to encourage repeat purchases and increase customer retention. Targeted loyalty programs can be designed for male customers, single customers,and customers in specific age groups.

Enhance product offerings:

Analyze the popular product categories and identify opportunities to expand the product range within those categories. This can attract more customers and increase sales. Additionally, identify complementary products or cross-selling opportunities to encourage customers to make additional purchases.