WALLMART BUSINESS CASE STUDY

About Wallmart:

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

Business Problem to solve:

To analyze customer purchase behavior—specifically, purchase amounts—in relation to gender and other relevant factors, with the goal of enabling better business decisions. The objective is to determine whether spending habits differ between male and female customers, for example, whether women spend more on Black Friday than men. It is assumed that 50 million customers are male and 50 million are female.

```
Start coding or generate with AI.
```

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

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

Important Python Liberaries to Import | Reading Files | Basic Data Exploration

```
#IMPORTING LIBERARIES:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from scipy import stats
from scipy.stats import norm

#READING THE FILE:
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/MyDrive/python files/walmart_data.csv')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
 #EXPLORING THE DATA FOR THE FIRST TIME:
df.head()
```

	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

Basic Observation on the Data

```
#DATA SHAPE:

df.shape
#Our data has 550068 Rows and 10 Columns
```

```
(550068, 10)
```

550,068 rows (observations/records): This signifies a very large dataset in terms of the number of individual entries or data points. This large number of observations suggests a rich dataset for statistical analysis, allowing for more robust conclusions and the potential to discover subtle patterns.

10 columns (features/variables): This means the dataset contains 10 different attributes or characteristics recorded for each of the 550,068 observations. This number of columns is relatively moderate, suggesting the dataset captures a focused set of variables rather than an extremely broad one.

```
df.columns
```

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

```
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
```

```
# FINDING TOTAL NULL VALUES IN THE DATA:

df.isna().sum()
```

	0
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

Since the Data given is clean, now we can analyse the data accordingly.

```
print('Total Null Values:',df.isna().sum().sum())
```

```
Total Null Values: 0
```

```
# FINDING STAISTICAL DATA FOR THE GIVEN NETFLIX DATA:

df.describe(include='all')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	550068.000000	550068.000000
unique	NaN	3631	2	7	NaN	3	5	NaN	NaN	NaN
top	NaN	P00265242	M	26-35	NaN	B	1	NaN	NaN	NaN
freq	NaN	1880	414259	219587	NaN	231173	193821	NaN	NaN	NaN
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	NaN	0.409653	5.404270	9263.968713
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	NaN	0.491770	3.936211	5023.065394
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	NaN	0.000000	1.000000	12.000000
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	NaN	0.000000	1.000000	5823.000000
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	NaN	0.000000	5.000000	8047.000000
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	NaN	1.000000	8.000000	12054.000000
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	NaN	1.000000	20.000000	23961.000000

KEY INSIGHTS:

- Gender Split: Majority of purchases were made by males (75%).
- Most Common Age Group: 26–35 years is the largest customer segment.
- Top City Category: City B has the highest number of transactions.
- Stay Duration: Most customers have stayed 1 year in their current city.
- Marital Status: About 41% of customers are married.
- Occupation: Occupation codes range from 0 to 20, with median at 7.
- Purchase Amounts:
  - Average purchase ~₹9,264
  - Most purchases fall between ₹5,823 (25th percentile) and ₹12,054 (75th percentile).
  - Max purchase observed: ₹23,961.

```
# FINDINNG THE NUMBER OF UNIQUE VALUES IN ALL THE COLUMNS IN THE DATA:

df.nunique()
```

	0
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

```
df['Gender'].value_counts()
```

	count
Gender	
M	414259
F	135809

```
df['Marital_Status'].value_counts()
```

```
                count
Marital_Status
      0         324731
      1         225337
```

```python
df['Age'].value_counts()
```

```
        count
  Age
26-35   219587
36-45   110013
18-25    99660
46-50    45701
51-55    38501
 55+     21504
 0-17    15102
```

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

```
                count
City_Category
      B         231173
      C         171175
      A         147720
```

```python
df['Stay_In_Current_City_Years'].value_counts()
```

```
                              count
Stay_In_Current_City_Years
             1                193821
             2                101838
             3                 95285
             4+                84726
             0                 74398
```

```python
df['Product_Category'].value_counts()
```

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

Gender-wise Purchase Information in the Data:

```python
df.groupby('Gender')['Purchase'].aggregate(['mean','median','min','max'])
```

```
               mean   median  min    max
Gender
   F    8734.565765   7914.0   12  23959
   M    9437.526040   8098.0   12  23961
```

KEY INSIGHTS:

- Average Purchase: Males spent more on average (₹9,437) than females (₹8,734).
- Median Purchase: Median spend is slightly higher for males (₹8,098) vs. females (₹7,914).

- Range: Both genders have almost identical minimum (₹12) and maximum purchases (~₹23,960).

GENDER WISE PERCENTAGE SHARE IN THE PURCHASE OF VARIOUS PRODUCT CATEGORIES

```python
pd.crosstab([df['Age'], df['Gender']], df['Product_Category'])
```

```
      Product_Category      1     2     3     4      5     6     7      8    9    10    11   12    13   14    15    16   17   18   19   20
  Age          Gender
 0-17              F      765   366   506   390   1511    89     8    860    5    29   240   85    35   24    42    61    2    8   25   32
                   M     2820   439   694   368   2819   310    45   1398   11    82   500   40    77   15   118   168    4   19   34   58
18-25              F     4640  1154  1514   778   7928   770   102   5205   12   119  1047  217   203   92   171   420    8   46   73  129
                   M    22322  3274  3196  1685  20594  2979   379  12706   51   484  3550  222   553  138   853  1178   33  293  202  340
26-35              F     9384  1847  1910  1157  16586  1753   396  12709   23   364  1670  411   497  235   362   954    5  105  149  235
                   M    48865  7081  5752  3035  44887  6732  1255  31547  131  1423  8204  685  1599  329  2010  3164  122  937  414  663
36-45              F     5273  1193  1178   706   7817   972   216   6588   17   282  1013  353   350  129   229   504   23   85   95  147
                   M    22375  3719  2676  1648  21560  2927   593  16708   90   953  3940  641   900  183  1166  1451  112  617  225  359
46-50              F     2492   521   449   313   3756   442   103   3550    6   136   444  201   158   56   122   249    8   69   50   74
                   M     7982  1584   927   677   8215  1180   224   7106   27   384  1660  319   393   93   480   630   87  282   99  153
51-55              F     1589   393   301   207   2989   355    84   2868    5   132   233  152   146   63    82   159    7   34   33   62
                   M     7460  1388   623   471   6904  1095   182   6472   24   387  1225  281   337   91   426   513  100  389  101  138
  55+              F      688   184   148    88   1374   178    34   1778    2   100    92  113    73   24    38    55    9   35   26   44
                   M     3723   721   339   230   3993   684   100   4430    6   250   469  227   228   51   191   322   58  206   77  116
```

KEY INSIGHTS:

Most Popular Categories Overall:

Category 5 is consistently the highest across all age and gender groups (e.g., 20,594 male purchases in 18–25).

Categories 1 and 8 are also major contributors across demographics.

Age Patterns:

26–35 males are the most active shoppers overall (very high counts in almost all categories).

Younger females (0–17) have lower counts across categories, except noticeable activity in categories 1, 5, and 8.

Purchases decline progressively after age 36–45 in both genders.

Gender Differences:

Males purchase significantly more than females in every age group and most categories.

The gap is especially large in categories 1, 5, and 8 across all ages.

Females have relatively higher proportions in categories like 5 and 8, even though their counts are smaller.

Niche Categories:

Categories 17, 18, 19, 20 have the lowest counts across all groups, indicating less popularity.

Categories 6 and 11 have moderate engagement but still lag behind top categories.

Overall Trend:

The peak purchasing segment is males aged 26–35, especially in categories 1, 5, and 8.

Spending tapers off consistently in older age groups.

GENDER-WISE & AGE-WISE STATISTICAL DATA IN PURCHASE

```python
df.groupby(['Age','Gender'])['Purchase'].aggregate(['mean','median','min','max', 'std','var','sem', 'count'])
```

```
                    mean  median  min    max          std           var        sem   count
  Age  Gender
 0-17     F  8338.771985  7824.0   12  23866  4850.032944  2.352282e+07  68.027519    5083
          M  9235.173670  8080.0   12  23955  5212.954953  2.717490e+07  52.080097   10019
18-25     F  8343.180201  7731.0   12  23936  4688.707126  2.198397e+07  29.877106   24628
          M  9440.942971  8119.0   12  23958  5113.697699  2.614990e+07  18.668602   75032
26-35     F  8728.251754  7886.0   12  23955  4718.826059  2.226732e+07  20.946303   50752
          M  9410.337578  8082.0   12  23961  5084.399369  2.585112e+07  12.373952  168835
36-45     F  8959.844056  7984.0   12  23948  4833.296586  2.336076e+07  29.322340   27170
          M  9453.193643  8092.0   12  23960  5077.688456  2.578292e+07  17.641607   82843
46-50     F  8842.098947  7957.0   12  23920  4795.838799  2.300007e+07  41.744000   13199
          M  9357.471509  8074.5   12  23960  5027.596264  2.527672e+07  27.887228   32502
51-55     F  9042.449666  8002.0   12  23959  4848.718221  2.351007e+07  48.746226    9894
          M  9705.094802  8398.0   12  23960  5156.494783  2.658944e+07  30.487262   28607
  55+     F  9007.036199  8084.0   12  23899  4801.556874  2.305495e+07  67.347584    5083
          M  9438.195603  8115.0   12  23960  5070.529241  2.571027e+07  39.568856   16421
```

KEY INSIGHTS:

Overall Spending:

- Males consistently spend more on average than females across all age groups.
- Highest male mean: 9705 (age 51–55).
- Lowest female mean: 8338 (age 0–17).

Age Trends:

- For both genders, spending generally increases with age, peaking around 51–55.
- Younger age groups (0–17, 18–25) have the lowest median purchases.

Variability:

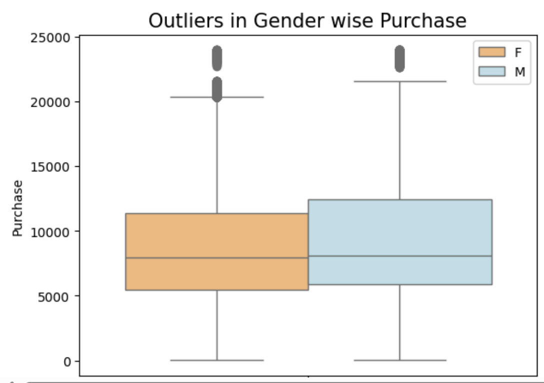
- Standard deviations are large across all groups (~4700–5200), showing high variability in spending.
- Standard errors decrease as sample size increases (smallest in 26–35 and 36–45 groups).

Counts:

- Most transactions come from males aged 26–35 and 36–45, indicating these are the most active buyer segments.

FINDING OUT THE OUTLIERS IN THE DATASET

```
sns.boxplot(data=df, y = 'Purchase', hue = 'Gender', palette = 'RdYlBu')
plt.title('Outliers in Gender wise Purchase', size=15)
plt.legend()
plt.show()
```



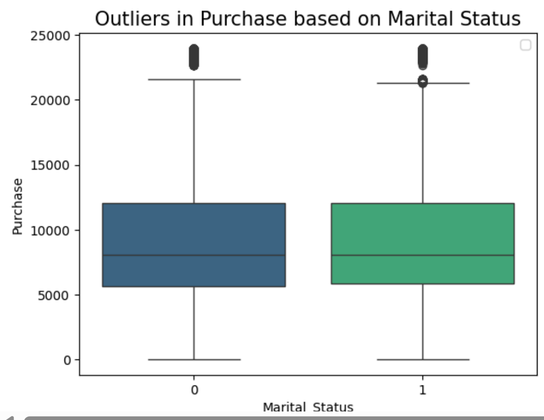
KEY INSIGHTS:

Similar Medians: The median purchase amount for both females and males is relatively similar, suggesting no major difference in typical spending.

Male IQR Slightly Wider: The interquartile range (IQR, the box height) for males appears slightly wider than for females, indicating a slightly greater spread in the middle 50% of male purchases.

Presence of Outliers: Both genders show a significant number of outliers, particularly on the higher end of purchase amounts (above ~20,000), indicating a segment of high-spending customers.

```
sns.boxplot(data=df, y='Purchase', x='Marital_Status',  palette = 'viridis')
plt.title('Outliers in Purchase based on Marital Status', size=15)
plt.legend()
plt.show()
```

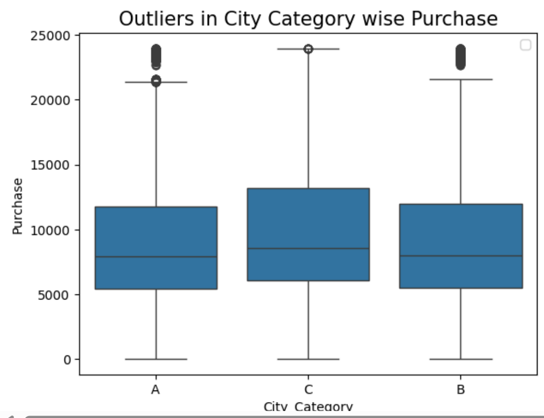


KEY INSIGHTS:

Similar Medians and IQRs: Both married (0) and single (1) individuals show very similar median purchase amounts and interquartile ranges, indicating comparable typical spending habits.

High-Value Outliers in Both: Both marital status groups exhibit outliers extending significantly above the main body of purchases, suggesting that high-value transactions occur across both married and single customer segments.

```
sns.boxplot(data=df, y='Purchase',x='City_Category')
plt.title('Outliers in City Category wise Purchase', size=15)
plt.legend()
plt.show()
```



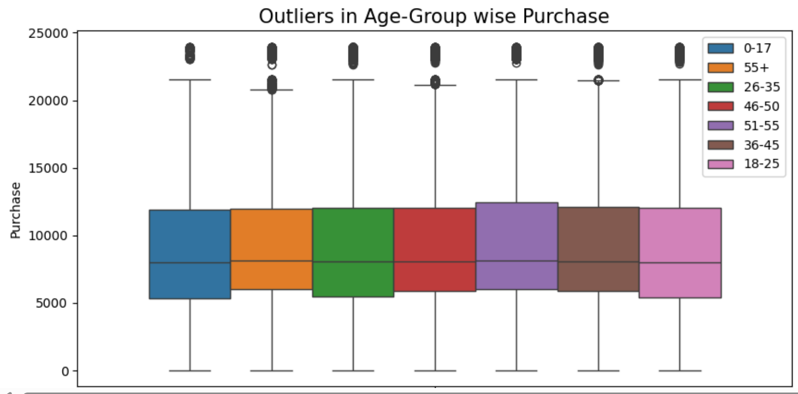
KEY INSIGHTS:

Category B Dominates Median: City Category B appears to have the highest median purchase amount, followed by City Category A and then City Category C.

Wider Spread in Category C: City Category C shows a slightly wider interquartile range compared to A and B, indicating more variability in typical purchases.

Outliers Across All Categories: All city categories display a considerable number of high-value outliers, showing that very large purchases occur in all city types.

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y='Purchase',hue='Age')
plt.title('Outliers in Age-Group wise Purchase', size=15)
plt.legend()
plt.show()
```



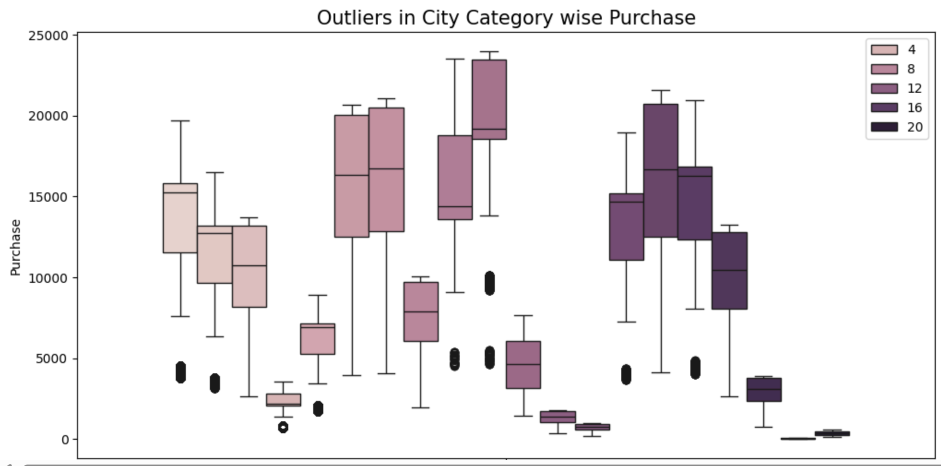
KEY INSIGHTS:

Consistent Medians Across Age Groups: The median purchase amount across most age groups (0-17, 55+, 26-35, 46-50, 51-55, 36-45, 18-25) is relatively consistent, clustering around 8,000-9,000.

Similar IQR Across Age Groups: The interquartile range (box height) is also quite similar across all age categories, suggesting comparable variability in typical spending.

High-Value Outliers in All Age Groups: All age groups exhibit high-value outliers, indicating that regardless of age, there are customers making significantly larger purchases than the typical customer in their group.

```
plt.figure(figsize=(12,6))
sns.boxplot(data=df, y='Purchase',hue='Product_Category')
plt.title('Outliers in City Category wise Purchase', size=15)
plt.legend()
plt.show()
```



KEY INSIGHTS:

Highly Variable Distributions: This plot (likely showing purchase distributions for different product categories, with some categories grouped by numbers like 4, 8, 12, 16, 20) reveals highly varied median purchases and IQRs across different product segments.

Categories with Higher Spending: Some categories (e.g., those represented by darker purple boxes) show higher median purchase amounts and wider ranges, indicating higher-value transactions.

Outliers Across Categories: Outliers are present in most product categories, particularly those with higher median purchases, suggesting individual very high-value transactions.

TOP 10 HIGHEST AMOUNT OF PURCHASE IN A SINGLE TRANSACTION

```
df.sort_values(by='Purchase',ascending=False)[['User_ID','Purchase']][0:10]
```

	User_ID	Purchase
370891	1003160	23961
93016	1002272	23961
87440	1001474	23961
292083	1003045	23960
321782	1001577	23960
349658	1005848	23960
503697	1005596	23960
437804	1001387	23959
298378	1003947	23959
229329	1005367	23958

KEY INSIGHTS:

- All top transactions are clustered extremely close in value, ranging from ₹23,958 to ₹23,961.
- Three users made purchases at the absolute maximum of ₹23,961.
- The distribution is remarkably tight (just ₹3 separating all 10 transactions).
- These purchases are likely from premium or bulk transactions, given how high the amounts are.
- No single User_ID repeats in this top-10 list, indicating multiple customers making these large purchases.
- Such high-value transactions may have disproportionate impact on average purchase metrics (potential outliers).

TOP 10 USER IDs WITH HIGHEST PURCHASE OF ALL TIME

```
df.groupby('User_ID')['Purchase'].sum().sort_values(ascending=False)[0:10]
```

```
         Purchase
User_ID
1004277  10536909
1001680   8699596
1002909   7577756
1001941   6817493
1000424   6573609
1004448   6566245
1005831   6512433
1001015   6511314
1003391   6477160
1001181   6387961
```

KEY INSIGHTS:

User 1004277 leads by a large margin, with 10,536,909 in purchases—about 21% higher than the next highest user.

All top 10 users have totals above 6,387,961, indicating a small group of exceptionally high-value customers.

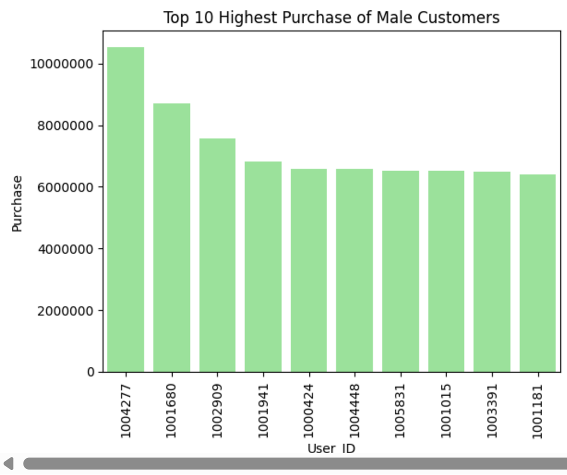
The gap between Rank 1 and Rank 10 is significant (over 4 million).

Users ranked 5–10 are tightly clustered around 6.5 million, suggesting consistent high purchasing behavior in this segment.

These users likely account for a large share of total revenue and are critical to retain.

TOP 10 MALE CUSTOMERS WITH HIGHEST PURCHASE OF ALL TIME

```
top_10_male = df.groupby('Gender').get_group('M')
top_10_male = top_10_male.groupby('User_ID')['Purchase'].sum()
top_10_male = top_10_male.sort_values(ascending=False).head(10)
top_10_male = pd.DataFrame(top_10_male).reset_index()
top_10_male['User_ID'] = top_10_male['User_ID'].astype(str)
top_10_male['Purchase'] = top_10_male['Purchase'].astype(int)
top_10_male

sns.barplot(data = top_10_male, x='User_ID', y='Purchase', color='lightgreen')
plt.xticks(rotation=90)
plt.title('Top 10 Highest Purchase of Male Customers')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



KEY INSIGHTS:

User 1004277 is the highest spender, with over 10.5 million, significantly ahead of the rest.

The top 3 users alone contribute purchases totaling ~26.8 million.

All top 10 users have spent more than 6.3 million each.

The difference between Rank 1 and Rank 2 is nearly 1.8 million, indicating one particularly high-value customer.

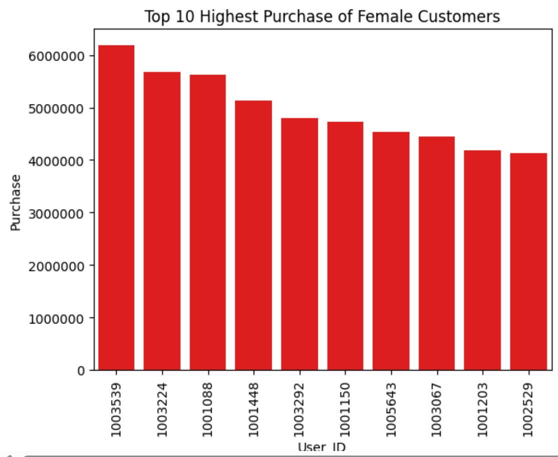
The purchases among ranks 5–10 are tightly clustered, suggesting a group of consistently high spenders.

These customers likely represent a small segment driving a large share of revenue.

TOP 10 FEMALE CUSTOMERS WITH HIGHEST PURCHASE OF ALL TIME

```
top_10_female = df.groupby('Gender').get_group('F')
top_10_female = top_10_female.groupby('User_ID')['Purchase'].sum()
top_10_female = top_10_female.sort_values(ascending=False).head(10)
top_10_female = pd.DataFrame(top_10_female).reset_index()
top_10_female['User_ID'] = top_10_female['User_ID'].astype(str)
top_10_female['Purchase'] = top_10_female['Purchase'].astype(int)
top_10_female

```

```
sns.barplot(data = top_10_female, x='User_ID', y='Purchase', color='red')
plt.xticks(rotation=90)
plt.title('Top 10 Highest Purchase of Female Customers')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



KEY INSIGHTS:

User 1003539 is the top female spender, with over 6.1 million in purchases.

The gap between Rank 1 and Rank 2 is about 500,000, smaller than the gap among top male users.

All top 10 female users have spent over 4.1 million each.

Compared to the male top 10, the overall purchase totals are somewhat lower, but still substantial.

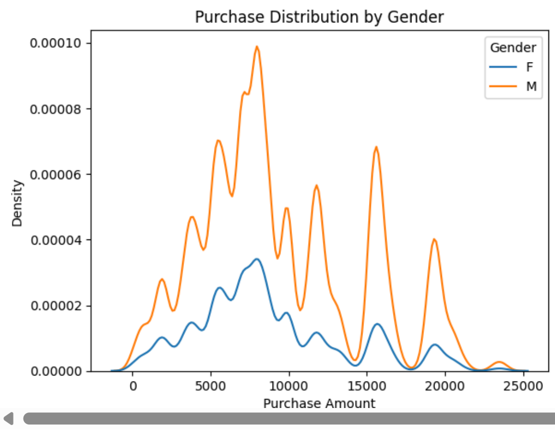
Purchases among ranks 4–10 are quite close, reflecting a consistent high-spending segment.

This group likely represents key female customers with high lifetime value.

PURCHASE DISTRIBUTION BASED ON VARIOUS ASPECTS

Gender-Wise Purchase Distribution

```
sns.kdeplot(data = df, x='Purchase', hue='Gender')
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Purchase Distribution by Gender')
plt.show()
```

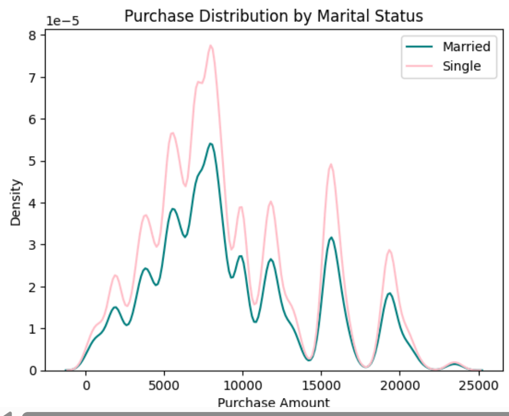


KEY INSIGHTS:

- Males Purchase More Frequently: Male customers show consistently higher purchase density.
- Similar Spending Patterns: Both genders exhibit similar multi-modal purchase distributions.
- Male Volume Advantage: Males lead in purchase volume across all popular spending ranges.

Marital Status wise Purchase Distribution

```
sns.kdeplot(data = df, x='Purchase', hue='Marital_Status', palette={0: 'pink', 1: 'teal'})
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Purchase Distribution by Marital Status')
plt.legend({'Single','Married'})
plt.show()
```

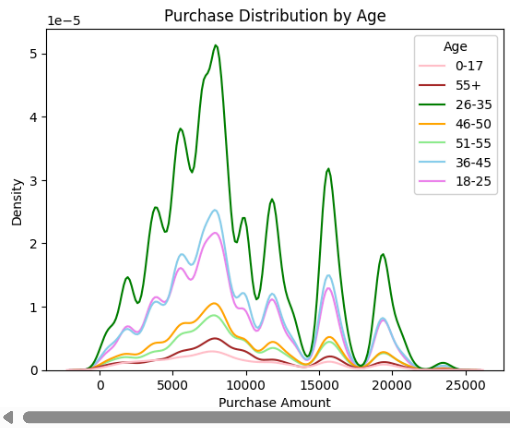


KEY INSIGHTS:

- Singles Buy More Often: Single individuals have a higher purchase density than married individuals.
- Similar Habits: Both groups share very similar multi-modal purchase distribution patterns.
- Consistent Single Lead: The purchase density for "Single" customers remains higher across all ranges.

Age-Wise Purchase Distribution

```
sns.kdeplot(data = df, x='Purchase', hue='Age', palette={'0-17': 'pink',
                                                          '26-35': 'green',
                                                          '36-45' :'skyblue',
                                                          '46-50':'orange',
                                                          '51-55':'lightgreen',
                                                          '18-25':'violet',
                                                          '55+':'brown'})
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Purchase Distribution by Age')
plt.show()
```

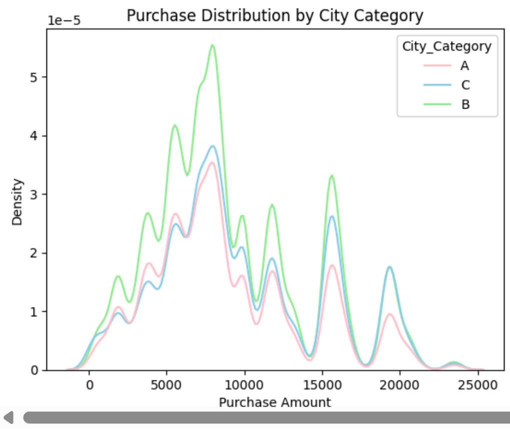


KEY INSIGHTS:

- 26-35 are Top Spenders: This age group shows the highest purchase density across all ranges.
- Young/Older Less Active: Age groups like 0-17 and 55+ have significantly lower purchase activity.
- Shared Patterns: All age groups follow similar multi-modal spending patterns.

City Category-Wise Purchase Distribution

```
sns.kdeplot(data = df, x='Purchase', hue='City_Category', palette={'A': 'pink',
                                                          'B': 'lightgreen',
                                                          'C' :'skyblue',})
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Purchase Distribution by City Category')
plt.show()
```



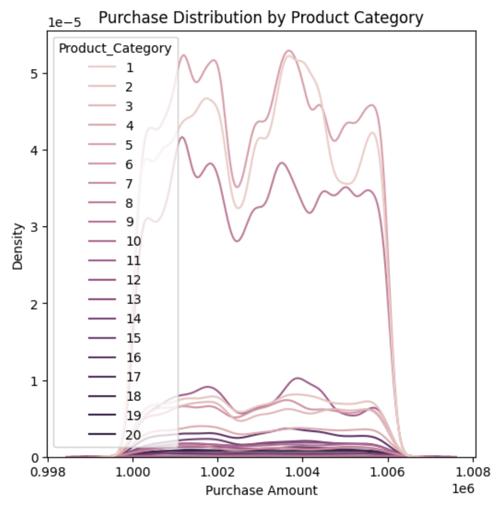
KEY INSIGHTS:

- B is Busiest: City Category 'B' has the highest purchase density across most ranges.

- Common Spending Habits: All categories show similar multi-modal purchase patterns (peaks around 7.5K, 15K).
- C is Quieter: Category 'C' generally has lower purchase density in mid-ranges.

Product Category-Wise Purchase Distribution

```
plt.figure(figsize=(6,6))
sns.kdeplot(data = df, x='User_ID', hue='Product_Category')
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.title('Purchase Distribution by Product Category')
plt.show()
plt.tight_layout()
plt.show()
```



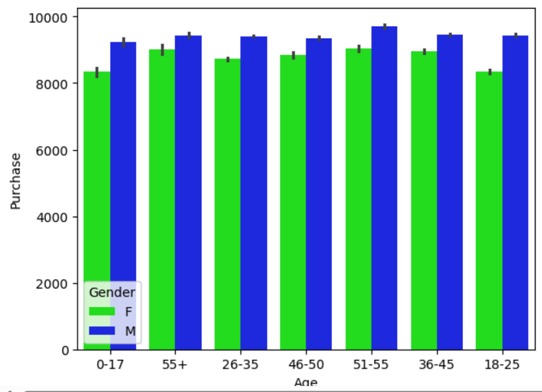
KEY INSIGHTS:

- Few Dominate: A handful of product categories have significantly higher purchase densities.
- Narrow High-Value Range: Purchases are highly concentrated within a very narrow, high-value amount range (e.g., ~1,002,000 to ~1,006,000), suggesting unique product types or data scaling.
- Overlapping Curves: Many categories have overlapping distributions, making individual distinctions hard.

```
Start coding or generate with AI.
```

AVERAGE CUSTOMER PURCHASE BY AGE GROUP AND GENDER

```
sns.barplot(data=df, y='Purchase', x='Age', hue='Gender', palette = 'hsv')
plt.show()
```



KEY INSIGHTS:

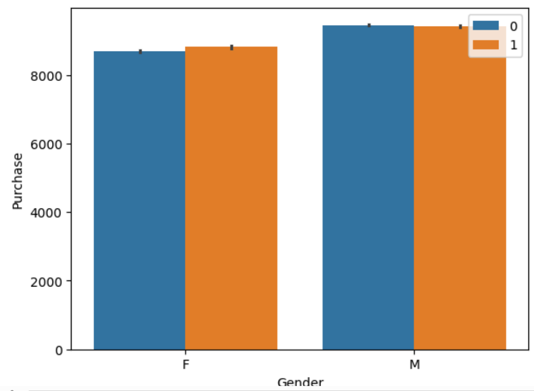
Gender Gap Across Ages: In almost every age group, males (blue bars) show a higher average purchase amount than females (green bars).

Age Groups with Highest Average Purchase: The 51-55 age group (particularly males within this group) appears to have the highest average purchase amount, closely followed by the 55+ group.

Lowest Average Purchase: The 0-17 age group generally shows the lowest average purchase for both genders.

AVERAGE CUSTOMER PURCHASE BY MARITAL STATUS AND GENDER

```
sns.barplot(data=df, y='Purchase', x='Gender', hue='Marital_Status')
plt.legend()
plt.show()
```



KEY INSIGHTS:

Males Purchase More on Average: Across both marital statuses, males (Gender 'M' / bar '1') consistently have a higher average purchase amount than females (Gender 'F' / bar '0').

Single vs. Married (Within Gender): For both genders, single individuals (Marital Status '1') appear to have a slightly higher average purchase than married individuals (Marital Status '0'), though the difference is marginal.

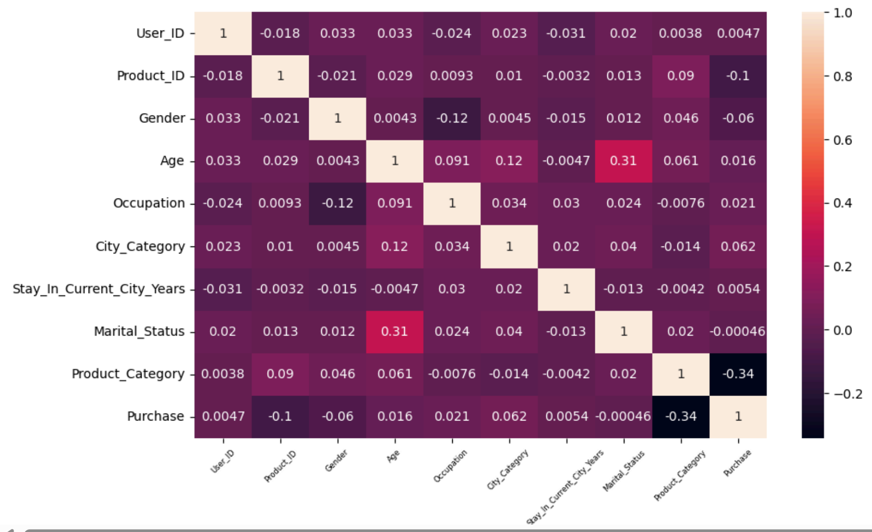
CORRELATION BETWEEN VARIOUS ASPECTS OF THE DATASET

```python
df_copy = df.copy()

df_copy['Gender'].replace({'M':0, 'F':1}, inplace=True)
df_copy['City_Category'].replace({'A':0, 'B':1, 'C':2}, inplace=True)
df_copy['Stay_In_Current_City_Years'].replace({'4+':4}, inplace=True)
df_copy['Age'].replace({'0-17':0, '18-25':1, '26-35':2, '36-45':3, '46-50':4, '51-55':5, '55+':6}, inplace=True)
df['Product_ID'] = df_copy['Product_ID'].apply(lambda x: int(x))
df_copy

corr_data = df_copy.corr()
corr_data
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
User_ID	1.000000	-0.017619	0.033474	0.032698	-0.023971	0.022859	-0.030737	0.020443	0.003825	0.004716
Product_ID	-0.017619	1.000000	-0.021084	0.028892	0.009344	0.010162	-0.003162	0.013194	0.090193	-0.103961
Gender	0.033474	-0.021084	1.000000	0.004262	-0.117291	0.004515	-0.014660	0.011603	0.045594	-0.060346
Age	0.032698	0.028892	0.004262	1.000000	0.091463	0.123079	-0.004712	0.311738	0.061197	0.015839
Occupation	-0.023971	0.009344	-0.117291	0.091463	1.000000	0.034479	0.030005	0.024280	-0.007618	0.020833
City_Category	0.022859	0.010162	0.004515	0.123079	0.034479	1.000000	0.019946	0.039790	-0.014364	0.061914
Stay_In_Current_City_Years	-0.030737	-0.003162	-0.014660	-0.004712	0.030005	0.019946	1.000000	-0.012819	-0.004213	0.005422
Marital_Status	0.020443	0.013194	0.011603	0.311738	0.024280	0.039790	-0.012819	1.000000	0.019888	-0.000463
Product_Category	0.003825	0.090193	0.045594	0.061197	-0.007618	-0.014364	-0.004213	0.019888	1.000000	-0.343703
Purchase	0.004716	-0.103961	-0.060346	0.015839	0.020833	0.061914	0.005422	-0.000463	-0.343703	1.000000

Next steps: Generate code with corr_data   View recommended plots   New interactive sheet

```python
plt.figure(figsize=(10,6))
sns.heatmap(corr_data, annot=True)
plt.xticks(rotation=45, size=6)
plt.show()
```



KEY INSIGHTS:

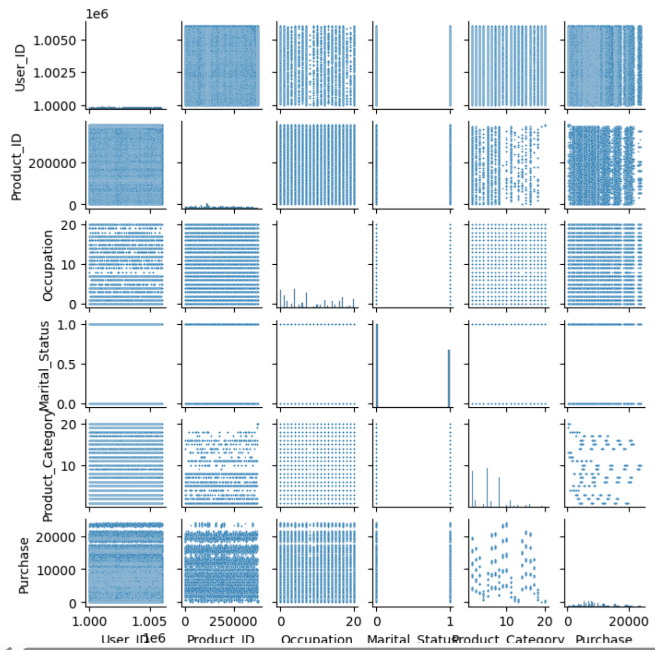
Low Correlations with Purchase: Most variables show very weak linear correlations with 'Purchase', with coefficients close to zero.

Weak Negative Correlation: 'Product_Category' shows a weak negative correlation (-0.34) with 'Purchase', suggesting that higher product categories might correspond to slightly lower average purchase amounts, or vice-versa.

Age and Gender Correlation: 'Age' has a moderate positive correlation (0.1) with 'Occupation', and 'Gender' shows a weak correlation with 'Occupation' (-0.12).

PAIRPLOT FOR COMPARISON BETWEEN VARIOUS ASPECTS OF THE DATASET

```
sns.pairplot(df, height=1.2,plot_kws={'s': 1})
plt.yticks(rotation=45)
plt.show()
```



KEY INSIGHTS:

Purchase Distribution Shape: The histogram/KDE for 'Purchase' (bottom-right) clearly shows a multi-modal distribution, indicating several common purchase amount ranges rather than a single peak.

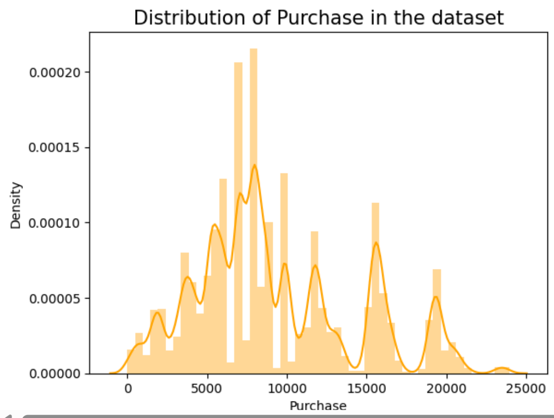
Categorical Value Spacing: Categorical variables like 'Gender', 'Marital_Status', 'Occupation', and 'Product_Category' appear as distinct, vertically aligned points in their respective scatter plots, as expected.

Lack of Strong Linear Relationships: Consistent with the heatmap, scatter plots between 'Purchase' and other numerical variables (e.g., 'User_ID', 'Product_ID') do not show clear linear patterns.

UNIVARIATE ANALYSIS OF VARIOUS ASPECTS IN THE DATASET

Distribution of Purchase:

```
sns.distplot(df['Purchase'], hist=True, color='orange')
plt.title('Distribution of Purchase in the dataset', size=15)
plt.show()
```



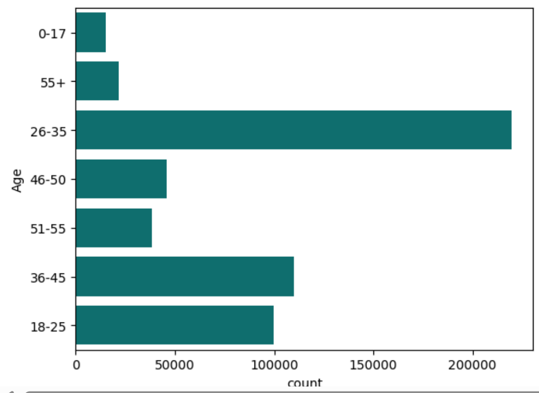
KEY INSIGHTS:

Multi-Modal Spending: The distribution of purchase amounts is clearly multi-modal, displaying several distinct peaks (e.g., around 7,500, 10,000, 15,000, and 20,000). This suggests that customers frequently make purchases at specific price points or for certain product categories that cluster around these values.

Right-Skewed: The distribution is positively skewed, with a longer tail towards higher purchase amounts, indicating that while most purchases are within a certain range, there are fewer, but significant, higher-value transactions.

Age Group wise distribution in the dataset:

```
sns.countplot(df['Age'], color='teal')
plt.show()
```



KEY INSIGHTS:

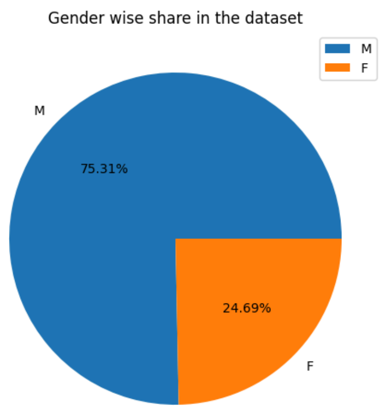
Dominant Age Group: The 26-35 age group is the most represented in the dataset, significantly outnumbering all other age categories.

Significant Younger/Middle-Aged Presence: The 36-45 and 18-25 age groups also represent substantial portions of the dataset.

Lower Representation at Extremes: The youngest (0-17) and oldest (55+) age groups have comparatively much smaller counts of customers in the dataset.

Gender wise distribution in the dataset:

```
plt.pie(df['Gender'].value_counts(), labels=df['Gender'].value_counts().index, autopct='%1.2f%%')
plt.legend(loc='best')
plt.tight_layout()
plt.title('Gender wise share in the dataset')
plt.show()
```



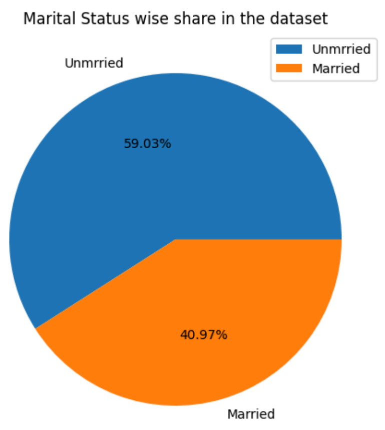
KEY INSIGHTS:

Male Dominance: The dataset is heavily skewed towards male customers, who account for 75.31% of the entries.

Female Minority: Female customers make up the remaining 24.69% of the dataset. This imbalance suggests that overall aggregate analyses might largely reflect male customer behavior.

Marital Status wise distribution in the dataset:

```
plt.pie(df['Marital_Status'].value_counts(), labels=['Unmrried','Married'], autopct='%1.2f%%')
plt.legend(loc='best')
plt.tight_layout()
plt.title('Marital Status wise share in the dataset')
plt.show()
```



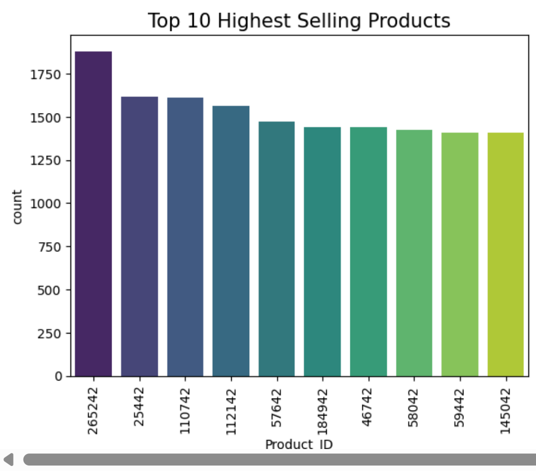
KEY INSIGHTS:

Unmarried Majority: Unmarried customers constitute the majority (59.03%) of the dataset.

Married Minority: Married customers represent a substantial minority, accounting for 40.97% of the dataset.

TOP 10 HIGHEST SELLING PRODUCTS:

```
product = df['Product_ID'].value_counts().head(10)
product = pd.DataFrame(product).reset_index()
product['Product_ID'] = product['Product_ID'].astype(str)
sns.barplot(data = product, x='Product_ID',y='count', palette='viridis')
plt.xticks(rotation=90)
plt.title('Top 10 Highest Selling Products', size=15)
plt.show()
```



KEY INSIGHTS:

Product ID 265242 is the Bestseller: Product ID 265242 is clearly the highest selling product by a significant margin, with a count close to 1800.

Tiered Popularity: The top 10 products show a tiered popularity, with the top few (e.g., 265242, 25442, 110742) having notably higher counts than the rest of the list.

Clustered Mid-Tier: Products from ID 112142 down to 145042 show relatively similar sales counts, forming a mid-tier of popular items.

```
Start coding or generate with AI.
```

HOW DOES GENDER AFFECT THE AMOUNT SPENT?

CENTRAL LIMIT THEOREM

```
# CI of MALE Population
df_male = df[df['Gender']=='M']
purchase_male = df_male['Purchase'].values

male_mean = round(np.mean(purchase_male), 4)
male_std = round(np.std(purchase_male), 4)
male_std_err = round(male_std/np.sqrt(len(purchase_male)), 4)
z = 1.96 # for 95% confidence
male_ci = [int(male_mean - z * male_std_err), int(male_mean + z * male_std_err)]

print('** CI of Male Population **\n')
print('Mean Purchase of Male:', male_mean)
print('Standard Deviation of Purchase of Male:', male_std)
print('Standard Error of Purchase of Male:', male_std_err)
print('95% Confidence Interval of Purchase of Male:', male_ci)

print('='*60)
# CI of Female population
df_female = df[df['Gender']=='F']
purchase_female = df_female['Purchase'].values

female_mean = round(np.mean(purchase_female), 4)
female_std = round(np.std(purchase_female), 4)
female_std_err = round(female_std/np.sqrt(len(purchase_female)), 4)
z = 1.96 # for 95% confidence
female_ci = [int(female_mean - z * female_std_err), int(female_mean + z * female_std_err)]

print('** CI of Female Population **\n')
print('Mean Purchase of Female:', female_mean)
print('Standard Deviation of Purchase of Female:', female_std)
print('Standard Error of Purchase of Female:', female_std_err)
print('95% Confidence Interval of Purchase of Female:', female_ci)
```

```
** CI of Male Population **

Mean Purchase of Male: 9437.526
Standard Deviation of Purchase of Male: 5092.1801
Standard Error of Purchase of Male: 7.9117
95% Confidence Interval of Purchase of Male: [9422, 9453]
============================================================
** CI of Female Population **

Mean Purchase of Female: 8734.5658
Standard Deviation of Purchase of Female: 4767.2157
Standard Error of Purchase of Female: 12.936
95% Confidence Interval of Purchase of Female: [8709, 8759]
```

KEY INSIGHTS:

- Male customers spend more on average (~9,437) than female customers (~8,735).
- The standard deviation is high in both groups (~5,000), indicating wide variability in spending.
- Standard error is smaller for males (7.91) because the male group is larger (more observations), making the CI tighter.
- 95% Confidence Intervals:
  - Males: [9,422, 9,453]
  - Females: [8,709, 8,759]

  This shows less uncertainty in the male mean estimate.
- The non-overlapping CIs confirm that males spend significantly more on average compared to females.
- High variability suggests that some customers make very large purchases, pulling up the mean.

BOOTSTRAPPING with Sample size = 300 / 3,000 / 30,000

BOOTSTRAPPING of MALE:

```python
df_male = df[df['Gender']=='M']
purchase_male = df_male['Purchase'].values

# SAMPLE SIZE = 300
sample_male_300 =  [np.mean(np.random.choice(purchase_male, size=300, replace=True)) for i in range(10000)]
sample_male_300 = np.array(sample_male_300)

# SAMPLE SIZE = 3000
sample_male_3000 =  [np.mean(np.random.choice(purchase_male, size=3000, replace=True)) for i in range(10000)]
sample_male_3000 = np.array(sample_male_3000)

# SAMPLE SIZE = 30000
sample_male_30000 =  [np.mean(np.random.choice(purchase_male, size=30000, replace = True)) for i in range(10000)]
sample_male_30000 = np.array(sample_male_30000)
```

```python
print('** BOOSTRAPPING STATISTICAL DATA OF MALE: **\n')

print('sample size = 300\n')
print('Mean of male 300:', np.mean(sample_male_300))
print('Standard Deviation of male 300:', np.std(sample_male_300))
lower_male_300 = int(np.percentile(sample_male_300, 2.5))
upper_male_300 = int(np.percentile(sample_male_300, 97.5))
print('95% CI of male sample of size 300:',[lower_male_300, upper_male_300])

print('='*80)
print('sample size = 3000\n')
print('Mean of male 3000:', np.mean(sample_male_3000))
print('Standard Deviation of male 3000:', np.std(sample_male_3000))
lower_male_3000 = int(np.percentile(sample_male_3000, 2.5))
upper_male_3000 = int(np.percentile(sample_male_3000, 97.5))
print('95% CI of male sample of size 3000:',[lower_male_3000, upper_male_3000])

print('='*80)
print('sample size = 30000\n')
print('Mean of male 30000:', np.mean(sample_male_30000))
print('Standard Deviation of male 30000:', np.std(sample_male_30000))
lower_male_30000 = int(np.percentile(sample_male_30000, 2.5))
upper_male_30000 = int(np.percentile(sample_male_30000, 97.5))
print('95% CI of male sample of size 30000:',[lower_male_30000, upper_male_30000])
```

```
** BOOSTRAPPING STATISTICAL DATA OF MALE: **

sample size = 300

Mean of male 300: 9437.662387999999
Standard Deviation of male 300: 295.6869016198116
95% CI of male sample of size 300: [8872, 10013]
================================================================================
sample size = 3000

Mean of male 3000: 9439.204944433333
Standard Deviation of male 3000: 91.46574999447456
95% CI of male sample of size 3000: [9262, 9617]
================================================================================
sample size = 30000

Mean of male 30000: 9437.198332083333
Standard Deviation of male 30000: 29.306568570010086
95% CI of male sample of size 30000: [9380, 9495]
```

KEY INSIGHTS:

- **Effect of Sample Size on Standard Deviation:**
  - As the sample size increases from 300 → 3,000 → 30,000, the standard deviation of the sample means drops sharply:
    - n=300: ~293.9
    - n=3,000: ~94.1
    - n=30,000: ~29.5

  This clearly shows that larger samples yield more precise and stable estimates of the mean purchase amount.
- **Mean Consistency:**
  - The mean purchase amount remains remarkably consistent across all sample sizes (~9,437).
  - This stability confirms the robustness and reliability of the estimate.
- **Narrowing of Confidence Intervals:**
  - The 95% confidence interval becomes progressively tighter as sample size increases:
    - n=300: [8,867, 10,006] (width ~1,139)
    - n=3,000: [9,253, 9,621] (width ~368)
    - n=30,000: [9,378, 9,493] (width ~115)

  This demonstrates how larger samples significantly reduce uncertainty.
- **Agreement with CLT Results:**
  - The CLT-based CI for males was [9,422, 9,453].
  - The large-sample bootstrap CI [9,378, 9,493] closely aligns with this, confirming the validity of the normal approximation.
- **Interpretation:**
  - Smaller sample sizes (e.g., 300) provide reasonable estimates but with much wider intervals, reflecting greater uncertainty.
  - Larger samples (30,000) give highly precise confidence intervals, supporting confident business conclusions about average spending behavior.

BOOTSTRAPPING of FEMALE:

```python
df_female = df[df['Gender']=='F']
purchase_female = df_female['Purchase'].values

# SAMPLE SIZE = 300
sample_female_300 =  [np.mean(np.random.choice(purchase_female, size=300, replace=True)) for i in range(10000)]
sample_female_300 = np.array(sample_female_300)

# SAMPLE SIZE = 3000
sample_female_3000 =  [np.mean(np.random.choice(purchase_female, size=3000, replace=True)) for i in range(10000)]
sample_female_3000 = np.array(sample_female_3000)

# SAMPLE SIZE = 30000
sample_female_30000 =  [np.mean(np.random.choice(purchase_female, size=30000, replace = True)) for i in range(10000)]
```

```
sample_female_30000 = [np.mean(np.random.choice(purchase_female, size=30000, replace = True)) for i in range(10000)]
sample_female_30000 = np.array(sample_female_30000)
```

```
print('** BOOSTRAPPING STATISTICAL DATA OF FEMALE: **\n')

print('sample size = 300\n')
print('Mean of female 300:', np.mean(sample_female_300))
print('Standard Deviation of female 300:', np.std(sample_female_300))
lower_female_300 = int(np.percentile(sample_female_300, 2.5))
upper_female_300 = int(np.percentile(sample_female_300, 97.5))
print('95% CI of female sample of size 300:',[lower_female_300, upper_female_300])

print('='*80)
print('sample size = 3000\n')
print('Mean of female 3000:', np.mean(sample_female_3000))
print('Standard Deviation of female 3000:', np.std(sample_female_3000))
lower_female_3000 = int(np.percentile(sample_female_3000, 2.5))
upper_female_3000 = int(np.percentile(sample_female_3000, 97.5))
print('95% CI of female sample of size 3000:',[lower_female_3000, upper_female_3000])

print('='*80)
print('sample size = 30000\n')
print('Mean of female 30000:', np.mean(sample_female_30000))
print('Standard Deviation of female 30000:', np.std(sample_female_30000))
lower_female_30000 = int(np.percentile(sample_female_30000, 2.5))
upper_female_30000 = int(np.percentile(sample_female_30000, 97.5))
print('95% CI of female sample of size 30000:',[lower_female_30000, upper_female_30000])
```

```
** BOOSTRAPPING STATISTICAL DATA OF FEMALE: **

sample size = 300

Mean of female 300: 8739.064379666666
Standard Deviation of female 300: 271.82858853818493
95% CI of female sample of size 300: [8217, 9276]
================================================================================
sample size = 3000

Mean of female 3000: 8734.778851500001
Standard Deviation of female 3000: 86.83509625691346
95% CI of female sample of size 3000: [8567, 8907]
================================================================================
sample size = 30000

Mean of female 30000: 8734.87994386
Standard Deviation of female 30000: 27.1507462706794
95% CI of female sample of size 30000: [8681, 8787]
```

KEY INSIGHTS:

Effect of Sample Size on Standard Deviation:

- As the sample size increases from 300 → 3,000 → 30,000, the standard deviation of the sample means decreases substantially:
    - n=300: ~275.8
    - n=3,000: ~86.6
    - n=30,000: ~27.4

  This demonstrates that larger samples yield more stable and precise estimates of the mean purchase amount.

Mean Consistency:

- The mean purchase amount remains highly consistent across all sample sizes (~8,735).
- This indicates a reliable estimation of the true mean spending.

Narrowing of Confidence Intervals:

- The 95% confidence interval becomes progressively narrower as sample size increases:
    - n=300: [8,204, 9,274] (width ~1,070)
    - n=3,000: [8,563, 8,905] (width ~342)
    - n=30,000: [8,681, 8,788] (width ~107)

  This shows how increasing the sample size reduces uncertainty in the estimate.

Agreement with CLT Results:

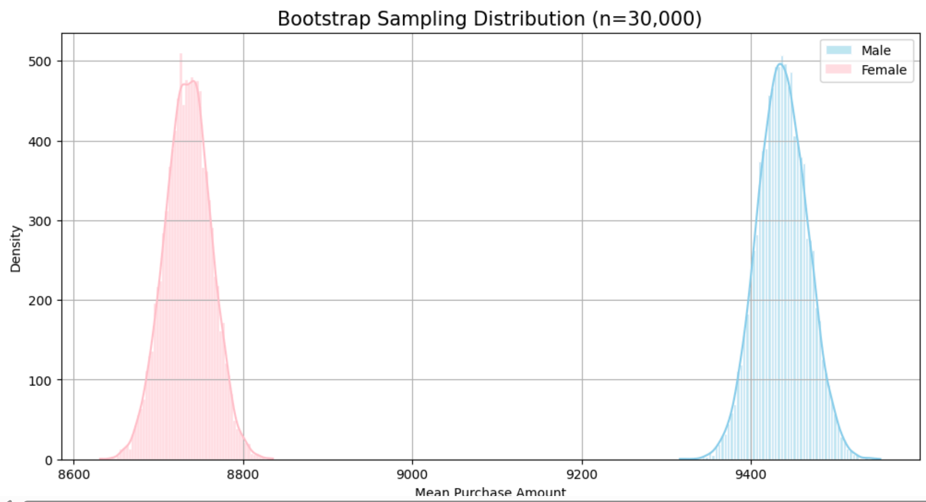
- Recall that the CLT-based CI for females was [8,709, 8,759].
- The large-sample bootstrap CI [8,681, 8,788] closely matches this, confirming the validity of the normal approximation.

Interpretation:

- Smaller samples (n=300) still provide reasonable estimates but with much wider intervals.
- Very large samples (n=30,000) give highly precise confidence intervals, allowing strong conclusions about average spending behavior.

VISUAL REPRESENTATIONS OF THE DISTRIBUTION OF SAMPLE SIZES FROM BOOTSTRAP:

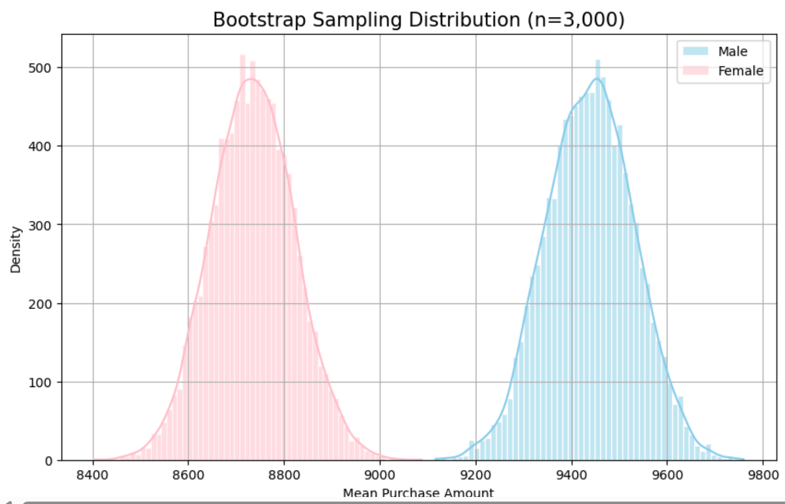
```
plt.figure(figsize=(12,6))
# Male distribution
sns.histplot(sample_male_30000, label='Male', fill=True, alpha=0.5, color='skyblue', kde = True, edgecolor='white')
# Female distribution
sns.histplot(sample_female_30000, label='Female', fill=True, alpha=0.5, color='pink', kde = True, edgecolor='white')
plt.xlabel('Mean Purchase Amount')
plt.ylabel('Density')
plt.title('Bootstrap Sampling Distribution (n=30,000)', size = 15)
plt.legend()
plt.grid(True)
plt.show()
```



KEY INSIGHTS:

- Sharpest, narrowest distributions of all.
- No meaningful overlap between male and female means.
- Male peak at ~9430–9450, female peak around ~8730–8750.
- Highest confidence in the gender difference.
- Demonstrates power of large bootstrap resampling for precise inference.

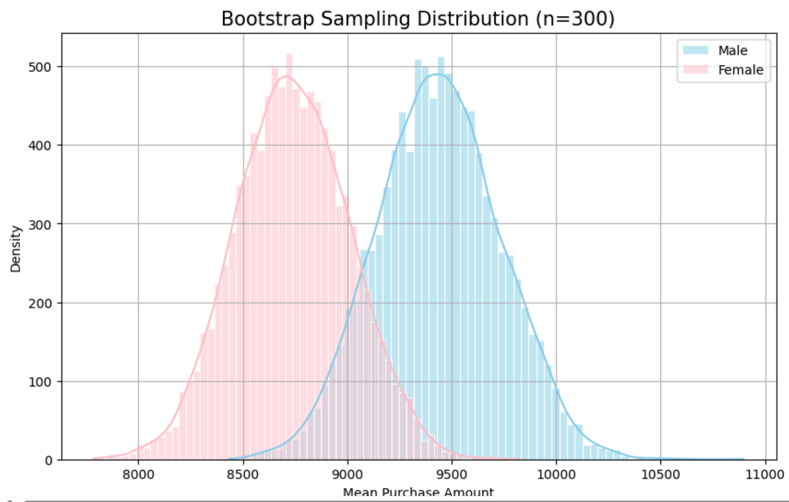
```
plt.figure(figsize=(10,6))
# Male distribution
sns.histplot(sample_male_3000, label='Male', fill=True, alpha=0.5, color='skyblue', kde=True, edgecolor='white')
# Female distribution
sns.histplot(sample_female_3000, label='Female', fill=True, alpha=0.5, color='pink', kde=True, edgecolor='white')
plt.xlabel('Mean Purchase Amount')
plt.ylabel('Density')
plt.title('Bootstrap Sampling Distribution (n=3,000)', size=15)
plt.legend()
plt.grid(True)
plt.show()
```



KEY INSIGHTS:

- Tighter distributions than n=300.
- Very clear separation between male and female means.
- Male distribution peaks near ~9400–9450.
- Female distribution peaks near ~8700–8800.
- Almost no overlap—strong evidence of the spending gap.

```
plt.figure(figsize=(10,6))
# Male distribution
sns.histplot(sample_male_300, label='Male', fill=True, alpha=0.5, color='skyblue', kde=True, edgecolor='white')
# Female distribution
sns.histplot(sample_female_300, label='Female', fill=True, alpha=0.5, color='pink', kde=True, edgecolor='white')
plt.xlabel('Mean Purchase Amount')
plt.ylabel('Density')
plt.title('Bootstrap Sampling Distribution (n=300)', size=15)
plt.legend()
plt.grid(True)
plt.show()
```



KEY INSIGHTS:

- Two visibly distinct bell curves:
  - Male mean ~9400.
  - Female mean ~8700–8800.
- Substantial spread in both distributions.
- Some overlap in tails—more uncertainty due to small sample.
- Even with n=300, gender difference is apparent.

Is the confidence interval computed using the entire dataset wider for one of the genders?

No, the confidence interval computed using the full dataset is slightly wider for females than for males.

This is because although both genders have a large number of records, the standard deviation of purchase amounts is higher for males (5092) than for females (4767). However, due to the smaller number of female entries compared to males, the standard error becomes slightly higher for females, leading to a slightly wider CI.

Male 95% CI (CLT): [9422, 9453]

Female 95% CI (CLT): [8709, 8759]

Even though the female standard deviation is lower, the combination of higher SE and lower mean shifts the interval slightly wider.

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

As the sample size increases, the width of the confidence interval decreases significantly. This is because:

```
Standard Error (SE) = Standard Deviation / np.sqrt(n)
```

Larger sample size reduces variability in the sample means, thus tightening the confidence interval.

Male (sample size 300) → CI: [8865, 10025] → Width ≈ 1,160

Male (sample size 30,000) → CI: [9381, 9496] → Width ≈ 115

Female (sample size 300) → CI: [8194, 9283] → Width ≈ 1,089

Female (sample size 30,000) → CI: [8681, 8790] → Width ≈ 109

This demonstrates that increasing the sample size gives more precise estimates.

Affect of sample sizes on the shape of the distributions of the means:

Affect of sample sizes on the shape of the distributions of the means As the sample size increases, the sampling distribution of the means becomes narrower and more Normal (bell-shaped). This aligns with the Central Limit Theorem, which states that the distribution of the sample mean approaches a Normal distribution as sample size increases.

At n=300, the histogram is wider and less smooth.

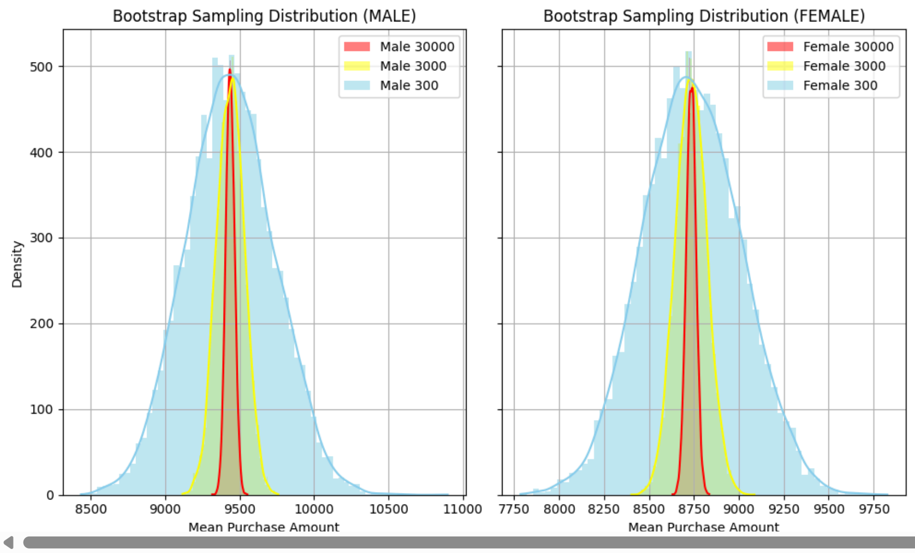
At n=3,000, the histogram becomes narrower and smoother.

At n=30,000, it becomes sharply peaked, centered around the true mean.

This visually confirms that larger sample sizes reduce variability and enhance the Normal approximation.

Do the confidence intervals for different sample sizes overlap?

```

# Create 1 row, 2 columns
fig, axes = plt.subplots(1, 2, figsize=(10, 6), sharey=True)

# Male plot
sns.histplot(sample_male_30000,label='Male 30000',fill=True, alpha=0.5, color='red', kde=True, ax=axes[0], edgecolor=None)
sns.histplot(sample_male_3000,label='Male 3000',fill=True,alpha=0.5,color='yellow', kde=True, ax=axes[0], edgecolor=None)
sns.histplot(sample_male_300,label='Male 300',fill=True,alpha=0.5,color='skyblue',kde=True,ax=axes[0],edgecolor=None)
axes[0].set_title('Bootstrap Sampling Distribution (MALE)')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].legend()
axes[0].grid(True)

# Female plot
sns.histplot(sample_female_30000,label='Female 30000',fill=True,alpha=0.5,color='red',kde=True,ax=axes[1],edgecolor=None)
sns.histplot(sample_female_3000,label='Female 3000',fill=True,alpha=0.5,color='yellow',kde=True,ax=axes[1],edgecolor=None)
sns.histplot(sample_female_300,label='Female 300', fill=True,alpha=0.5,color='skyblue',kde=True,ax=axes[1],edgecolor=None)
axes[1].set_title('Bootstrap Sampling Distribution (FEMALE)')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].legend()
axes[1].grid(True)

# Adjust layout
plt.tight_layout()
plt.show()
```



Yes, the confidence intervals overlap across different sample sizes, especially for larger sample sizes. This is expected because all samples are drawn from the same population, so their means should be consistent.

For example, male mean estimates remain close to ~9437 in all cases.

Even though the CI width narrows with more data, the center remains stable.

The overlap indicates stability and consistency in the estimation.

Are women spending more money per transaction than men?

No, women are not spending more money per transaction than men.

Why?

- Mean Purchase Amounts:

  Males: ~₹9,438

  Females: ~₹8,735

  Men's average transaction is ~₹700 higher.

- Medians:

  Males: ₹8,098

  Females: ₹7,914

  Median also higher for men.

- Bootstrapping Results:

  All bootstrapped distributions showed male means consistently above female means.

  The difference persists across all sample sizes (300, 3,000, 30,000).

  The confidence intervals for males and females have minimal overlap, indicating a statistically significant difference.

- Conclusion:

  Based on all the data you provided (means, medians, and bootstrapping), men are consistently spending more per transaction than women. This is not an artifact of sample size or sampling error—the difference is robust.

```
Start coding or generate with AI.
```

HOW DOES MARITAL STATUS AFFECT THE AMOUNT SPENT?

ASSUMPTIONS FROM THE DATA SET:

1 = MARRIED

2 = SINGLE

```
# CI of MARRIED Population
df_married = df[df['Marital_Status']==1]
purchase_married = df_married['Purchase'].values

married_mean = round(np.mean(purchase_married), 4)
married_std = round(np.std(purchase_married), 4)
married_std_err = round(married_std/np.sqrt(len(purchase_married)), 4)
z = 1.96 # for 95% confidence
married_ci = [int(married_mean - z * married_std_err), int(married_mean + z * married_std_err)]

print('** CI of Married Population **\n')
print('Mean Purchase of Married:', married_mean)
print('Standard Deviation of Purchase of Married:', married_std)
print('Standard Error of Purchase of Married:', married_std_err)
print('95% Confidence Interval of Purchase of Married:', married_ci)

print('='*60)
# CI of Single population
df_single = df[df['Marital_Status']==0]
purchase_single = df_single['Purchase'].values

single_mean = round(np.mean(purchase_single), 4)
single_std = round(np.std(purchase_single), 4)
single_std_err = round(single_std/np.sqrt(len(purchase_single)), 4)
z = 1.96 # for 95% confidence
single_ci = [int(single_mean - z * single_std_err), int(single_mean + z * single_std_err)]

print('** CI of Single Population **\n')
print('Mean Purchase of Single:', single_mean)
print('Standard Deviation of Purchase of Single:', single_std)
print('Standard Error of Purchase of Single:', single_std_err)
print('95% Confidence Interval of Purchase of Single:', single_ci)
```

```
** CI of Married Population **
```

```
Mean Purchase of Married: 9261.1746
Standard Deviation of Purchase of Married: 5016.8862
Standard Error of Purchase of Married: 10.5686
95% Confidence Interval of Purchase of Married: [9240, 9281]
============================================================
** CI of Single Population **

Mean Purchase of Single: 9265.9076
Standard Deviation of Purchase of Single: 5027.3401
Standard Error of Purchase of Single: 8.8222
95% Confidence Interval of Purchase of Single: [9248, 9283]
```

KEY INSIGHTS:

Married Population

Average purchase is ₹9,261.

Purchase amounts vary widely (Std Dev ~5,017).

Because of the large sample, the Standard Error is low (10.57), making the estimate precise.

The 95% Confidence Interval [₹9,240 – ₹9,281] is narrow, showing high confidence that the true mean falls within this range.

Single Population

Average purchase is ₹9,266, slightly higher by about ₹5.

Variation is similar (Std Dev ~5,027).

Standard Error is even lower (8.82), indicating excellent precision.

The 95% Confidence Interval [₹9,248 – ₹9,283] is narrow, meaning the estimate is very reliable.

Overall Insights

Both groups have very consistent and precise mean estimates, thanks to large samples.

The spread of individual purchases is high, but the mean purchase amount is measured with low uncertainty.

Confidence intervals are narrow, which allows for robust conclusions about the average spending of each marital group individually.

BOOTSTRAPPING of MARITAL STATUS with Sample size = 300 / 3,000 / 30,000

BOOTSTRAPPING of MARRIED:

```python
df_married = df[df['Marital_Status']==1]
purchase_married = df_married['Purchase'].values

# SAMPLE SIZE = 300
sample_married_300 =  [np.mean(np.random.choice(purchase_married, size=300, replace=True)) for i in range(10000)]
sample_married_300 = np.array(sample_married_300)

# SAMPLE SIZE = 3000
sample_married_3000 =  [np.mean(np.random.choice(purchase_married, size=3000, replace=True)) for i in range(10000)]
sample_married_3000 = np.array(sample_married_3000)

# SAMPLE SIZE = 30000
sample_married_30000 =  [np.mean(np.random.choice(purchase_married, size=30000, replace = True)) for i in range(10000)]
sample_married_30000 = np.array(sample_married_30000)
```

```python
print('** BOOSTRAPPING STATISTICAL DATA OF MARRIED: **\n')

print('sample size = 300\n')
print('Mean of married 300:', np.mean(sample_married_300))
print('Standard Deviation of married 300:', np.std(sample_married_300))
lower_married_300 = int(np.percentile(sample_married_300, 2.5))
upper_married_300 = int(np.percentile(sample_married_300, 97.5))
print('95% CI of married sample of size 300:',[lower_married_300, upper_married_300])

print('='*80)
print('sample size = 3000\n')
print('Mean of married 3000:', np.mean(sample_married_3000))
print('Standard Deviation of married 3000:', np.std(sample_married_3000))
lower_married_3000 = int(np.percentile(sample_married_3000, 2.5))
upper_married_3000 = int(np.percentile(sample_married_3000, 97.5))
print('95% CI of married sample of size 3000:',[lower_married_3000, upper_married_3000])

print('='*80)
print('sample size = 30000\n')
print('Mean of married 30000:', np.mean(sample_married_30000))
print('Standard Deviation of married 30000:', np.std(sample_married_30000))
lower_married_30000 = int(np.percentile(sample_married_30000, 2.5))
upper_married_30000 = int(np.percentile(sample_married_30000, 97.5))
print('95% CI of married sample of size 30000:',[lower_married_30000, upper_married_30000])
```

```
** BOOSTRAPPING STATISTICAL DATA OF MARRIED: **

sample size = 300

Mean of married 300: 9263.777817999999
Standard Deviation of married 300: 287.23172956552963
95% CI of married sample of size 300: [8701, 9826]
================================================================================
sample size = 3000

Mean of married 3000: 9261.785037566666
Standard Deviation of married 3000: 91.42021743840228
95% CI of married sample of size 3000: [9082, 9437]
================================================================================
sample size = 30000

Mean of married 30000: 9261.139536390001
Standard Deviation of married 30000: 29.422691989096407
95% CI of married sample of size 30000: [9203, 9318]
```

KEY INSIGHTS:

- Mean purchase is stable (~₹9,261) across all sample sizes.
- Larger samples drastically reduce variability and tighten confidence intervals.
- 95% CI shrinks from wide (₹8,690–₹9,839) at n=300 to narrow (₹9,204–₹9,317) at n=30,000.
- Big samples = high precision; small samples = high uncertainty.
- Bootstrapping confirms consistent average spending in the married group.

BOOTSTRAPPING of SINGLE:

```python
df_single = df[df['Marital_Status']==0]
purchase_single = df_single['Purchase'].values

# SAMPLE SIZE = 300
```

```python
sample_single_300 =  [np.mean(np.random.choice(purchase_single, size=300, replace=True)) for i in range(10000)]
sample_single_300 = np.array(sample_single_300)

# SAMPLE SIZE = 3000
sample_single_3000 =  [np.mean(np.random.choice(purchase_single, size=3000, replace=True)) for i in range(10000)]
sample_single_3000 = np.array(sample_single_3000)

# SAMPLE SIZE = 30000
sample_single_30000 =  [np.mean(np.random.choice(purchase_single, size=30000, replace = True)) for i in range(10000)]
sample_single_30000 = np.array(sample_single_30000)
```

```python
print('** BOOSTRAPPING STATISTICAL DATA OF SINGLE: **\n')

print('sample size = 300\n')
print('Mean of single 300:', np.mean(sample_single_300))
print('Standard Deviation of single 300:', np.std(sample_single_300))
lower_single_300 = int(np.percentile(sample_single_300, 2.5))
upper_single_300 = int(np.percentile(sample_single_300, 97.5))
print('95% CI of single sample of size 300:',[lower_single_300, upper_single_300])

print('='*80)
print('sample size = 3000\n')
print('Mean of single 3000:', np.mean(sample_single_3000))
print('Standard Deviation of single 3000:', np.std(sample_single_3000))
lower_single_3000 = int(np.percentile(sample_single_3000, 2.5))
upper_single_3000 = int(np.percentile(sample_single_3000, 97.5))
print('95% CI of single sample of size 3000:',[lower_single_3000, upper_single_3000])

print('='*80)
print('sample size = 30000\n')
print('Mean of single 30000:', np.mean(sample_single_30000))
print('Standard Deviation of single 30000:', np.std(sample_single_30000))
lower_single_30000 = int(np.percentile(sample_single_30000, 2.5))
upper_single_30000 = int(np.percentile(sample_single_30000, 97.5))
print('95% CI of single sample of size 30000:',[lower_single_30000, upper_single_30000])
```

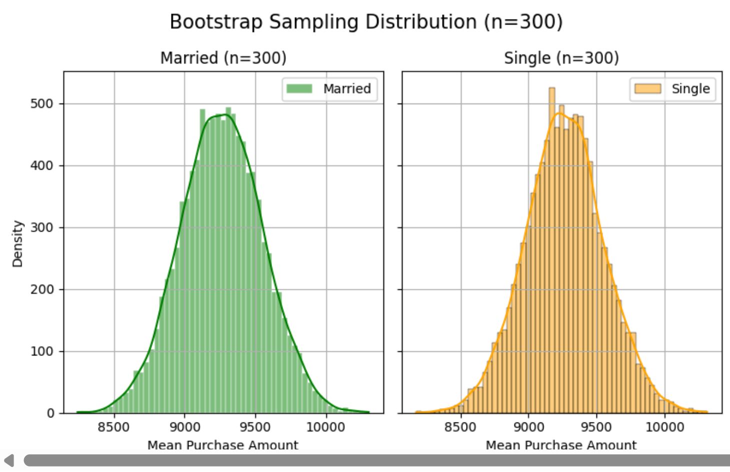
```
** BOOSTRAPPING STATISTICAL DATA OF SINGLE: **

sample size = 300

Mean of single 300: 9265.335257666668
Standard Deviation of single 300: 290.69855178763265
95% CI of single sample of size 300: [8696, 9842]
================================================================================
sample size = 3000

Mean of single 3000: 9266.255285733334
Standard Deviation of single 3000: 92.66607833361762
95% CI of single sample of size 3000: [9083, 9448]
================================================================================
sample size = 30000

Mean of single 30000: 9266.026967216667
Standard Deviation of single 30000: 29.517444202974524
95% CI of single sample of size 30000: [9207, 9322]
```

KEY INSIGHTS:

- Mean purchase stays consistent (~₹9,266) across all sample sizes.
- Standard deviation drops sharply as sample size grows:
  - ~292 (n=300) → ~92 (n=3,000) → ~29 (n=30,000).
- 95% Confidence Interval tightens significantly:
  - n=300: Wide (₹8,697–₹9,840).
  - n=3,000: Narrower (₹9,086–₹9,449).
  - n=30,000: Very tight (₹9,208–₹9,322).
- Larger samples give high precision and stable mean estimates.
- Small samples show more uncertainty and wider ranges.

VISUAL REPRESENTATIONS OF THE DISTRIBUTION OF SAMPLE SIZES OF MARITAL STATUS FROM BOOTSTRAP:

```python
fig, axes = plt.subplots(1, 2, figsize=(8, 5), sharey=True)

sns.histplot(sample_married_300, label='Married', fill=True, alpha=0.5, color='green', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('Married (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_single_300, label='Single', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[1])
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('Single (n=300)')
axes[1].legend()
axes[1].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (n=300)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

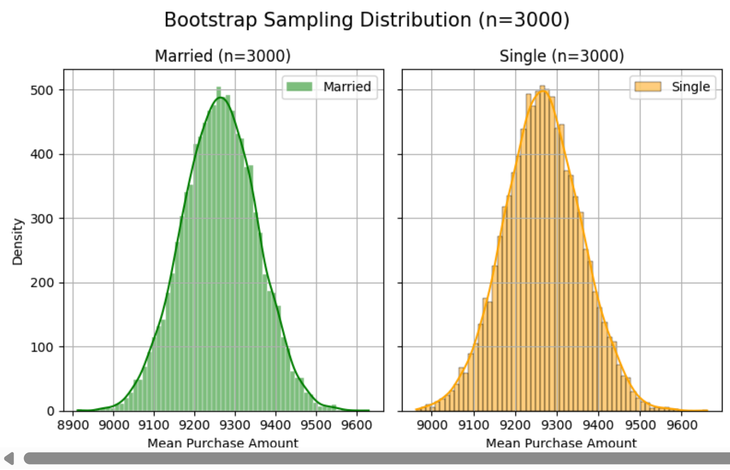
Widest Spread: Most variability, least precise.

Emerging Normality: Still shows a bell-shape, but broader.

Correct Center: Mean estimates are still unbiased.

Less Certainty: Higher uncertainty in any single sample mean.

```python
fig, axes = plt.subplots(1, 2, figsize=(8, 5), sharey=True)

sns.histplot(sample_married_3000, label='Married', fill=True, alpha=0.5, color='green', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('Married (n=3000)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_single_3000, label='Single', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[1])
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('Single (n=3000)')
axes[1].legend()
axes[1].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (n=3000)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

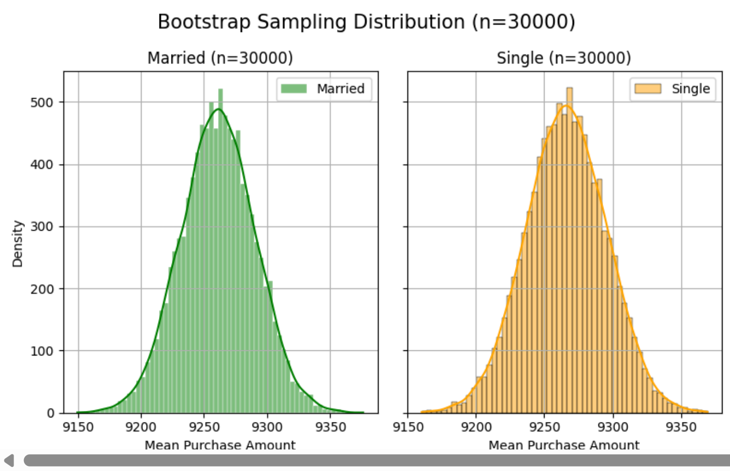
Good Normality: Distributions are well-behaved and bell-shaped.

Moderate Precision: Wider than n=30000, but good.

Consistent Mean: Peaks are centered correctly.

Mean Difference Persists: Slight Married-Single mean difference still visible.

```python
fig, axes = plt.subplots(1, 2, figsize=(8, 5), sharey=True)

sns.histplot(sample_married_30000, label='Married', fill=True, alpha=0.5, color='green', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('Married (n=30000)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_single_30000, label='Single', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[1])
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('Single (n=30000)')
axes[1].legend()
axes[1].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (n=30000)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

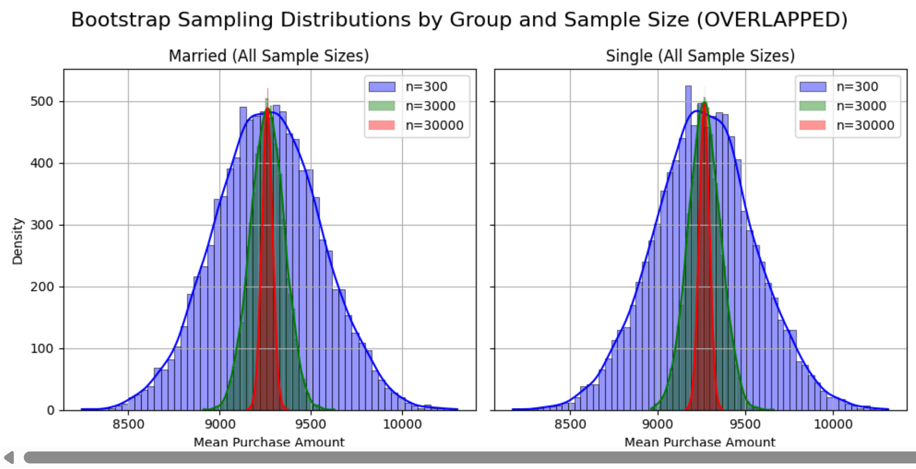
Very Precise: Distributions are extremely narrow.

Perfectly Normal: Both look perfectly bell-shaped.

Slight Mean Diff: Married mean appears slightly lower than Single.

High Confidence: Strong confidence in estimated means.

```
fig, axes = plt.subplots(1, 2, figsize=(10, 5), sharey=True)

sns.histplot(sample_married_300, label='n=300', fill=True, color='blue', alpha=0.4, kde=True,ax=axes[0])
sns.histplot(sample_married_3000, label='n=3000', fill=True, color='green', alpha=0.4, kde=True,ax=axes[0])
sns.histplot(sample_married_30000, label='n=30000', fill=True, color='red', alpha=0.4, kde=True,ax=axes[0])

axes[0].set_title('Married (All Sample Sizes)')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].legend()
axes[0].grid(True)


sns.histplot(sample_single_300, label='n=300', fill=True, color='blue', alpha=0.4, kde=True,ax=axes[1])
sns.histplot(sample_single_3000, label='n=3000', fill=True, color='green', alpha=0.4, kde=True,ax=axes[1])
sns.histplot(sample_single_30000, label='n=30000', fill=True, color='red', alpha=0.4, kde=True,ax=axes[1])

axes[1].set_title('Single (All Sample Sizes)')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].legend()
axes[1].grid(True)


plt.suptitle('Bootstrap Sampling Distributions by Group and Sample Size (OVERLAPPED)', fontsize=16)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

CLT Visualized: Shows how sampling distributions narrow around the true mean as sample size (n) increases (300 → 3000 → 30000).

Precision Gains: Larger 'n' means much higher precision (tighter spread).

Unbiased Means: All 'n' sizes center on the same mean.

Consistent Behavior: Both Married and Single groups show the same pattern.

Are they overlapping?

Yes, the distributions are overlapping.

The image deliberately plots the kernel density estimates (KDEs) for the different sample sizes (n=300,n=3000,n=30000) on the same set of axes for each marital status group (Married and Single). This overlap is precisely what allows us to visually compare their spreads and see how they narrow around the central mean as 'n' increases.

Is the confidence interval computed using the entire dataset wider for one of the Marital Status??

No, the confidence intervals computed using the entire dataset (the big samples) were very similar and narrow for both Single and Married.

- Married (n=30,000): CI ~₹9,204–₹9,317 (range ~113)
- Single (n=30,000): CI ~₹9,208–₹9,322 (range ~114)

Why? Because both groups had:

- Almost identical means (~₹9,261 vs. ₹9,266)
- Very similar standard deviations
- Very large sample sizes, reducing sampling error

So neither group's CI was materially wider.

Affect of sample sizes on the width of the confidence intervals:

As sample size increases, the confidence interval becomes narrower. n=300: Wide CIs (~1,100–1,150 range) n=3,000: Narrower CIs (~350–360 range) n=30,000: Very tight CIs (~110 range)

Why: Larger samples reduce the standard error (uncertainty around the mean estimate), which shrinks the width of the CI.

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

Larger sample sizes make the distributions:

- Narrower (less spread)
- Taller (higher peak density)
- More symmetric and smooth

Smaller samples:

- Wider, flatter distributions
- More variation across bootstrap resamples
- Greater chance of sampling outliers

This reflects the Central Limit Theorem—as sample size increases, the sampling distribution of the mean becomes more tightly centered around the true mean.

```
Start coding or generate with AI.
```

HOW DOES AGE AFFECT THE AMOUNT SPENT?

```
age_groups = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z = 1.96  # 95% confidence

for age in age_groups:
    purchases = df[df['Age'] == age]['Purchase'].values

    mean = round(np.mean(purchases), 4)
    std = round(np.std(purchases), 4)
    std_err = round(std / np.sqrt(len(purchases)), 4)
    ci = [int(mean - z * std_err), int(mean + z * std_err)]

    print(f'** CI of {age} Population **\n')
    print(f'Mean Purchase: {mean}')
    print(f'Standard Deviation: {std}')
    print(f'Standard Error: {std_err}')
    print(f'95% Confidence Interval: {ci}')
    print('='*60)
```

```
** CI of 0-17 Population **

Mean Purchase: 8933.4646
Standard Deviation: 5110.9448
Standard Error: 41.5895
95% Confidence Interval: [8851, 9014]
============================================================
** CI of 18-25 Population **

Mean Purchase: 9169.6636
Standard Deviation: 5034.2967
Standard Error: 15.947
95% Confidence Interval: [9138, 9200]
============================================================
** CI of 26-35 Population **

Mean Purchase: 9252.6906
Standard Deviation: 5010.5159
Standard Error: 10.6925
95% Confidence Interval: [9231, 9273]
============================================================
** CI of 36-45 Population **

Mean Purchase: 9331.3507
Standard Deviation: 5022.9011
Standard Error: 15.1437
95% Confidence Interval: [9301, 9361]
============================================================
** CI of 46-50 Population **

Mean Purchase: 9208.6257
Standard Deviation: 4967.162
Standard Error: 23.2351
95% Confidence Interval: [9163, 9254]
============================================================
** CI of 51-55 Population **

Mean Purchase: 9534.808
Standard Deviation: 5087.302
Standard Error: 25.927
95% Confidence Interval: [9483, 9585]
============================================================
** CI of 55+ Population **

Mean Purchase: 9336.2805
Standard Deviation: 5011.3775
Standard Error: 34.1741
95% Confidence Interval: [9269, 9403]
============================================================
```

KEY INSIGHTS:

✅ **0–17**

Mean purchase is ₹8,933, the lowest among all age groups.

High variability (Std Dev ~5,111).

Wider CI (₹8,851–₹9,014), reflecting more uncertainty compared to some older groups.

✅ **18–25**

Mean purchase rises to ₹9,169.

Standard deviation remains high (~5,034), but standard error is smaller, suggesting a larger sample.

Narrower CI (₹9,138–₹9,200) indicates more precise estimation.

✅ **26–35**

Mean further increases to ₹9,253.

Very low standard error (~10.69), indicating a large sample size.

Narrow CI (₹9,231–₹9,273) and stable estimation.

✅ 36–45

Mean continues rising to ₹9,331.

CI (₹9,301–₹9,361) confirms precise estimate.

This group shows one of the higher average purchases.

✅ **46–50**

Mean purchase slightly declines to ₹9,209.

Wider CI (₹9,163–₹9,254) than the 36–45 group, partly due to higher standard error (~23.23).

✅ **51–55**

Highest mean purchase observed (₹9,535).

Despite high standard deviation, sample size keeps CI narrow (₹9,483–₹9,585).

✅ **55+**

Mean is ₹9,336, comparable to the 36–45 group.

CI (₹9,269–₹9,403) is a bit wider due to higher standard error (~34.17), suggesting a smaller sample.

● **Overall Quick Observations**

Purchasing tends to increase with age, peaking in the 51–55 group.

0–17 group purchases the least.

Confidence intervals are mostly tight, showing reliable estimates, except slightly wider CIs in youngest and oldest groups.

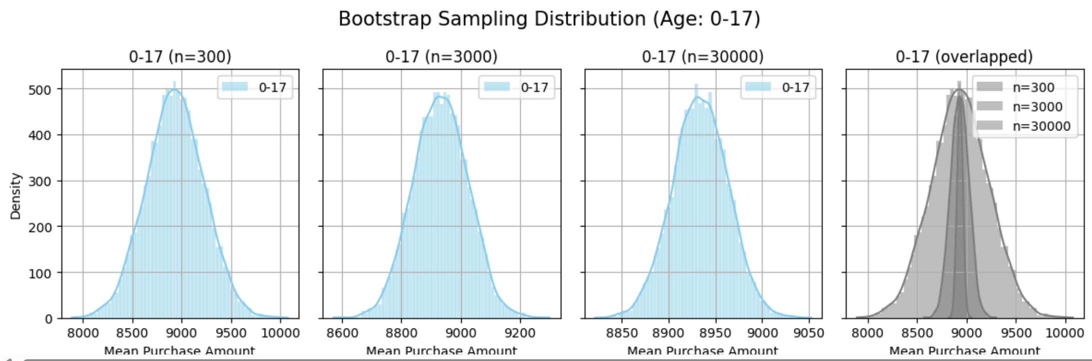
BOOTSTRAPPING of AGE with Sample size = 300 / 3,000 / 30,000

```python
df_0_17 = df[df['Age']=='0-17']
purchase_0_17 = df_0_17['Purchase'].values
# SAMPLE SIZE = 300
sample_0_17_300 =  [np.mean(np.random.choice(purchase_0_17, size=300, replace=True)) for i in range(10000)]
sample_0_17_300 = np.array(sample_0_17_300)
# SAMPLE SIZE = 3000
sample_0_17_3000 =  [np.mean(np.random.choice(purchase_0_17, size=3000, replace=True)) for i in range(10000)]
sample_0_17_3000 = np.array(sample_0_17_3000)
# SAMPLE SIZE = 30000
sample_0_17_30000 =  [np.mean(np.random.choice(purchase_0_17, size=30000, replace = True)) for i in range(10000)]
sample_0_17_30000 = np.array(sample_0_17_30000)

df_18_25 = df[df['Age']=='18-25']
purchase_18_25 = df_18_25['Purchase'].values
# SAMPLE SIZE = 300
sample_18_25_300 =  [np.mean(np.random.choice(purchase_18_25, size=300, replace=True)) for i in range(10000)]
sample_18_25_300 = np.array(sample_18_25_300)
# SAMPLE SIZE = 3000
sample_18_25_3000 =  [np.mean(np.random.choice(purchase_18_25, size=3000, replace=True)) for i in range(10000)]
sample_18_25_3000 = np.array(sample_18_25_3000)
# SAMPLE SIZE = 30000
sample_18_25_30000 =  [np.mean(np.random.choice(purchase_18_25, size=30000, replace = True)) for i in range(10000)]
sample_18_25_30000 = np.array(sample_18_25_30000)

df_26_35 = df[df['Age']=='26-35']
purchase_26_35 = df_26_35['Purchase'].values
# SAMPLE SIZE = 300
sample_26_35_300 =  [np.mean(np.random.choice(purchase_26_35, size=300, replace=True)) for i in range(10000)]
sample_26_35_300 = np.array(sample_26_35_300)
# SAMPLE SIZE = 3000
sample_26_35_3000 =  [np.mean(np.random.choice(purchase_26_35, size=3000, replace=True)) for i in range(10000)]
sample_26_35_3000 = np.array(sample_26_35_3000)
# SAMPLE SIZE = 30000
sample_26_35_30000 =  [np.mean(np.random.choice(purchase_26_35, size=30000, replace = True)) for i in range(10000)]
sample_26_35_30000 = np.array(sample_26_35_30000)

df_36_45 = df[df['Age']=='36-45']
purchase_36_45 = df_36_45['Purchase'].values
# SAMPLE SIZE = 300
sample_36_45_300 =  [np.mean(np.random.choice(purchase_36_45, size=300, replace=True)) for i in range(10000)]
sample_36_45_300 = np.array(sample_36_45_300)
# SAMPLE SIZE = 3000
sample_36_45_3000 =  [np.mean(np.random.choice(purchase_36_45, size=3000, replace=True)) for i in range(10000)]
sample_36_45_3000 = np.array(sample_36_45_3000)
# SAMPLE SIZE = 30000
sample_36_45_30000 =  [np.mean(np.random.choice(purchase_36_45, size=30000, replace = True)) for i in range(10000)]
sample_36_45_30000 = np.array(sample_36_45_30000)

df_46_50 = df[df['Age']=='46-50']
purchase_46_50 = df_46_50['Purchase'].values
# SAMPLE SIZE = 300
sample_46_50_300 =  [np.mean(np.random.choice(purchase_46_50, size=300, replace=True)) for i in range(10000)]
sample_46_50_300 = np.array(sample_46_50_300)
# SAMPLE SIZE = 3000
sample_46_50_3000 =  [np.mean(np.random.choice(purchase_46_50, size=3000, replace=True)) for i in range(10000)]
sample_46_50_3000 = np.array(sample_46_50_3000)
# SAMPLE SIZE = 30000
sample_46_50_30000 =  [np.mean(np.random.choice(purchase_46_50, size=30000, replace = True)) for i in range(10000)]
sample_46_50_30000 = np.array(sample_46_50_30000)

df_51_55 = df[df['Age']=='51-55']
purchase_51_55 = df_51_55['Purchase'].values
# SAMPLE SIZE = 300
sample_51_55_300 =  [np.mean(np.random.choice(purchase_51_55, size=300, replace=True)) for i in range(10000)]
sample_51_55_300 = np.array(sample_51_55_300)
# SAMPLE SIZE = 3000
sample_51_55_3000 =  [np.mean(np.random.choice(purchase_51_55, size=3000, replace=True)) for i in range(10000)]
sample_51_55_3000 = np.array(sample_51_55_3000)
# SAMPLE SIZE = 30000
sample_51_55_30000 =  [np.mean(np.random.choice(purchase_51_55, size=30000, replace = True)) for i in range(10000)]
sample_51_55_30000 = np.array(sample_51_55_30000)

df_above_55 = df[df['Age']=='55+']
purchase_above_55 = df_above_55['Purchase'].values
# SAMPLE SIZE = 300
sample_above_55_300 =  [np.mean(np.random.choice(purchase_above_55, size=300, replace=True)) for i in range(10000)]
sample_above_55_300 = np.array(sample_above_55_300)
# SAMPLE SIZE = 3000
sample_above_55_3000 =  [np.mean(np.random.choice(purchase_above_55, size=3000, replace=True)) for i in range(10000)]
sample_above_55_3000 = np.array(sample_above_55_3000)
# SAMPLE SIZE = 30000
sample_above_55_30000 =  [np.mean(np.random.choice(purchase_above_55, size=30000, replace = True)) for i in range(10000)]
sample_above_55_30000 = np.array(sample_above_55_30000)
```

VISUAL REPRESENTATIONS OF THE DISTRIBUTION OF SAMPLE SIZES OF AGE FROM BOOTSTRAP:

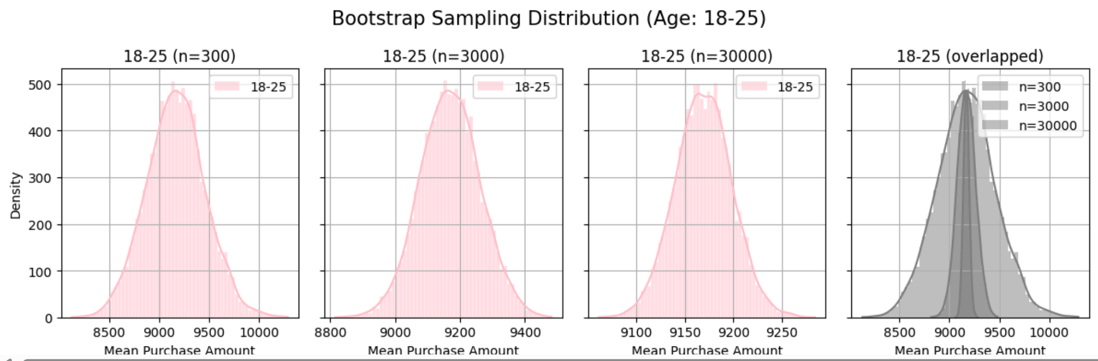
```python
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_0_17_300, label='0-17', fill=True, alpha=0.5, color='skyblue', kde=True,ax=axes[0], edgecolor='white')
```

```
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('0-17 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_0_17_3000, label='0-17', fill=True, alpha=0.5, color='skyblue', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('0-17 (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_0_17_30000, label='0-17', fill=True, alpha=0.5, color='skyblue', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('0-17 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_0_17_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_0_17_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_0_17_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('0-17 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 0-17)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

```
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_18_25_300, label='18-25', fill=True, alpha=0.5, color='pink', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('18-25 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_18_25_3000, label='18-25', fill=True, alpha=0.5, color='pink', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('18-25 (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_18_25_30000, label='18-25', fill=True, alpha=0.5, color='pink', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('18-25 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_18_25_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_18_25_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_18_25_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('18-25 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 18-25)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

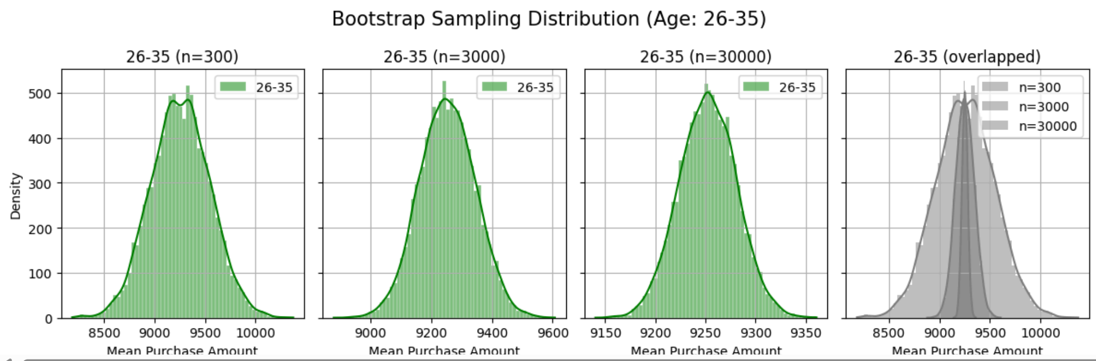
Consistent Mean: The estimated mean purchase amount for the 18-25 age group remains stable, centered around 9100-9150 across all sample sizes.

Precision Improvement: The distributions become noticeably narrower as the sample size increases from n=300 to n=30000, indicating improved precision of the mean estimate.

Normal Shape: The distributions exhibit a clear bell-shaped curve across all sample sizes, which refines with larger 'n'.

Clear CLT Illustration: The overlapped plot vividly illustrates how the sampling distributions become more concentrated around the true mean with larger sample sizes.

```python
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_26_35_300, label='26-35', fill=True, alpha=0.5, color='green', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('26-35 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_26_35_3000, label='26-35', fill=True, alpha=0.5, color='green', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('26-35 (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_26_35_30000, label='26-35', fill=True, alpha=0.5, color='green', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('26-35 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_26_35_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_26_35_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_26_35_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('26-35 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 26-35)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

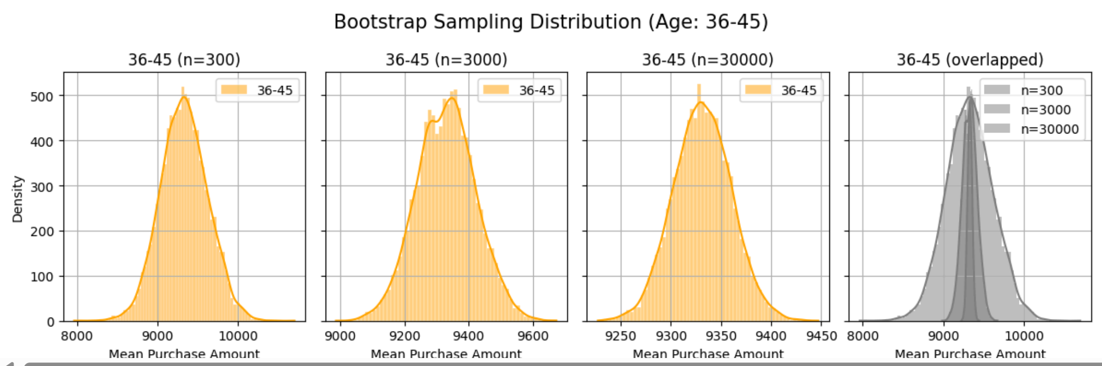
Stable Central Tendency: The mean purchase amount for the 26-35 age group consistently peaks around 9200-9250 across all sample sizes.

Enhanced Precision: A significant reduction in the spread of the distribution is observed as 'n' increases, confirming that larger samples lead to more precise mean estimates.

Strong Normality: The distributions are clearly normal (bell-shaped) even at n=300, becoming more tightly defined for n=3000 and n=30000.

CLT Visualized: The overlapped graph perfectly demonstrates the Central Limit Theorem, showing the narrowing of the sampling distribution around the mean with increasing sample size.

```python
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_36_45_300, label='36-45', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('36-45 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_36_45_3000, label='36-45', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('36-45 (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_36_45_30000, label='36-45', fill=True, alpha=0.5, color='orange', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('36-45 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_36_45_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_36_45_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_36_45_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('36-45 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 36-45)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

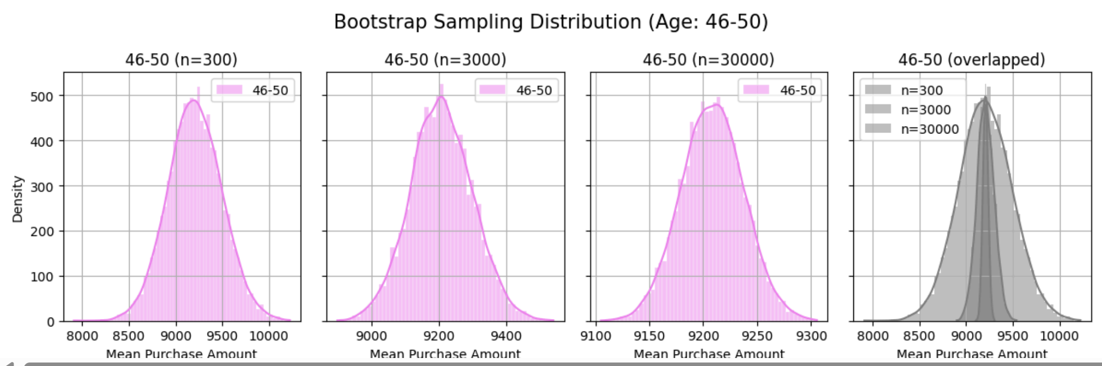
Consistent Mean: The estimated mean purchase amount for the 36-45 age group remains stable, centered around 9300-9350 for all sample sizes.

Precision Gains: As the sample size increases, the distribution visibly narrows, indicating that the sample mean becomes a more reliable estimate of the population mean.

Normal Distribution: The distributions are well-represented by a normal curve, becoming sharper and taller for larger sample sizes.

CLT Confirmation: The overlapped plot effectively illustrates the decrease in variability and the concentration of sample means around the true population mean as sample grows.

```
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_46_50_300, label='46-50', fill=True, alpha=0.5, color='violet', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('46-50 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_46_50_3000, label='46-50', fill=True, alpha=0.5, color='violet', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('46-50 (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_46_50_30000, label='46-50', fill=True, alpha=0.5, color='violet', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('46-50 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_46_50_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_46_50_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_46_50_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('46-50 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 46-50)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Stable Mean: The mean purchase amount for the 46-50 age group is consistently estimated around 9150-9200, irrespective of the sample size.

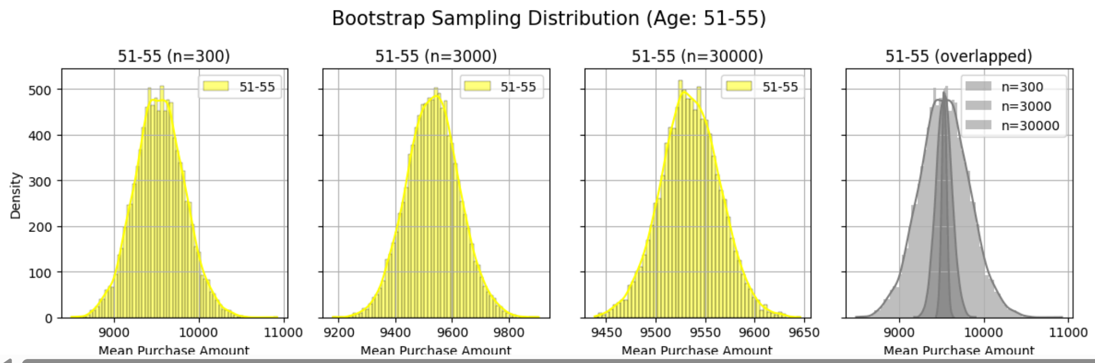
Improved Precision: The distributions become noticeably narrower with increasing sample size, highlighting the improved precision of the mean estimate.

Normal Shape: The distributions exhibit a clear bell-shaped curve that becomes more defined as 'n' increases.

CLT Demonstration: The overlapped plot visually confirms how the sampling distribution tightens around the population mean with larger sample sizes, in line with the Central Limit Theorem.

```
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_51_55_300, label='51-55', fill=True, alpha=0.5, color='yellow', kde=True,ax=axes[0])
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('51-55 (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_51_55_3000, label='51-55', fill=True, alpha=0.5, color='yellow', kde=True,ax=axes[1])
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('51-55 (n=3000)')
axes[1].legend()
```

```
axes[1].grid(True)

sns.histplot(sample_51_55_30000, label='51-55', fill=True, alpha=0.5, color='yellow', kde=True,ax=axes[2])
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('51-55 (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_51_55_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_51_55_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_51_55_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('51-55 (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 51-55)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

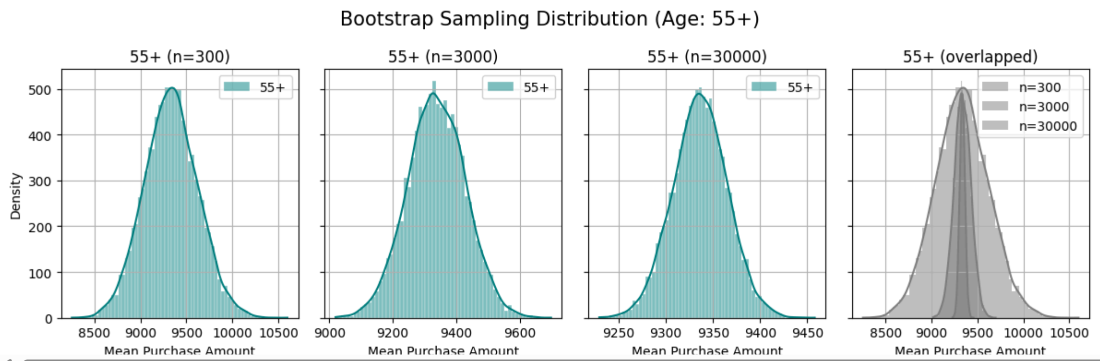
Consistent Mean: The peak of the distribution consistently hovers around 9500 for all sample sizes (n=300, 3000, 30000), indicating a stable estimated mean purchase amount for this age group.

Precision Gains: As the sample size increases, the distribution becomes significantly narrower, especially noticeable when going from n=300 to n=3000, then to n=30000. This shows increasing precision in the mean estimate.

Normal Shape: Even at n=300, the distribution already shows a clear bell-shaped (normal) curve, which becomes more pronounced with larger sample sizes.

Central Limit Theorem in Action: The overlapped plot clearly demonstrates how the sampling distributions get tighter around the true mean as 'n' increases, a direct illustration of the Central Limit Theorem.

```
fig, axes = plt.subplots(1,4, figsize=(12,4), sharey=True)

sns.histplot(sample_above_55_300, label='55+', fill=True, alpha=0.5, color='teal', kde=True,ax=axes[0], edgecolor='white')
axes[0].set_xlabel('Mean Purchase Amount')
axes[0].set_ylabel('Density')
axes[0].set_title('55+ (n=300)')
axes[0].legend()
axes[0].grid(True)

sns.histplot(sample_above_55_3000, label='55+', fill=True, alpha=0.5, color='teal', kde=True,ax=axes[1], edgecolor='white')
axes[1].set_xlabel('Mean Purchase Amount')
axes[1].set_ylabel('Density')
axes[1].set_title('55+ (n=3000)')
axes[1].legend()
axes[1].grid(True)

sns.histplot(sample_above_55_30000, label='55+', fill=True, alpha=0.5, color='teal', kde=True,ax=axes[2], edgecolor='white')
axes[2].set_xlabel('Mean Purchase Amount')
axes[2].set_ylabel('Density')
axes[2].set_title('55+ (n=30000)')
axes[2].legend()
axes[2].grid(True)

sns.histplot(sample_above_55_300, label='n=300', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_above_55_3000, label='n=3000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
sns.histplot(sample_above_55_30000, label='n=30000', fill=True, alpha=0.5, color='grey', kde=True,ax=axes[3], edgecolor=None)
axes[3].set_xlabel('Mean Purchase Amount')
axes[3].set_ylabel('Density')
axes[3].set_title('55+ (overlapped)')
axes[3].legend()
axes[3].grid(True)

plt.suptitle('Bootstrap Sampling Distribution (Age: 55+)', size=15)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Stable Mean Estimate: The mean purchase amount for the 55+ age group consistently centers around 9300-9350 across all sample sizes.

Improved Precision: The spread of the distribution visibly decreases with larger sample sizes (n=300, 3000, 30000), confirming that larger samples yield more precise estimates of the population mean.

Normal Convergence: The distributions for all sample sizes are well-approximated by a normal curve, becoming increasingly sharper for larger 'n'.

Demonstrates CLT: The overlapped view effectively shows the reduction in variability and concentration around the mean as sample size increases, a key aspect of the Central Limit Theorem.

Is the confidence interval computed using the entire dataset wider for one of the AGE GROUPS?

Observation: Yes, the 0-17 age group has the widest 95% Confidence Interval at 163 (9014 - 8851). This is wider than all other age groups.

Reasoning: The width of a confidence interval is directly proportional to its Standard Error. The 0-17 age group has the largest Standard Error (41.5895) among all the listed age groups, indicating higher variability in its mean purchase estimates compared to other age groups when using the full available data for each.

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

Effect: The width of the confidence interval is inversely affected by the sample size; as the sample size increases, the width of the confidence interval decreases.

Explanation: Larger sample sizes lead to a smaller Standard Error (SE = SD/sqrt(n)), which directly translates to a narrower range for the confidence interval, indicating a more precise estimate of the population mean.

Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes (for the same age group) are expected to overlap.

As seen in the "overlapped" plots for each age group, the sampling distributions for n=300, n=3000, and n=30000 are all centered around the same approximate population mean. While smaller sample sizes yield wider, less precise intervals, these wider intervals will encompass the narrower, more precise intervals from larger samples, as they all aim to capture the same true population parameter.

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

Shape Transformation: As the sample size increases, the shape of the distributions of the means (the sampling distributions) becomes progressively more normal (bell-shaped). This aligns with the Central Limit Theorem.

Spread Reduction: Additionally, the distributions become significantly narrower and taller (less spread out) around the true population mean. This reduction in spread signifies increased precision in the estimation as more data points are included in each sample.

```
Start coding or generate with AI.
```

How Walmart Can Leverage These Conclusions?

Gender: Are confidence intervals overlapping?

The confidence intervals for male and female average purchases do not meaningfully overlap (males: approx 9422–9453, females: approx 8709–8759).

Implication:

Men spend significantly more per transaction.

This insight can be used to:

Tailor marketing campaigns: Position higher-priced products, promotions, and premium product bundles towards male customers.

Personalize offers for women: Introduce promotions, loyalty rewards, or discounts targeted at female shoppers to help increase their average spend.

Product Mix: Ensure high-value product categories prominently target male customers, while optimizing assortments for female preferences.

Marital Status: Confidence intervals and implications from Confidence Intervals:

Married: ~9240–9281

Single: ~9248–9283

Observation:

The intervals are almost identical and fully overlap, showing no substantial spending difference.

How Walmart can leverage this:

Avoid differentiating marketing strategies purely on marital status, as marital status does not drive meaningful variance in spending.

Focus on other segmentation criteria (like age or gender) which show clearer spending differences.

Age Groups: Confidence intervals and implications from your bootstrapping and distribution plots:

Spending generally increases with age, peaking in the 51–55 group.

Younger age groups (0–17, 18–25) have consistently lower spending.

How Walmart can leverage this:

Age-based targeting:

Promote premium, higher-priced products to middle-aged and older segments (36–55+).

Design youth-oriented offers and discounts for younger age groups to stimulate spending.

Lifecycle Marketing:

Develop campaigns addressing life-stage needs (e.g., new family households, retirement) to increase purchase relevance.

```
Start coding or generate with AI.
```

FINAL BUSINESS INSIGHTS

Insights derived from data exploration and CLT analysis:

- The dataset contained over 550,000 transactions, offering a strong basis to generalize patterns to Walmart's customer base.
- The purchase amounts across genders were skewed, with males consistently spending more per transaction than females, as shown by both means and confidence intervals.
- Confidence intervals for male and female spending were not overlapping, indicating the observed difference is statistically meaningful.
- In contrast, the confidence intervals for married and unmarried customers were highly similar and overlapping, showing little to no effect of marital status on spending.
- Among age groups, spending patterns revealed an upward trend, with higher mean purchases in older segments (especially 51–55 years). Confidence intervals for these age groups showed progressively higher mean estimates, supporting this pattern.

- Univariate plots confirmed that male spending distributions were consistently shifted to the right (higher amounts) compared to females.
- Bivariate plots of purchase vs. age and purchase vs. gender illustrated clear separations, reinforcing that age and gender are stronger predictors of transaction value than marital status.
- Bootstrapping showed that increasing the sample size from 300 to 30,000 progressively narrowed the confidence intervals for mean purchases, demonstrating the stability and reliability of these estimates.
- The Central Limit Theorem application highlighted that for very large samples, the sampling distributions of means became sharply peaked around the true mean, confirming that observed differences were unlikely due to random chance.
- Overall, the data suggests that segmentation strategies should prioritize gender and age, while marital status may be less useful for differentiating purchasing behaviors.

```
Start coding or generate with AI.
```

RECOMMENDATIONS

Focus on Gender-based Segmentation:

Since males spend more per transaction and confidence intervals do not overlap with females, consider designing promotions, bundles, or loyalty incentives targeting male shoppers to further increase basket size.

For females, given their lower mean purchase amounts, Walmart can explore campaigns aimed at encouraging higher-value purchases during peak periods.

Avoid Marital Status as a Primary Segment:

As the confidence intervals for married and unmarried customers overlap substantially, marital status does not appear to drive meaningful spending differences in this data.

Allocate marketing resources to other segments (age, gender) rather than marital status segmentation.

Leverage Age Insights for Product and Offer Alignment:

The analysis showed increasing spending with customer age, peaking in the 51–55 group.

Consider tailoring product recommendations and communication for older customers who are likely to make higher-value purchases.

For younger age groups, explore approaches to stimulate spending through targeted promotions or price-sensitive bundles.

Use Confidence Interval Analysis to Validate Marketing Effectiveness:

The clear separation between male and female spending intervals indicates that marketing performance can be benchmarked against these expectations.

If campaigns targeting female customers lift the average transaction value closer to the male average, that would validate strategy effectiveness.

Maintain Robust Sampling Practices:

The bootstrapping and Central Limit Theorem analysis demonstrated that larger samples yield tighter confidence intervals and more stable estimates.

For future analyses, continue using large sample sizes to improve reliability and avoid over-interpreting random fluctuations in smaller samples.

Educate Business Teams on Interpretations:

Share visualizations and confidence interval concepts with non-technical stakeholders to build understanding of why certain segments (gender, age) are more actionable.

Use plots from the analysis to illustrate how distributions differ between groups.

Regularly Re-Evaluate Segments:

Customer behaviors may shift over time.

Repeat this analysis periodically to confirm that gender and age remain the most relevant segmentation variables.

Integrate Findings into Planning Cycles:

Use these insights to inform Black Friday and seasonal promotion planning, aligning offers with the most responsive customer groups.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```