```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
import scipy.stats as stats
```

Business Case: Walmart - Confidence Interval and CLT

- 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

1. Defining Problem Statement and Analyzing basic metrics

- The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

```python
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```python
data =pd.read_csv("/content/drive/MyDrive/walmart_data.csv")
```

```python
data.head()
```

{"type":"dataframe","variable_name":"data"}

```python
data.shape
```

```
(550068, 10)
```

- This shows that we have 550068 rows and 10 columns.

```python
data.size
```

```
5500680
```

```python
data.ndim
```

```
2
```

- Data is two-dimensional

```
data.columns
```

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

```
data.dtypes
```

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

```
data.nunique()
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
```

```
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

- Gives a summary of the DataFrame, including the number of entries, column names, data types, and memory usage.

2.Missing Value & Outlier Detection

```
data.isnull().sum()

User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

- This data has no null values

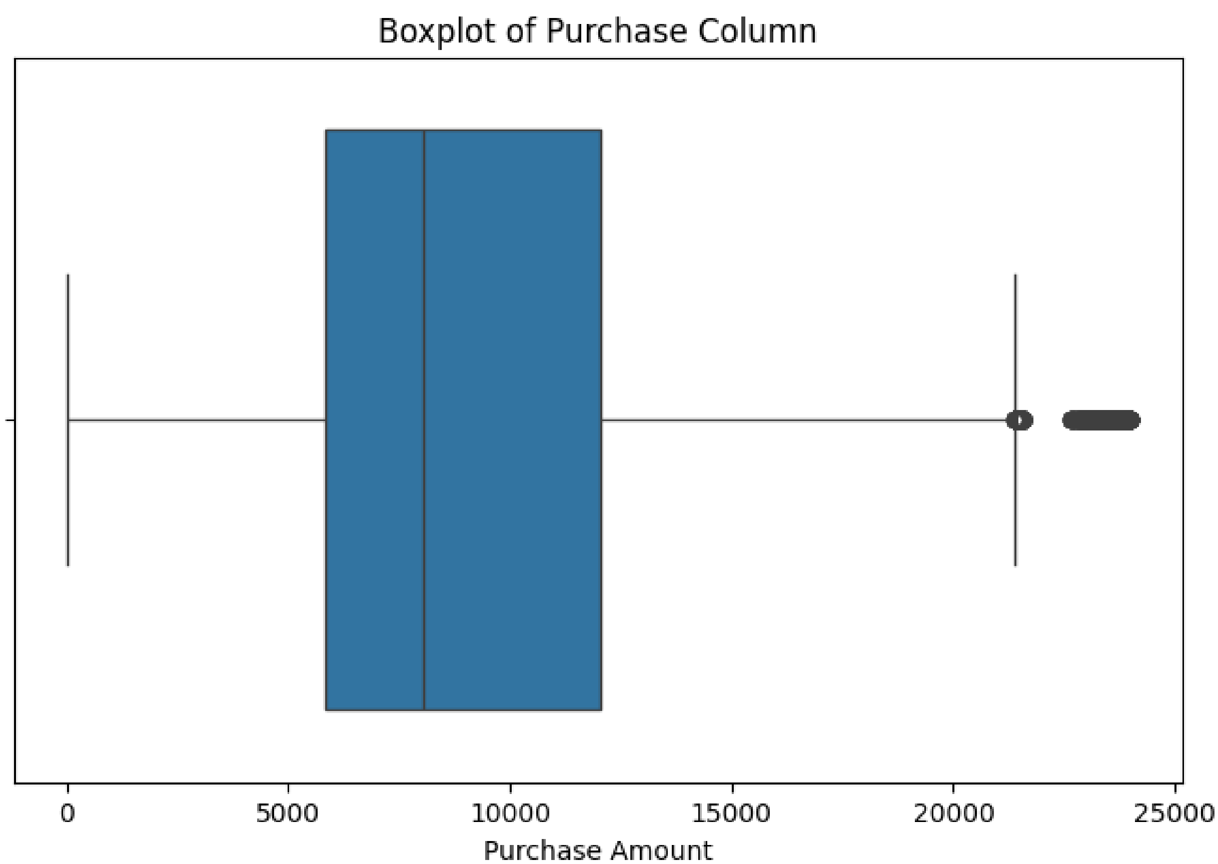
```
data.describe()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"User_ID\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 367117.89753373514,\n   \"min\": 1727.5915855306216,\n   \"max\": 1006040.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1003028.8424013031,\n   1003077.0,\n   550068.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Occupation\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 194475.49735336297,\n   \"min\": 0.0,\n   \"max\": 550068.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   8.076706879876669,\n   7.0,\n   550068.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Marital_Status\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 194478.25991330712,\n   \"min\": 0.0,\n   \"max\": 550068.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.40965298835780306,\n   1.0,\n   0.4917701263166973\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Product_Category\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\":

```
194476.16701795225,\n        \"min\": 1.0,\n        \"max\":
550068.0,\n        \"num_unique_values\": 7,\n        \"samples\": [\n
550068.0,\n          5.404270017525106,\n          8.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Purchase\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
191363.80903912007,\n        \"min\": 12.0,\n        \"max\":
550068.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
9263.968712959126,\n          8047.0,\n          550068.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


- Provides summary statistics for numerical columns in a DataFrame, such as count, mean, standard deviation, min, and max.

```
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['Purchase'])
plt.title("Boxplot of Purchase Column")
plt.xlabel("Purchase Amount")
plt.show()
```



- With this boxplot, we can see that there are outliers in purchase column.

```
data.head()
```

{"type":"dataframe","variable_name":"data"}

```
data.tail()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"User_ID\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 2,\n   \"min\": 1006033,\n   \"max\": 1006039,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   1006035,\n   1006039,\n   1006036\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Product_ID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"P00372445\",\n   \"P00375436\",\n   \"P00371644\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"F\",\n   \"M\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"26-35\",\n   \"46-50\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Occupation\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 7,\n   \"min\": 0,\n   \"max\": 15,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"City_Category\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"C\",\n   \"B\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Stay_In_Current_City_Years\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"3\",\n   \"2\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Marital_Status\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Product_Category\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 20,\n   \"max\": 20,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   20\n   ],\n   \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 128,\n        \"min\": 137,\n
\"max\": 490,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          371\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


- head(n) shows the first n rows, while tail(n) shows the last n rows of a DataFrame.

Data Exploration

```
female_customers = data[data['Gender']== 'F']
female_customers.shape
```

```
(135809, 10)
```

```
male_customers = data[data['Gender']== 'M']
male_customers.shape
```

```
(414259, 10)
```

```
female_mean = female_customers['Purchase'].mean()
male_mean = male_customers['Purchase'].mean()
```

```
female_mean
```

```
8734.565765155476
```

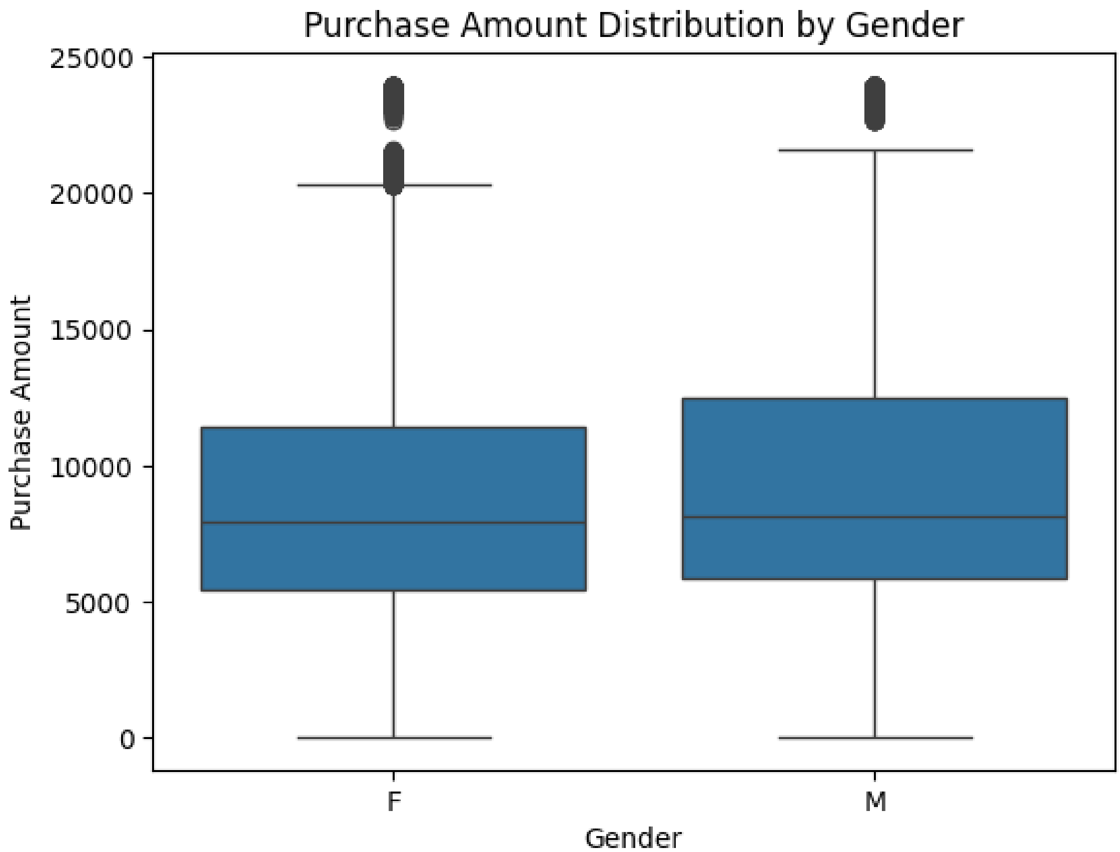
- The female average Purchase is 8734.56

```
male_mean
```

```
9437.526040472265
```

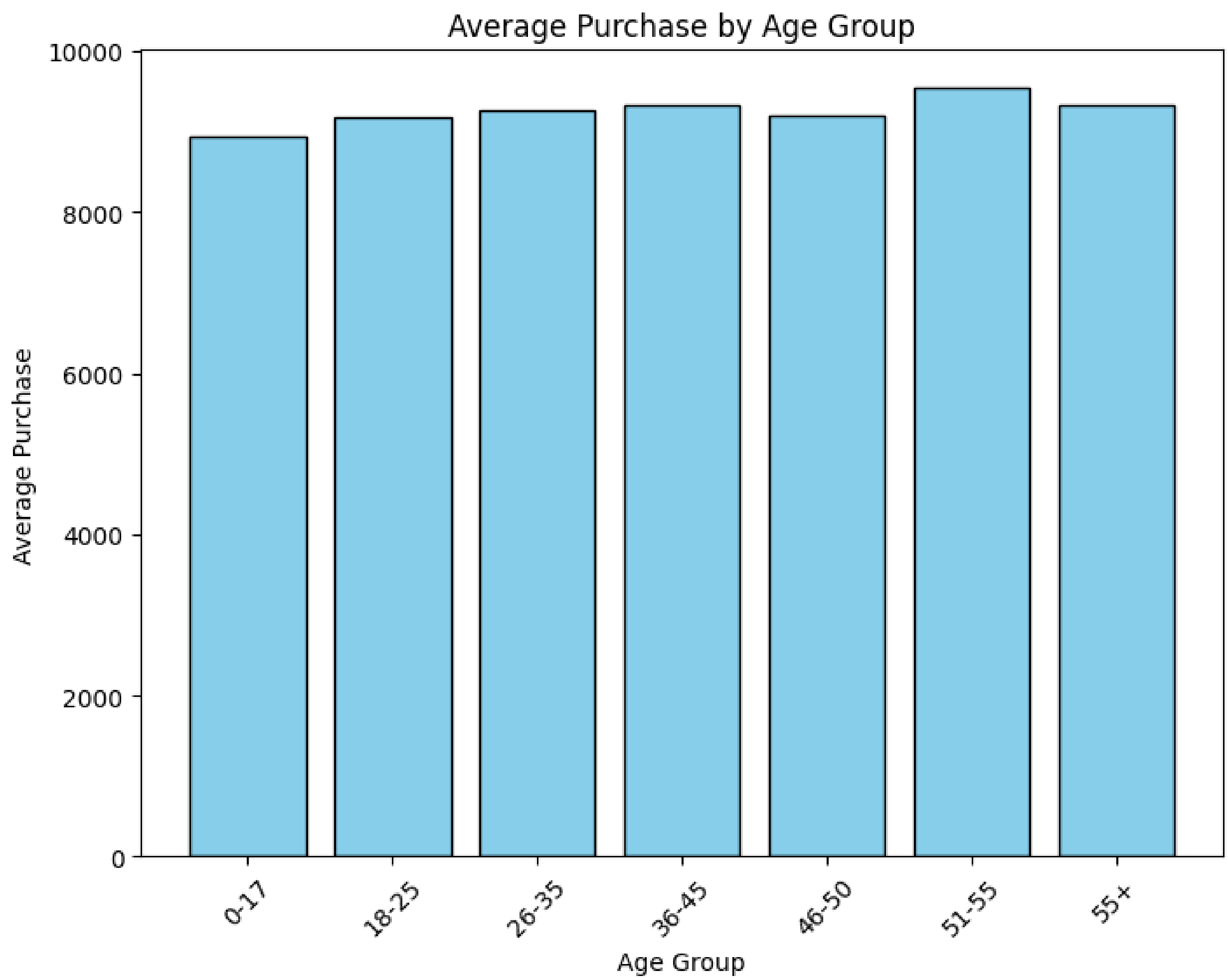
- The male average purchase is 9437.52

```
sns.boxplot(x='Gender', y='Purchase', data=data)
plt.title("Purchase Amount Distribution by Gender")
plt.xlabel("Gender")
plt.ylabel("Purchase Amount")
plt.show()
```



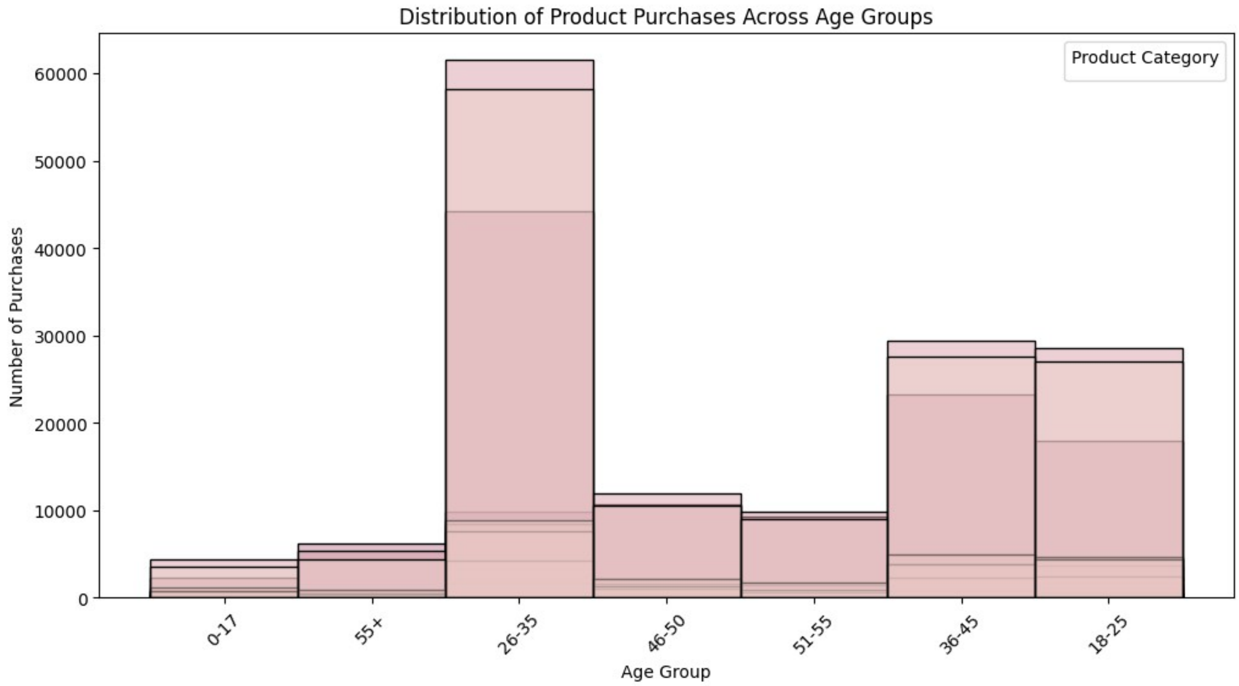
- Here, we can conclude that Men spend more than Women.
- The average purchase of males is higher than females.
- The Quartile values for male customers is slightly high than female customer
- The rows for female customers is 135809 whereas, for male it is 414259.
- This shows that transactions have been more under men customers.
- Quartile values being higher, more transcations gives more average to male customers

```
plt.figure(figsize=(8, 6))
plt.bar(age_wise_avg_purchase['Age'],
age_wise_avg_purchase['Purchase'], color='skyblue', edgecolor='black')
plt.xlabel('Age Group')
plt.ylabel('Average Purchase')
plt.title('Average Purchase by Age Group')
plt.xticks(rotation=45)
plt.show()
```



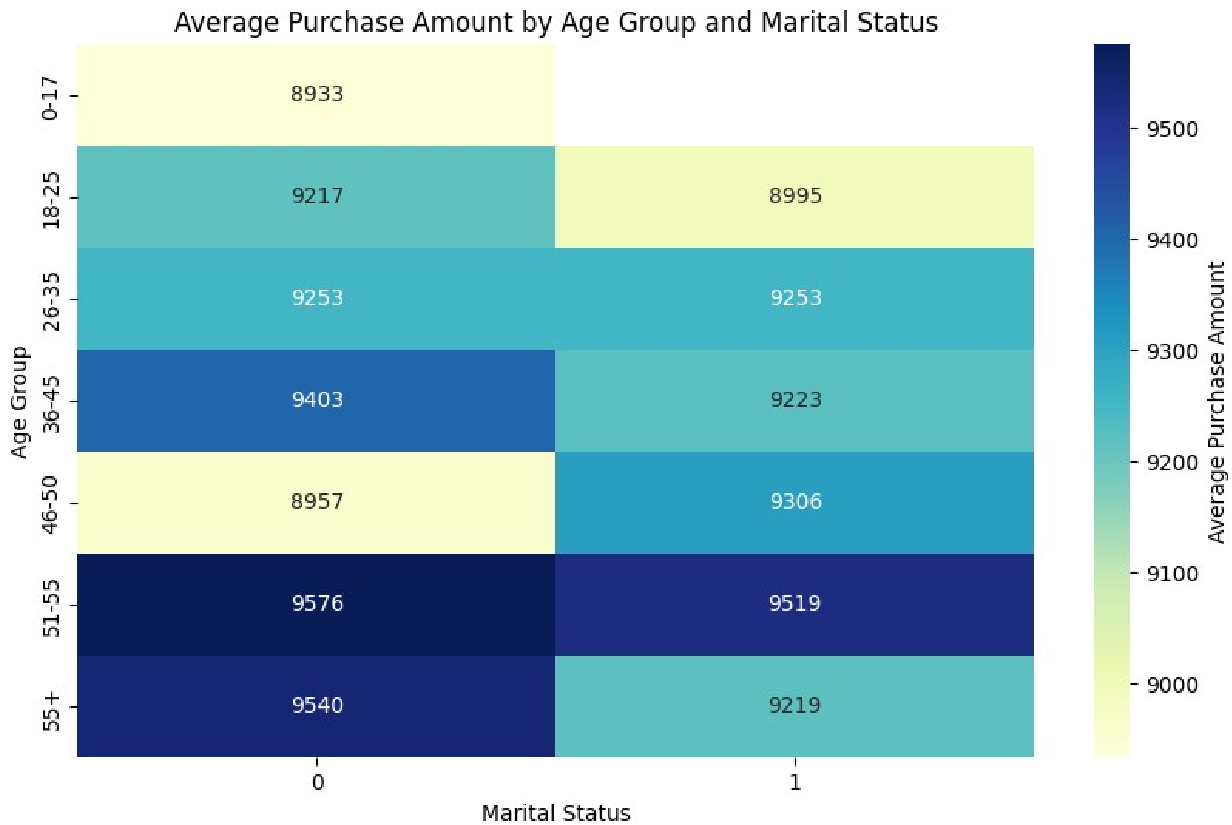
- This shows the age group of 51-55 has highest average purchase followed by 55+ age group.

```
#What products are different age groups buying?

plt.figure(figsize=(12, 6))
sns.histplot(data=data, x='Age', hue='Product_Category',  kde=False)

plt.xlabel('Age Group')
plt.ylabel('Number of Purchases')
plt.title('Distribution of Product Purchases Across Age Groups')
plt.legend(title='Product Category')
plt.xticks(rotation=45)
plt.show()

WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



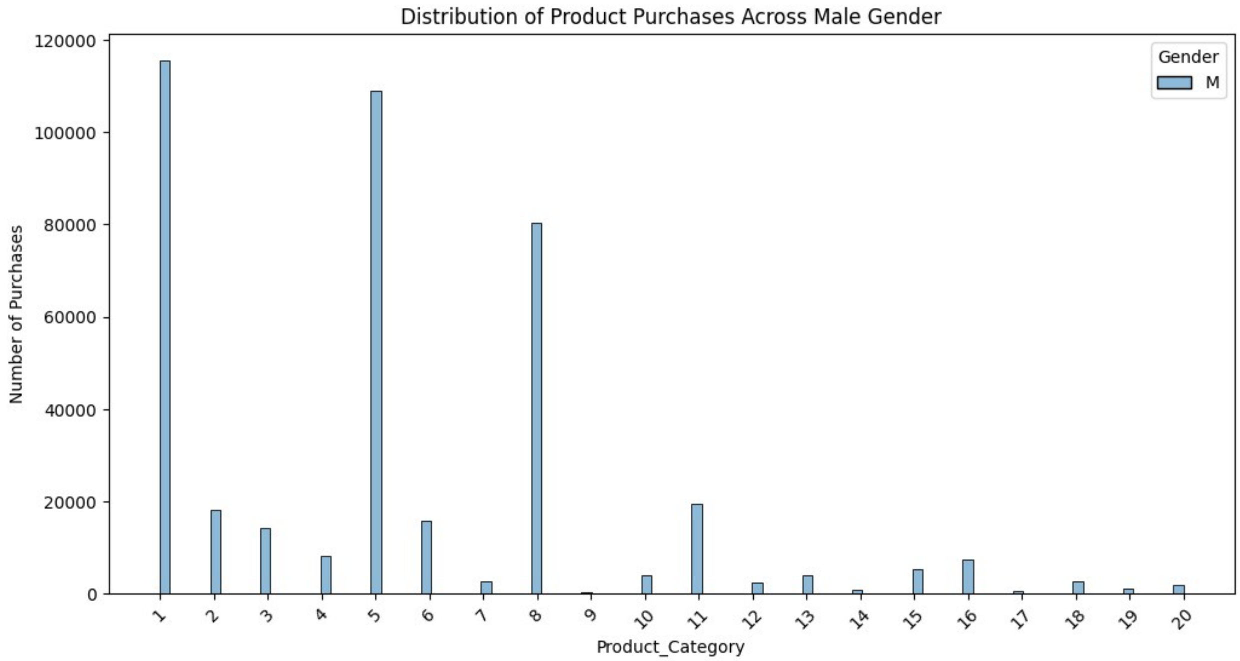
- This shows that 26-35 age group has more purchases with different product catergories.

```
pivot_table = data.pivot_table(values='Purchase', index='Age',
columns='Marital_Status', aggfunc='mean')
print(pivot_table)

Marital_Status            0            1
Age
0-17            8933.464640          NaN
18-25           9216.752419  8994.509992
26-35           9252.566484  9252.882410
36-45           9402.515329  9223.098451
46-50           8956.529551  9305.535821
51-55           9575.827475  9518.735088
55+             9539.774959  9218.510315

plt.figure(figsize=(10, 6))
sns.heatmap(pivot_table, annot=True, fmt=".0f", cmap="YlGnBu",
cbar_kws={'label': 'Average Purchase Amount'})
plt.title('Average Purchase Amount by Age Group and Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Age Group')
plt.show()
```



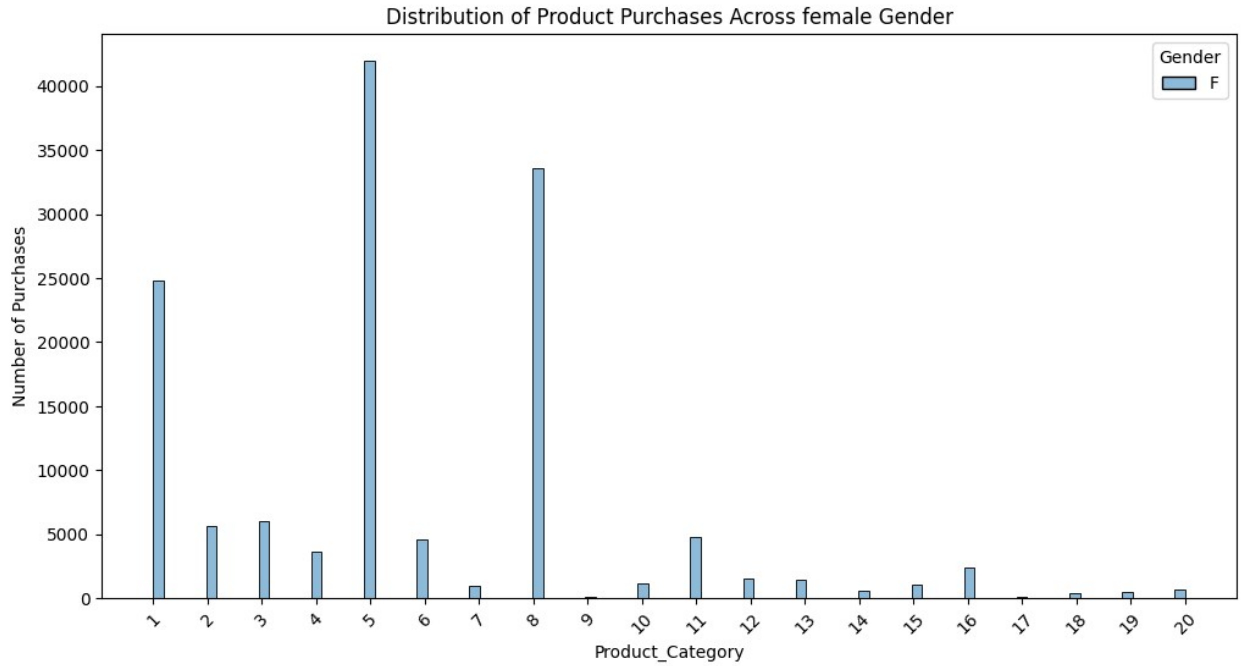
- Age group 51-55 and unmarried have high mean purchase.

```
plt.figure(figsize=(12, 6))
sns.histplot(data=male_customers, x='Product_Category', hue='Gender',
kde=False)

plt.xlabel('Product_Category')
plt.xticks(np.arange(1, 21, 1))
plt.ylabel('Number of Purchases')
plt.title('Distribution of Product Purchases Across Male Gender')

plt.xticks(rotation=45)
plt.show()
```



- Product Category 1 is most sold across male gender.

```
plt.figure(figsize=(12, 6))
sns.histplot(data=female_customers, x='Product_Category',
hue='Gender',  kde=False)

plt.xlabel('Product_Category')
plt.xticks(np.arange(1, 21, 1))
plt.ylabel('Number of Purchases')
plt.title('Distribution of Product Purchases Across female Gender')
plt.xticks(rotation=45)
plt.show()
```



- Product Category 5 is most sold across Female Gender.

4. How does gender affect the amount spent?

```
# Defining a function to compute the 95% CI using the CLT
def compute_clt_ci(data, gender):
    gender_data = data[data['Gender'] == gender]['Purchase']
    n = len(gender_data)
    mean = gender_data.mean()
    std_dev = gender_data.std()


    z_score = 1.96  # For 95% confidence
    margin_of_error = z_score * (std_dev / np.sqrt(n))
    lower_ci = mean - margin_of_error
    upper_ci = mean + margin_of_error

    return mean, (lower_ci, upper_ci)

#Defining a function to compute the 95% CI using Bootstrapping
def compute_bootstrap_ci(data, gender, n_iterations=1000):
    gender_data = data[data['Gender'] == gender]['Purchase']
    means = []

    for _ in range(n_iterations):
        sample = gender_data.sample(len(gender_data), replace=True)
        means.append(sample.mean())

    # Compute the 2.5th and 97.5th percentiles
    ci_lower = np.percentile(means, 2.5)
```

```
    ci_upper = np.percentile(means, 97.5)

    return np.mean(means), (ci_lower, ci_upper), means

all_means = {}
for gender in ['F', 'M']:
  clt_mean, clt_ci = compute_clt_ci(data, gender)
  bootstrap_mean, bootstrap_ci, means = compute_bootstrap_ci(data,
gender)
  all_means[f'{gender}_full'] = means
  print(f"\nGender: {gender}")
  print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
  print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")


Gender: F
CLT Mean: 8734.57, 95% CI: (8709.211081242413, 8759.920449068539)
Bootstrap Mean: 8734.21, 95% CI: (8707.200805174914,
8759.374582133732)

Gender: M
CLT Mean: 9437.53, 95% CI: (9422.019162420047, 9453.032918524483)
Bootstrap Mean: 9437.46, 95% CI: (9423.074905493906,
9453.401350000844)
```

- The width of the confidence interval depends on the variability (standard deviation) within each gender's data and the sample size.
- If one gender variability is high then the varaiability is wider.
- We can observe that Female customers variability is high showing presence of more outliers.

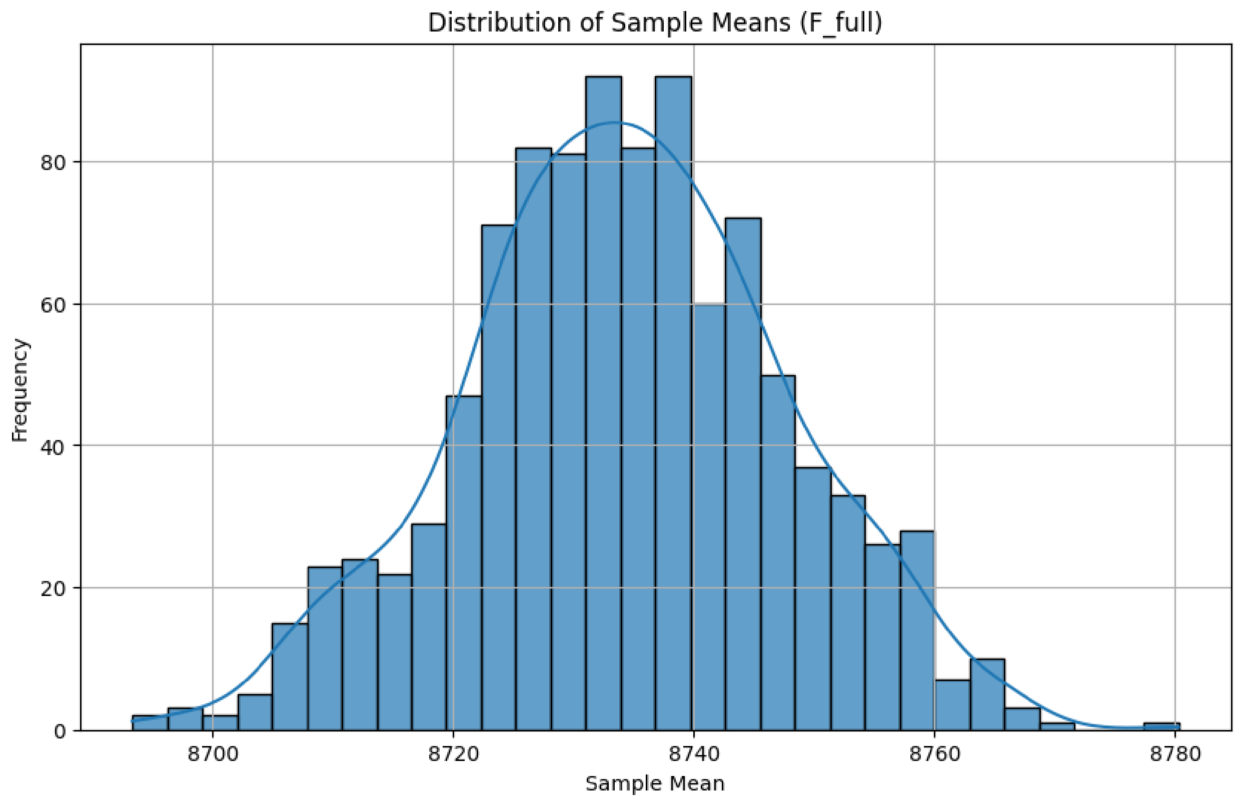
```
sample_sizes = [300, 3000, 30000]


for sample_size in sample_sizes:

  print(f"\nSample Size: {sample_size}")
  for gender in ['F', 'M']:
    sample_data = data[data['Gender'] ==
gender].sample(min(sample_size, len(data[data['Gender'] == gender])),
replace=False)
    clt_mean, clt_ci = compute_clt_ci(sample_data, gender)
    bootstrap_mean, bootstrap_ci, means =
compute_bootstrap_ci(sample_data, gender)
    all_means[f'{gender}_{sample_size}'] = means
    print(f"Gender: {gender}")
    print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
    print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")
```

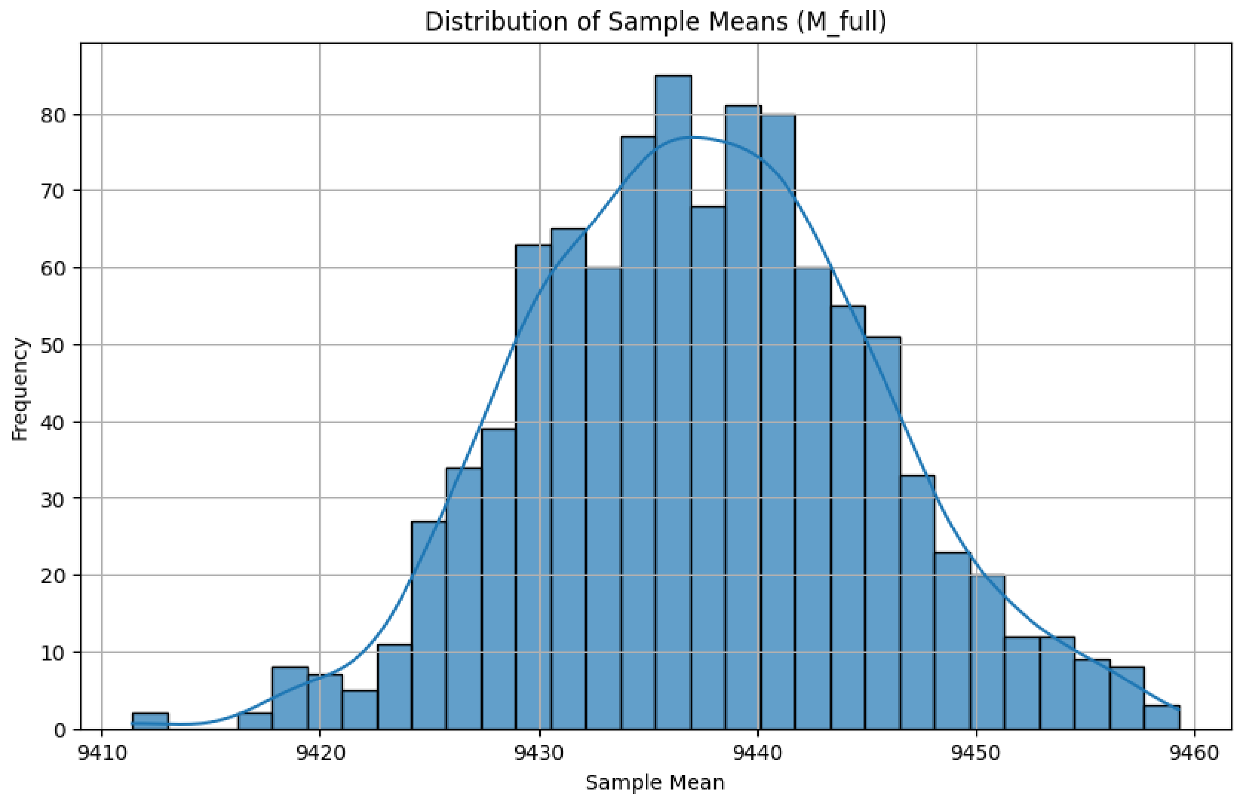
```
Sample Size: 300
Gender: F
CLT Mean: 9114.63, 95% CI: (8556.329607196933, 9672.923726136401)
Bootstrap Mean: 9110.20, 95% CI: (8548.393583333333,
9646.304666666667)
Gender: M
CLT Mean: 9504.73, 95% CI: (8930.343263441948, 10079.116736558051)
Bootstrap Mean: 9511.97, 95% CI: (8957.661, 10109.564416666666)

Sample Size: 3000
Gender: F
CLT Mean: 8724.01, 95% CI: (8550.946761315765, 8897.073238684236)
Bootstrap Mean: 8723.44, 95% CI: (8554.103650000001,
8896.173166666667)
Gender: M
CLT Mean: 9450.98, 95% CI: (9266.910410730161, 9635.050255936505)
Bootstrap Mean: 9451.51, 95% CI: (9251.544566666666, 9640.90445)

Sample Size: 30000
Gender: F
CLT Mean: 8736.11, 95% CI: (8682.287884957908, 8789.931381708759)
Bootstrap Mean: 8736.89, 95% CI: (8683.150668333334,
8791.857567500001)
Gender: M
CLT Mean: 9497.36, 95% CI: (9439.507215402635, 9555.2069179307)
Bootstrap Mean: 9497.13, 95% CI: (9435.874285, 9555.674176666667)
```

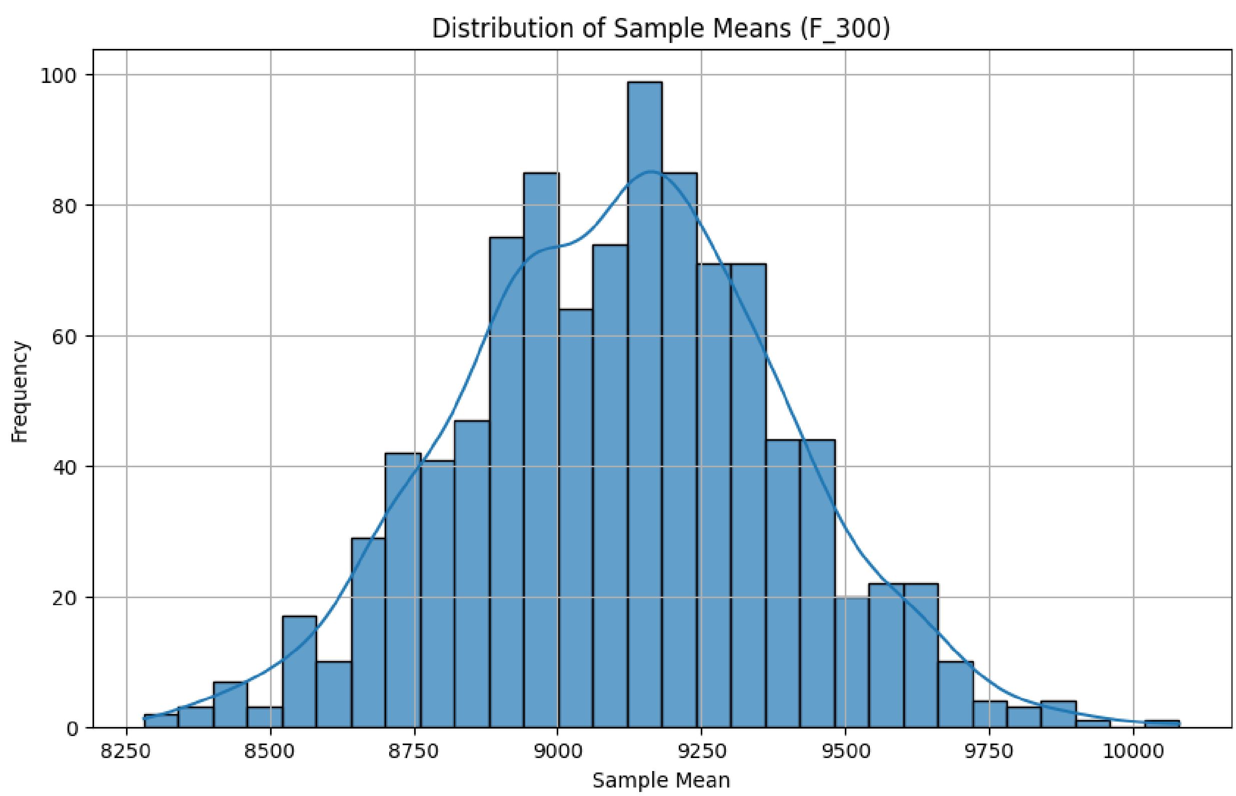
- For female customers, the interval for 300 sample size is 1,116 and for male is 1,149.
- For sample size of 3,000, the female customers interval is 347 and for male customers is 369.
- For sample size of 30,000, the female customers interval is 107 and for male customers is 116.
- With increase of the sample size, the interval decreased by 1/3.
- Larger sample sizes typically result in narrower confidence intervals because they provide more precise estimates of the population parameters.
- These results indicate that the confidence intervals for males and females do not overlap, suggesting a significant difference in the average amounts spent between the two genders.
- CLT vs. Bootstrapping: The intervals overlap, indicating consistent estimates.
- Male vs. Female: The intervals do not overlap, indicating significant differences between the groups.

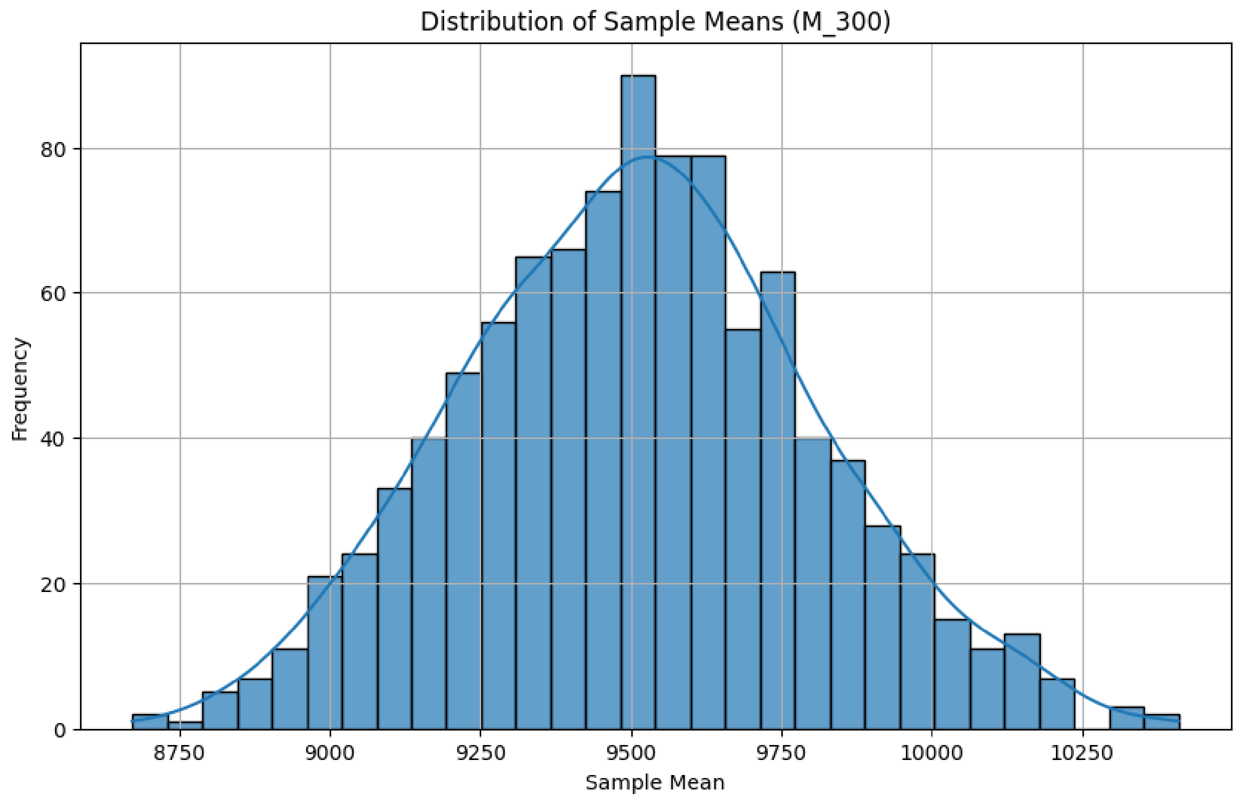
```
for key, means in all_means.items():
  plt.figure(figsize=(10, 6))
  sns.histplot(means, bins=30, kde=True, edgecolor='k', alpha=0.7)
```

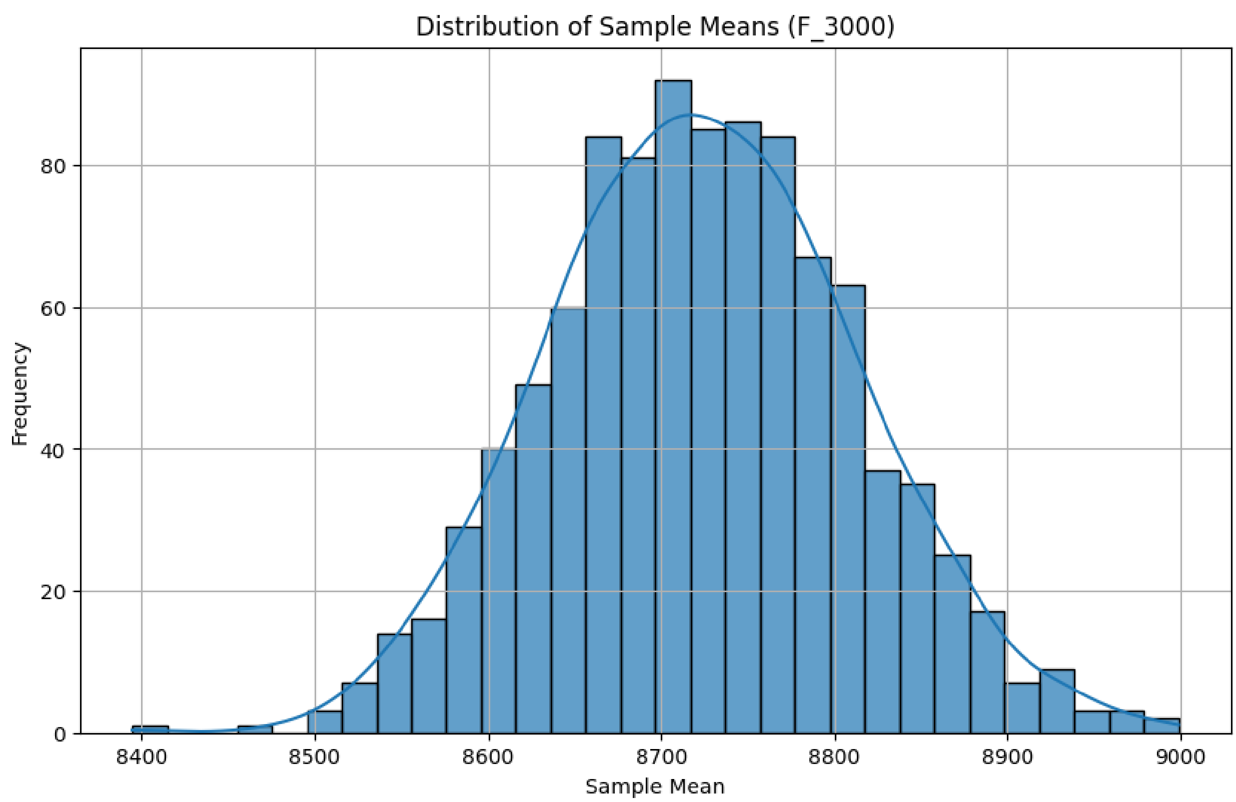
```
# Use seaborn's histplot with kde=True
plt.title(f'Distribution of Sample Means ({key})')
plt.xlabel('Sample Mean')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

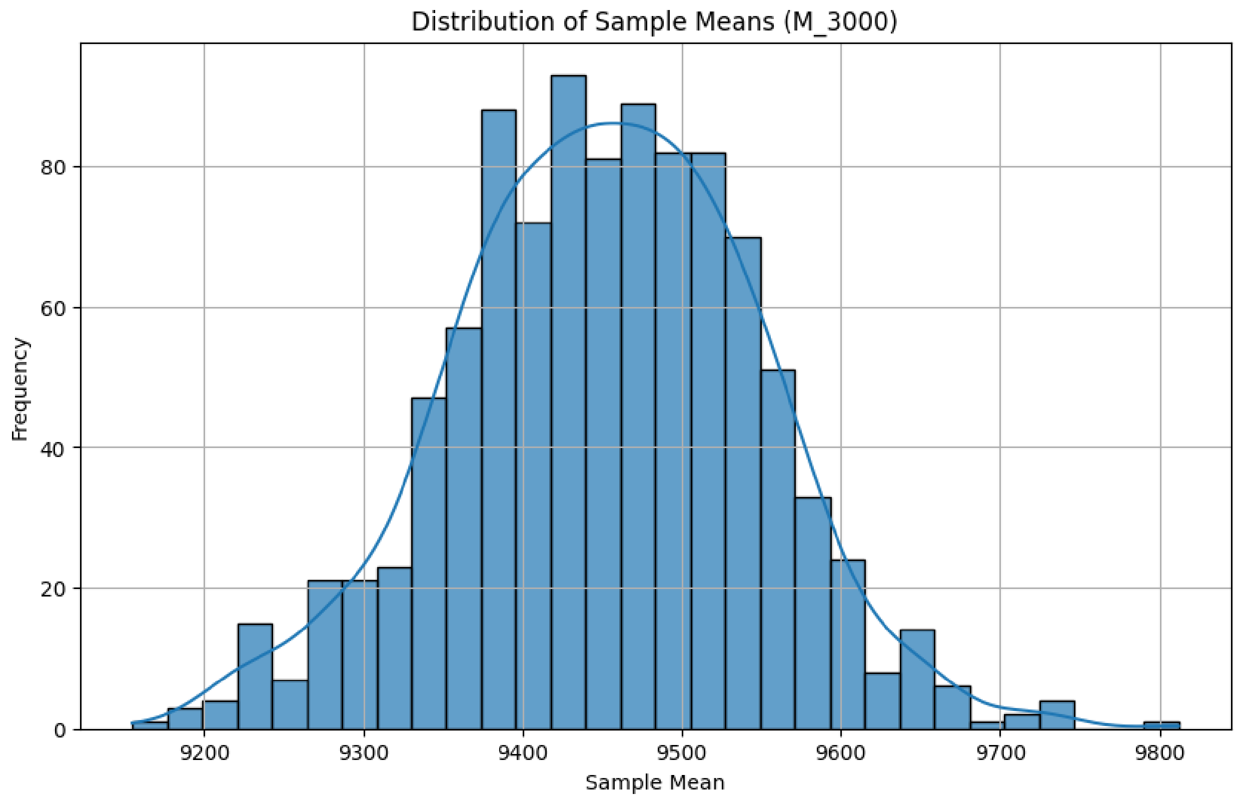


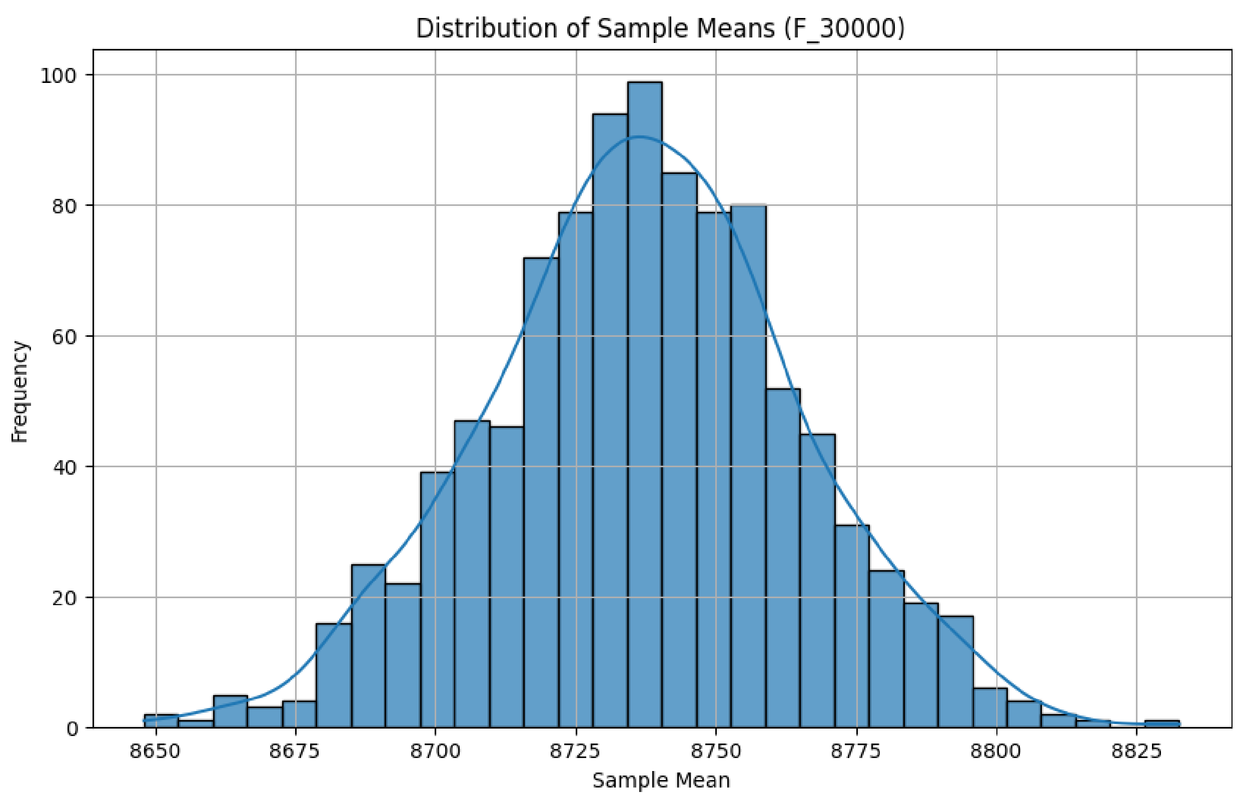

Distribution of Sample Means (M_full)
Frequency
Sample Mean
0
10
20
30
40
50
60
70
80
9410
9420
9430
9440
9450
9460

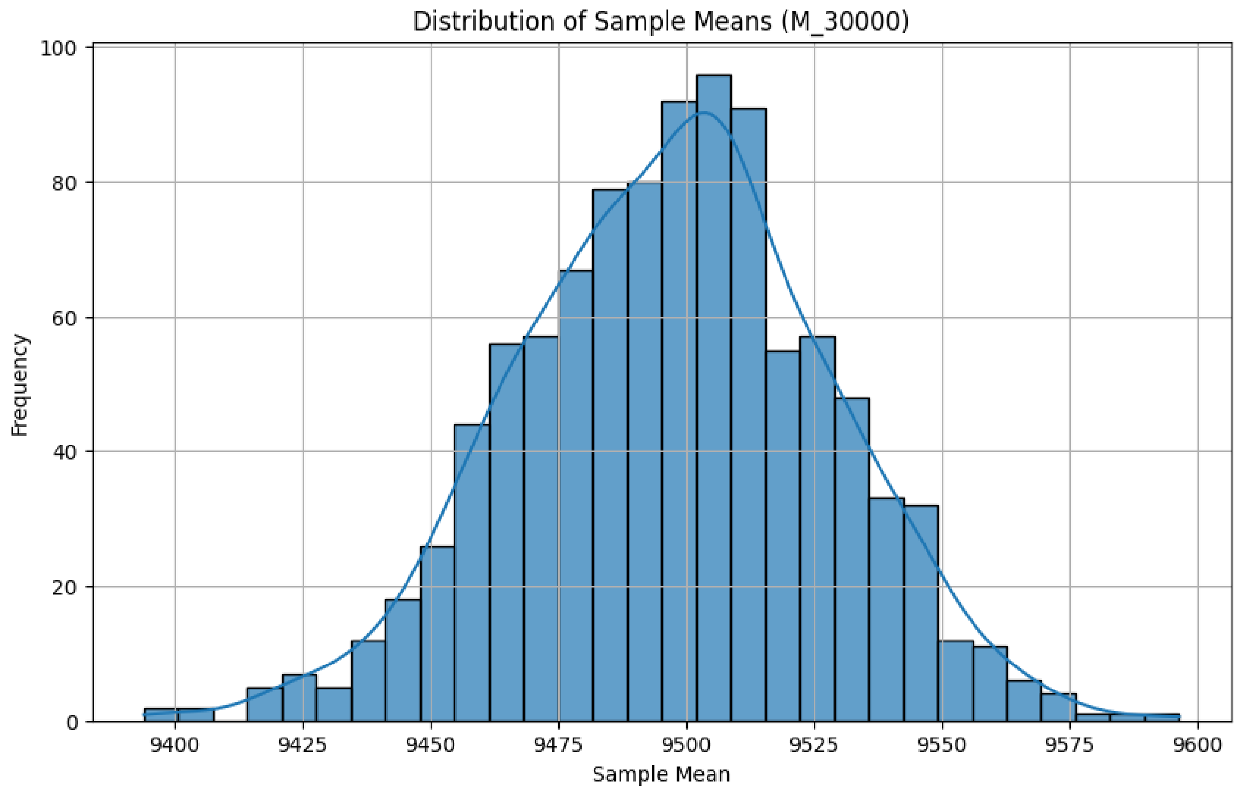

Distribution of Sample Means (F_300)
Frequency
Sample Mean
0
20
40
60
80
100
8250
8500
8750
9000
9250
9500
9750
10000


Distribution of Sample Means (M_300)
Frequency
Sample Mean
0
20
40
60
80
8750
9000
9250
9500
9750
10000
10250


Distribution of Sample Means (F_3000)
Frequency
Sample Mean
0
20
40
60
80
8400
8500
8600
8700
8800
8900
9000


Distribution of Sample Means (M_3000)
Frequency
Sample Mean
0
20
40
60
80
9200
9300
9400
9500
9600
9700
9800


Distribution of Sample Means (F_30000)
Frequency
Sample Mean
0
20
40
60
80
100
8650
8675
8700
8725
8750
8775
8800
8825



- As the sample size increases, the distribution of the sample means becomes more normal and tighter around the true mean, due to the Central Limit Theorem.
- With smaller sample sizes, the distribution may be more spread out and less normal, showing more variability.
- Larger sample sizes result in a distribution that is more peaked and centered around the true population mean.

5. How does Marital_Status affect the amount spent?

```
def compute_clt_ci2(data, Marital_Status):
    Marital_Status_data = data[data['Marital_Status'] ==
Marital_Status]['Purchase']
    n = len(Marital_Status_data)
    mean = Marital_Status_data.mean()
    std_dev = Marital_Status_data.std()


    z_score = 1.96  # For 95% confidence
    margin_of_error = z_score * (std_dev / np.sqrt(n))
    lower_ci = mean - margin_of_error
    upper_ci = mean + margin_of_error

    return mean, (lower_ci, upper_ci)

#Defining a function to compute the 95% CI using Bootstrapping
```

```
def compute_bootstrap_ci2(data,Marital_Status, n_iterations=1000):
   Marital_Status_data = data[data['Marital_Status'] ==
Marital_Status]['Purchase']
   means = []
   for _ in range(n_iterations):
        sample =Marital_Status_data.sample(len(Marital_Status_data),
replace=True)
        means.append(sample.mean())

   ci_lower = np.percentile(means, 2.5)
   ci_upper = np.percentile(means, 97.5)
   return np.mean(means), (ci_lower, ci_upper), means

all_means2 = {}
for Marital_Status in [ 0, 1]:
  clt_mean, clt_ci = compute_clt_ci2(data, Marital_Status)
  bootstrap_mean, bootstrap_ci, means = compute_bootstrap_ci2(data,
Marital_Status)
  all_means2[f'{Marital_Status}_full'] = means
  print(f"\nMarital_Status: {Marital_Status}")
  print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
  print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")



Marital_Status: 0
CLT Mean: 9265.91, 95% CI: (9248.61610045097, 9283.199137392043)
Bootstrap Mean: 9266.25, 95% CI: (9250.026530805499,
9282.076558058825)

Marital_Status: 1
CLT Mean: 9261.17, 95% CI: (9240.460046422771, 9281.889101741976)
Bootstrap Mean: 9260.91, 95% CI: (9241.125505132313,
9281.792060225353)
```

- The width of the confidence interval depends on the variability within each data and the sample size.
- If variability is high then the varaiability is wider.
- Compared to unarried purchases, unmarried has more variablity.
- The overlap between CLT and Bootstrap confidence intervals indicates that both methods provide similar, consistent results.
- Larger sample sizes result in narrower confidence intervals, offering more precise estimates.
- There is minimal overlap between the confidence intervals of different marital statuses, suggesting significant differences between the two groups.

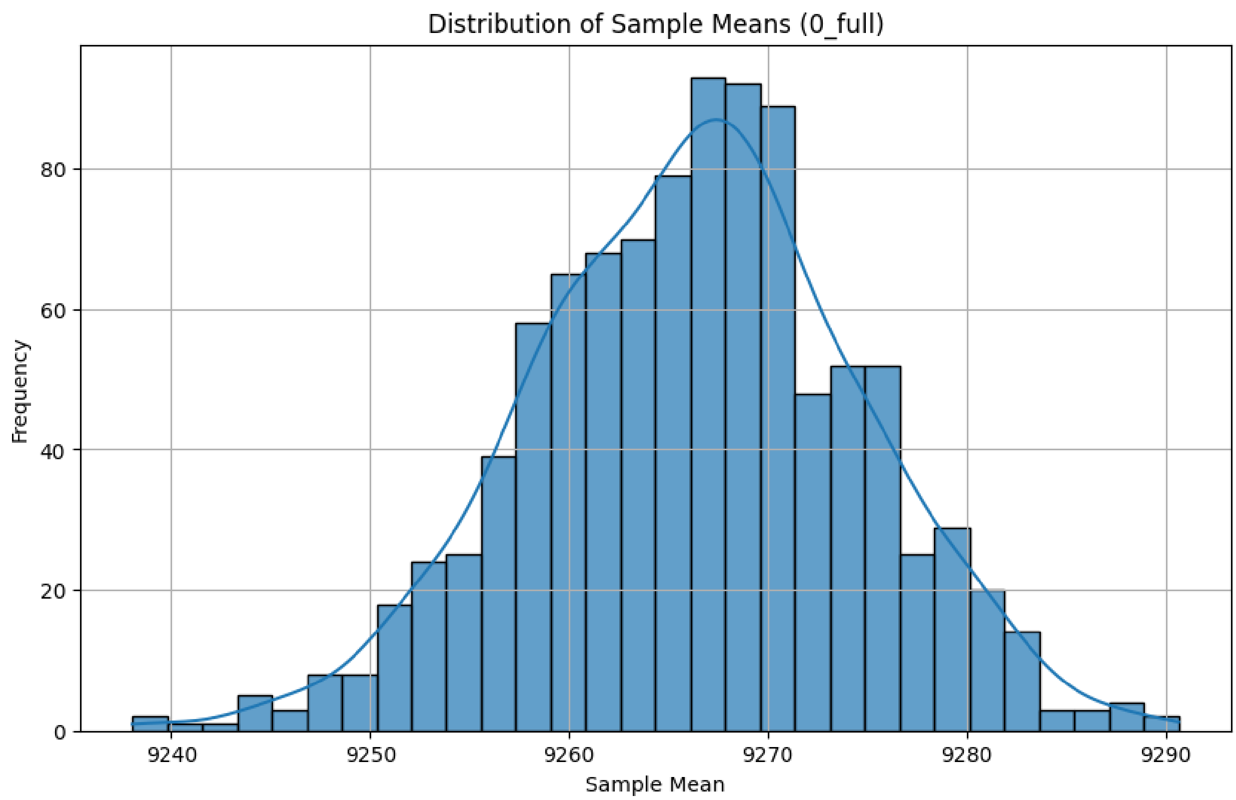
```
sample_sizes = [300, 3000, 30000]
```

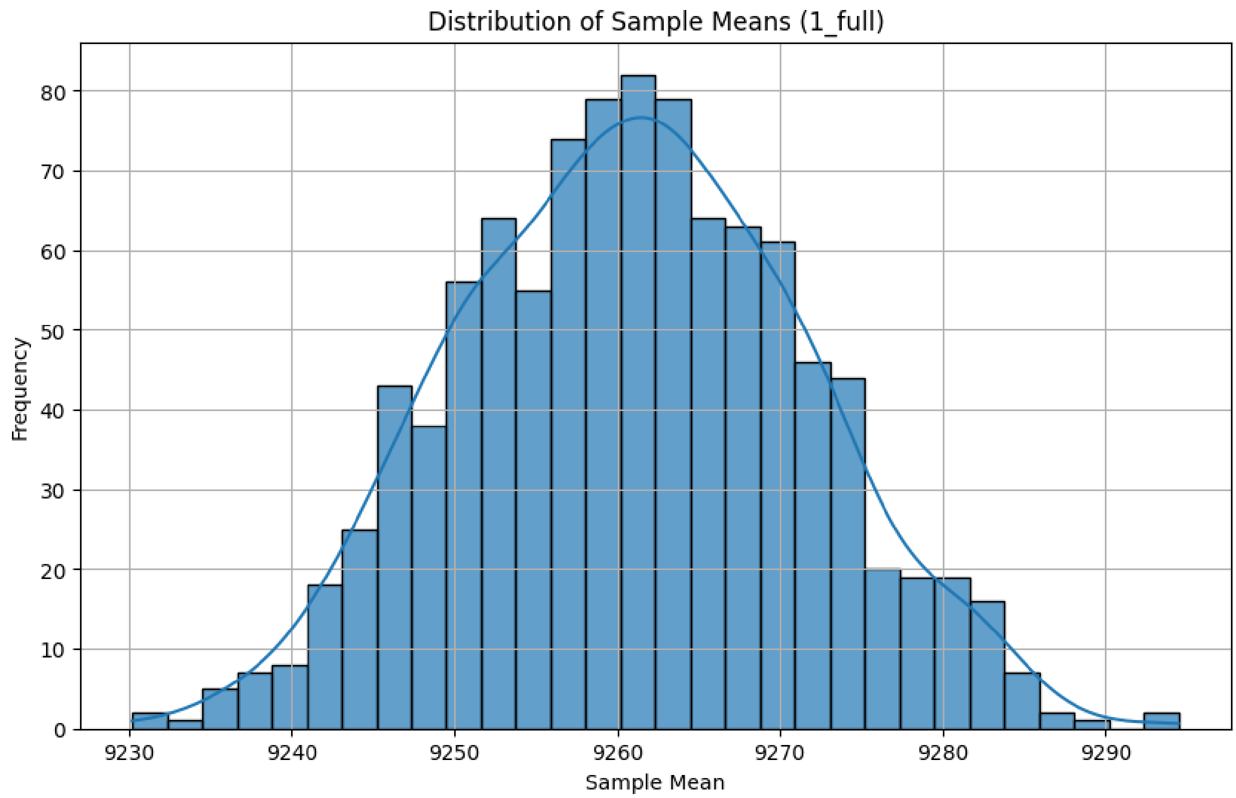
```
for sample_size in sample_sizes:

  print(f"\nSample Size: {sample_size}")
  for Marital_Status in [0, 1]:
    sample_data = data[data['Marital_Status'] ==
Marital_Status].sample(min(sample_size,
len(data[data['Marital_Status'] == Marital_Status])), replace=False)
    clt_mean, clt_ci = compute_clt_ci2(sample_data, Marital_Status)
    bootstrap_mean, bootstrap_ci, means =
compute_bootstrap_ci2(sample_data, Marital_Status)
    all_means2[f'{Marital_Status}_{sample_size}'] = means
    print(f"Marital_Status: {Marital_Status}")
    print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
    print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")


Sample Size: 300
Marital_Status: 0
CLT Mean: 9624.33, 95% CI: (9066.916631104905, 10181.736702228427)
Bootstrap Mean: 9625.87, 95% CI: (9068.813, 10182.615333333333)
Marital_Status: 1
CLT Mean: 9192.01, 95% CI: (8637.601292624784, 9746.412040708548)
Bootstrap Mean: 9191.21, 95% CI: (8652.075416666667,
9727.761750000001)

Sample Size: 3000
Marital_Status: 0
CLT Mean: 9199.88, 95% CI: (9019.034954995523, 9380.727711671143)
Bootstrap Mean: 9199.26, 95% CI: (9015.636583333335,
9377.205183333332)
Marital_Status: 1
CLT Mean: 9301.97, 95% CI: (9123.136583813735, 9480.799416186266)
Bootstrap Mean: 9301.13, 95% CI: (9121.122316666666, 9470.012775)

Sample Size: 30000
Marital_Status: 0
CLT Mean: 9281.78, 95% CI: (9224.935231731763, 9338.619101601571)
Bootstrap Mean: 9282.69, 95% CI: (9226.6158, 9342.220125833333)
Marital_Status: 1
CLT Mean: 9244.00, 95% CI: (9187.186390741956, 9300.815009258045)
Bootstrap Mean: 9244.55, 95% CI: (9187.280818333333,
9303.142164166667)
```

- For sample size of 300, the Unmarried customers interval is 1,114 and for married customers is 1,113.
- For sample size of 3,000, the Unmarried customers interval is 361 and for married customers is 357.
- For sample size of 30,000, the Unmarried customers interval is 113 and for married customers is 113.

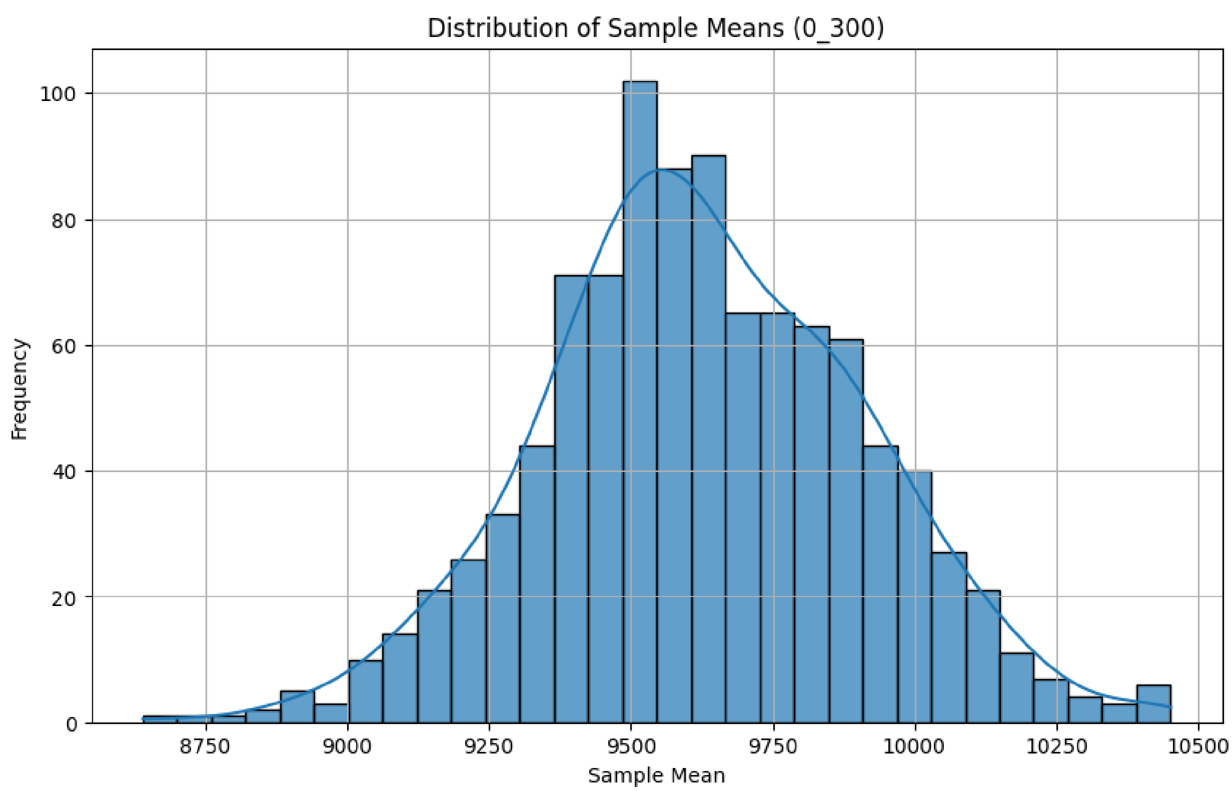
- With increase of the sample size, the interval decreased by 1/3.
- Larger sample sizes typically result in narrower confidence intervals because they provide more precise estimates of the population parameters.

```
for key, means in all_means2.items():
  plt.figure(figsize=(10, 6))
  sns.histplot(means, bins=30, kde=True, edgecolor='k', alpha=0.7)
  # Use seaborn's histplot with kde=True
  plt.title(f'Distribution of Sample Means ({key})')
  plt.xlabel('Sample Mean')
  plt.ylabel('Frequency')
  plt.grid(True)
  plt.show()
```

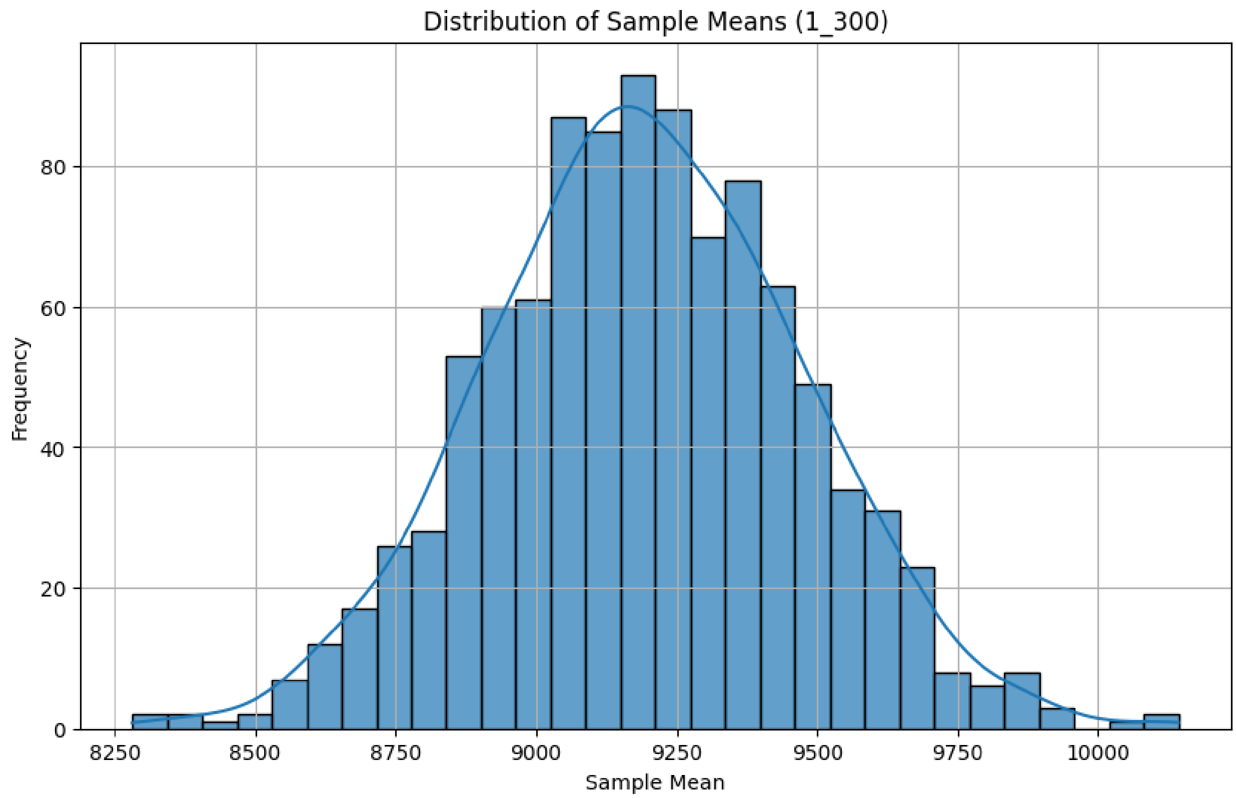


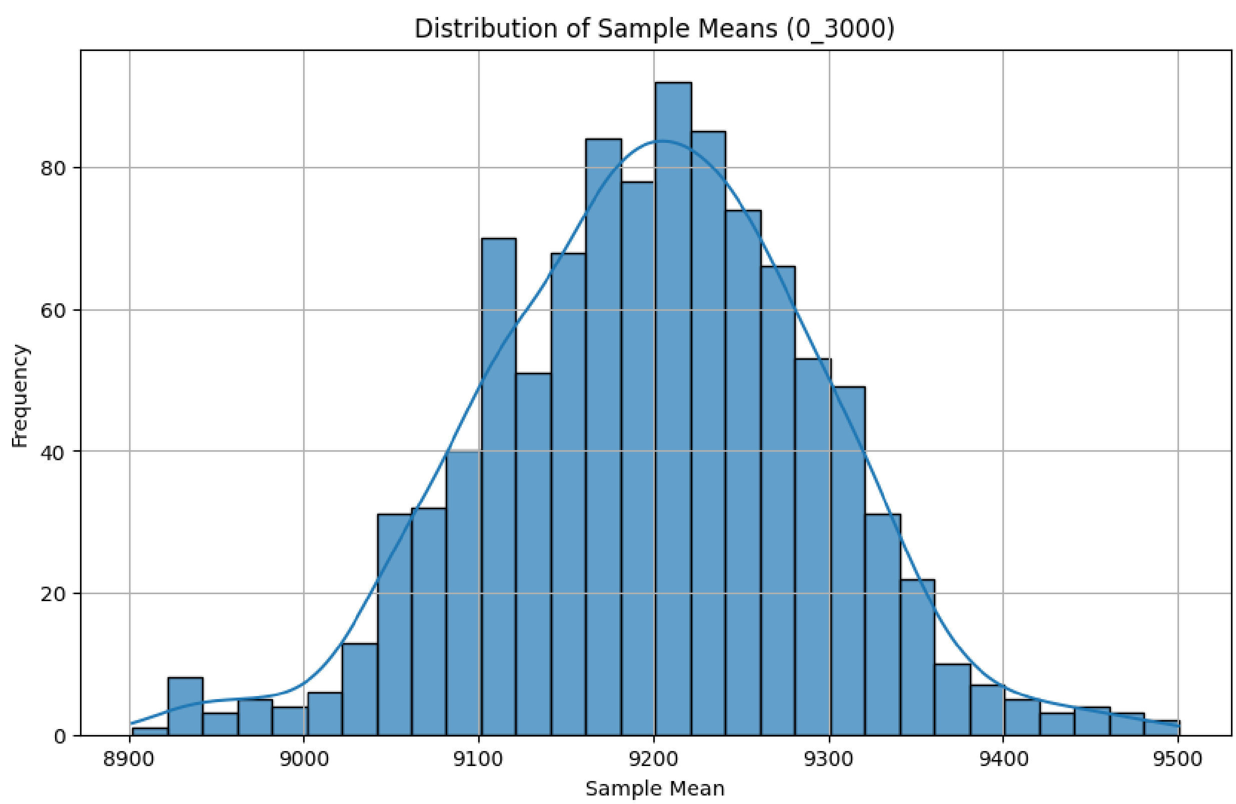


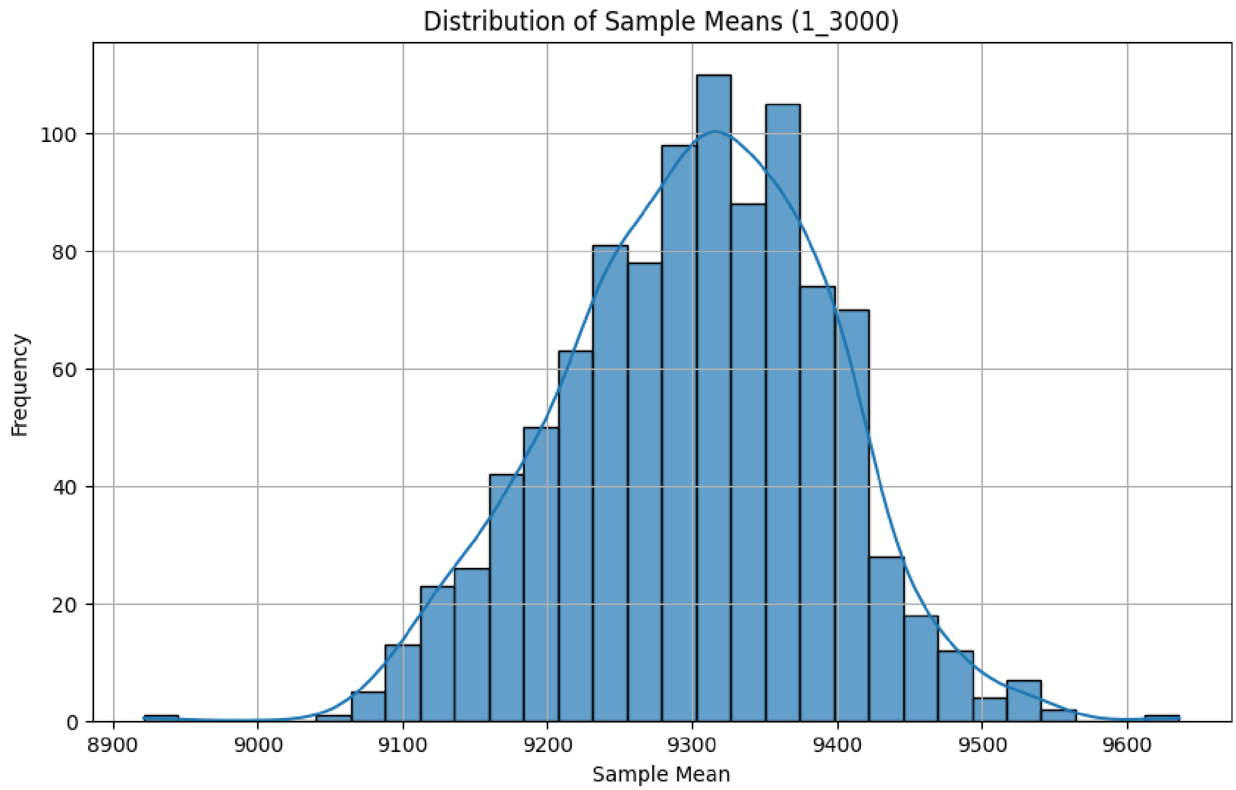
Distribution of Sample Means (1_full)
Frequency
Sample Mean
0
10
20
30
40
50
60
70
80
9230
9240
9250
9260
9270
9280
9290

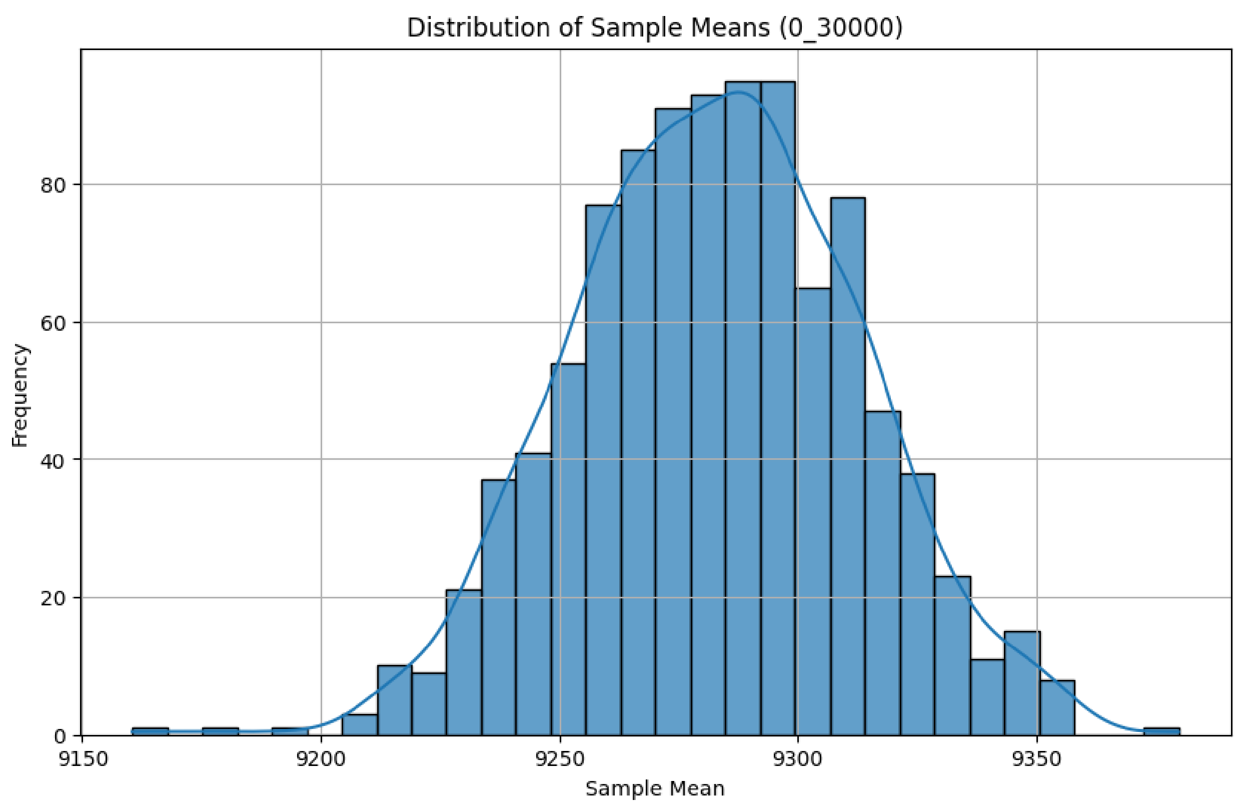


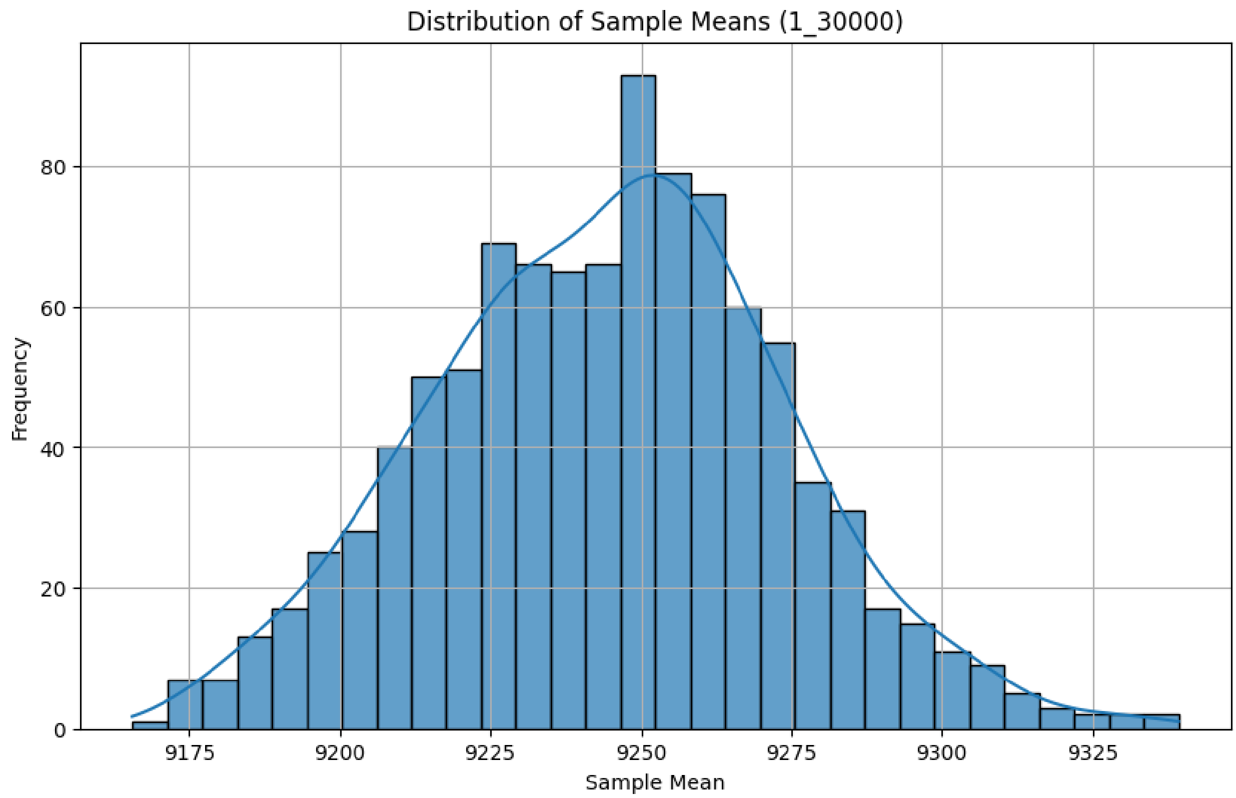
Distribution of Sample Means (0_300)
Frequency
Sample Mean
0
20
40
60
80
100
8750
9000
9250
9500
9750
10000
10250
10500


Distribution of Sample Means (1_300)
Frequency
Sample Mean
0
20
40
60
80
8250
8500
8750
9000
9250
9500
9750
10000


Distribution of Sample Means (0_3000)
Frequency
Sample Mean
0
20
40
60
80
8900
9000
9100
9200
9300
9400
9500


Distribution of Sample Means (1_3000)
Frequency
Sample Mean
0
20
40
60
80
100
8900
9000
9100
9200
9300
9400
9500
9600


Distribution of Sample Means (0_30000)
Frequency
Sample Mean
0
20
40
60
80
9150
9200
9250
9300
9350



- As you increase the data, the distribution of sample means starts to resemble a normal distribution and comes closer to the true mean,due to the Central Limit Theorem.
- When dealing with smaller datasets, the distribution tends to be more dispersed and less normal, indicating higher variability.
- Increasing the sample size leads to a distribution that is more concentrated and centered around the actual population mean.

6. How does Age affect the amount spent?

```
def compute_clt_ci(data, age_group):
    age_data = data[data['Age'] == age_group]['Purchase']
    n = len(age_data)
    mean = age_data.mean()
    std_dev = age_data.std()

    z_score = 1.96  # For 95% confidence
    margin_of_error = z_score * (std_dev / np.sqrt(n))
    ci_lower = mean - margin_of_error
    ci_upper = mean + margin_of_error

    return mean, (ci_lower, ci_upper)
```

```
def compute_bootstrap_ci(data, age_group, n_iterations=1000):
    age_data = data[data['Age'] == age_group]['Purchase']
    means = []


    for _ in range(n_iterations):
        sample = age_data.sample(len(age_data), replace=True)
        means.append(sample.mean())


    ci_lower = np.percentile(means, 2.5)
    ci_upper = np.percentile(means, 97.5)

    return np.mean(means), (ci_lower, ci_upper), means

all_means = {}
age_groups = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55',
'55+']

for age_group in age_groups:
    clt_mean, clt_ci = compute_clt_ci(data, age_group)
    bootstrap_mean, bootstrap_ci, means = compute_bootstrap_ci(data,
age_group)
    all_means[f'{age_group}_full'] = means
    print(f"\nAge Group: {age_group}")
    print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
    print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")
```

```
Age Group: 0-17
CLT Mean: 8933.46, 95% CI: (8851.946472627222, 9014.982808262726)
Bootstrap Mean: 8933.03, 95% CI: (8848.731497483777,
9016.344067010992)

Age Group: 18-25
CLT Mean: 9169.66, 95% CI: (9138.407374412845, 9200.919838109732)
Bootstrap Mean: 9170.03, 95% CI: (9140.009782259684, 9201.57581050572)

Age Group: 26-35
CLT Mean: 9252.69, 95% CI: (9231.73329130396, 9273.647974435815)
Bootstrap Mean: 9252.62, 95% CI: (9232.486668154308,
9273.083690063619)

Age Group: 36-45
CLT Mean: 9331.35, 95% CI: (9301.668865554733, 9361.032524281014)
Bootstrap Mean: 9331.70, 95% CI: (9300.39451019425, 9362.697308954395)

Age Group: 46-50
CLT Mean: 9208.63, 95% CI: (9163.084305814993, 9254.167089121662)
Bootstrap Mean: 9209.45, 95% CI: (9164.116691647885,
```

```
9253.476763090523)

Age Group: 51-55
CLT Mean: 9534.81, 95% CI: (9483.990538993237, 9585.625522927234)
Bootstrap Mean: 9534.44, 95% CI: (9484.224839614555, 9587.75845042986)

Age Group: 55+
CLT Mean: 9336.28, 95% CI: (9269.297603592036, 9403.263315306773)
Bootstrap Mean: 9334.56, 95% CI: (9270.800499906994,
9400.452183314732)
```

- The age group 0-17 has the widest confidence interval. This indicates more variability in the amount spent by individuals in this age group compared to others.

```
sample_sizes = [300, 3000, 30000]

for sample_size in sample_sizes:
    print(f"\nSample Size: {sample_size}")
    for age_group in age_groups:
        age_data = data[data['Age'] == age_group]
        sample_size = min(sample_size, len(age_data))
        sample_data = age_data.sample(sample_size, replace=False)

        clt_mean, clt_ci = compute_clt_ci(sample_data, age_group)
        bootstrap_mean, bootstrap_ci, means =
compute_bootstrap_ci(sample_data, age_group)
        all_means[f'{age_group}_{sample_size}'] = means

        print(f"Age Group: {age_group}")
        print(f"CLT Mean: {clt_mean:.2f}, 95% CI: {clt_ci}")
        print(f"Bootstrap Mean: {bootstrap_mean:.2f}, 95% CI:
{bootstrap_ci}")


Sample Size: 300
Age Group: 0-17
CLT Mean: 9072.75, 95% CI: (8484.407810866182, 9661.08552246715)
Bootstrap Mean: 9063.79, 95% CI: (8497.798999999999, 9625.33125)
Age Group: 18-25
CLT Mean: 9742.82, 95% CI: (9115.85552755069, 10369.791139115978)
Bootstrap Mean: 9730.94, 95% CI: (9108.227083333333,
10367.619999999999)
Age Group: 26-35
CLT Mean: 9180.90, 95% CI: (8629.256369784784, 9732.536963548551)
Bootstrap Mean: 9176.72, 95% CI: (8645.056666666667,
9717.123416666665)
Age Group: 36-45
CLT Mean: 9122.96, 95% CI: (8584.212436001557, 9661.700897331777)
Bootstrap Mean: 9131.36, 95% CI: (8602.338833333333,
9639.077916666665)
```

```
Age Group: 46-50
CLT Mean: 9552.58, 95% CI: (9006.94342826669, 10098.209905066642)
Bootstrap Mean: 9545.43, 95% CI: (8978.458, 10108.701083333333)
Age Group: 51-55
CLT Mean: 9865.43, 95% CI: (9298.203503133891, 10432.65649686611)
Bootstrap Mean: 9857.68, 95% CI: (9270.00825, 10449.984833333334)
Age Group: 55+
CLT Mean: 9304.53, 95% CI: (8754.840561317815, 9854.21943868218 7)
Bootstrap Mean: 9296.99, 95% CI: (8776.453833333333, 9823.791)

Sample Size: 3000
Age Group: 0-17
CLT Mean: 8870.33, 95% CI: (8684.828665491075, 9055.821334508926)
Bootstrap Mean: 8868.25, 95% CI: (8688.706450000001, 9056.626225)
Age Group: 18-25
CLT Mean: 9133.06, 95% CI: (8954.08001774569, 9312.033315587645)
Bootstrap Mean: 9135.63, 95% CI: (8968.786133333333,
9320.741816666667)
Age Group: 26-35
CLT Mean: 9126.04, 95% CI: (8946.169953191687, 9305.917380141645)
Bootstrap Mean: 9124.05, 95% CI: (8935.368124999999,
9311.611200000001)
Age Group: 36-45
CLT Mean: 9446.79, 95% CI: (9264.609744349938, 9628.970255650063)
Bootstrap Mean: 9446.64, 95% CI: (9266.630966666668, 9636.81615)
Age Group: 46-50
CLT Mean: 9274.28, 95% CI: (9096.511561007123, 9452.045105659545)
Bootstrap Mean: 9270.15, 95% CI: (9090.05625, 9450.603366666666)
Age Group: 51-55
CLT Mean: 9680.92, 95% CI: (9498.849860960598, 9862.991472372736)
Bootstrap Mean: 9680.01, 95% CI: (9499.651808333334,
9862.011358333333)
Age Group: 55+
CLT Mean: 9444.43, 95% CI: (9263.826327984625, 9625.025005348707)
Bootstrap Mean: 9445.95, 95% CI: (9250.8538, 9623.843175)

Sample Size: 30000
Age Group: 0-17
CLT Mean: 8933.46, 95% CI: (8851.946472627222, 9014.982808262726)
Bootstrap Mean: 8932.15, 95% CI: (8850.004160045028,
9010.631245861476)
Age Group: 18-25
CLT Mean: 9158.49, 95% CI: (9078.2082621631, 9238.76220532465)
Bootstrap Mean: 9158.58, 95% CI: (9078.229204741094,
9234.681792146735)
Age Group: 26-35
CLT Mean: 9170.56, 95% CI: (9091.13333319582, 9249.99247795502)
Bootstrap Mean: 9167.28, 95% CI: (9093.176337571182,
9246.113451860681)
```

```
Age Group: 36-45
CLT Mean: 9315.85, 95% CI: (9235.215574623127, 9396.477975899981)
Bootstrap Mean: 9316.07, 95% CI: (9233.784124619255,
9399.905146669316)
Age Group: 46-50
CLT Mean: 9191.51, 95% CI: (9112.240618283873, 9270.782689887232)
Bootstrap Mean: 9190.42, 95% CI: (9116.272184147796,
9271.527506290557)
Age Group: 51-55
CLT Mean: 9572.47, 95% CI: (9490.698415772338, 9654.233911071788)
Bootstrap Mean: 9572.37, 95% CI: (9490.14313170441, 9654.110334723877)
Age Group: 55+
CLT Mean: 9349.87, 95% CI: (9269.910647608918, 9429.83071115151)
Bootstrap Mean: 9349.66, 95% CI: (9270.215029466297,
9428.399018341941)
```

Sample Size: 300

- Widest CLT CI: Age Group 18-25, with a range of 1253.93.
- Widest Bootstrap CI: Age Group 18-25, with a range of 1259.39.
- The age group 18-25 has the widest confidence interval. This indicates more variability in the amount spent by individuals in this age group compared to others.

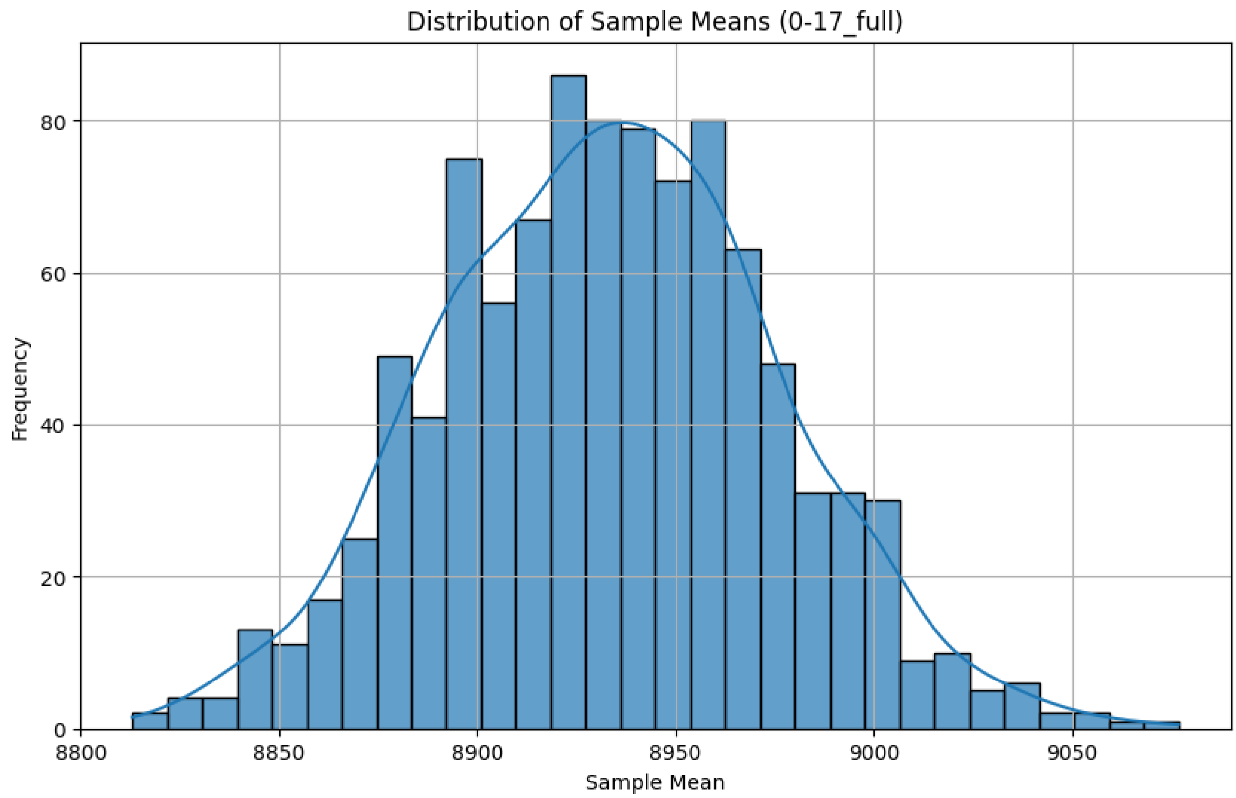
Sample Size: 3,000

- Widest CLT CI: Age Group 0-17, with a range of 370.99.
- Widest Bootstrap CI: Age Group 55+, with a range of 381.99.
- The age groups 0-17 (CLT CI) and 55+ (Bootstrap CI) have the widest confidence intervals. This indicates more variability in the amount spent by individuals in these age groups compared to others.

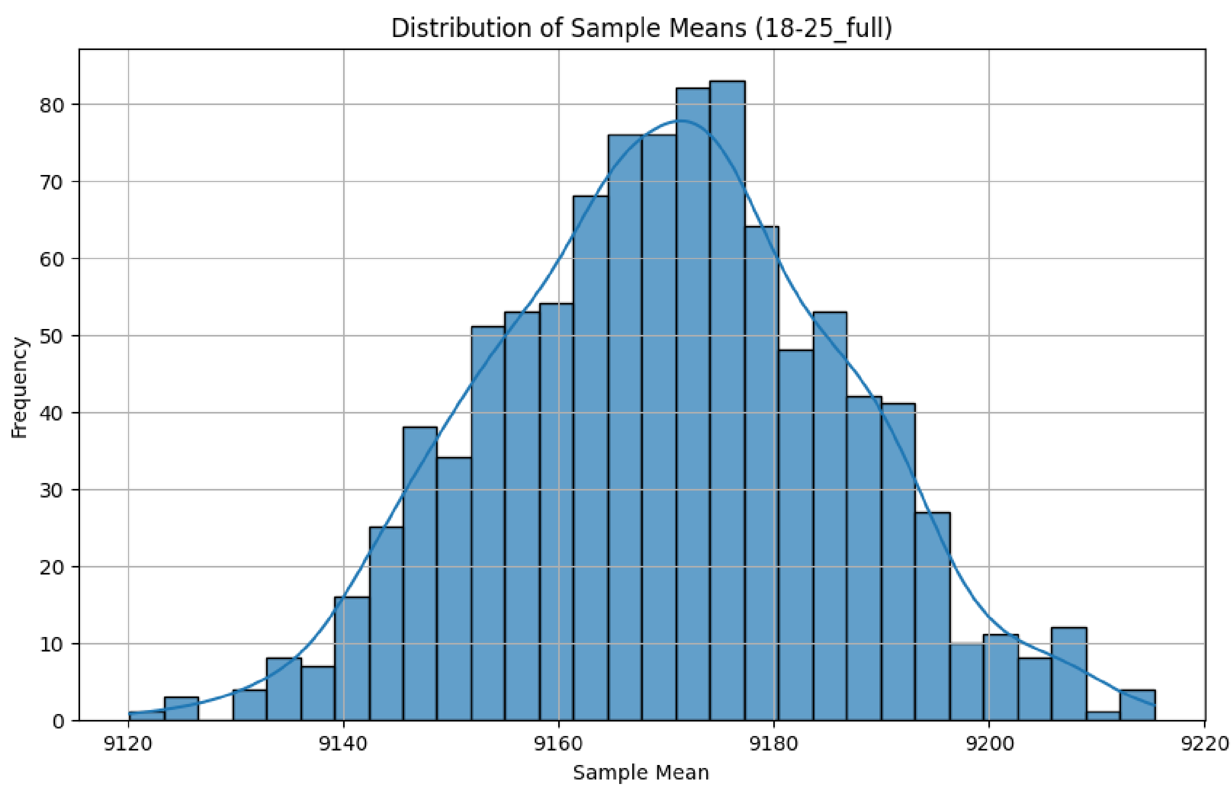
Sample Size: 30,000

- Widest CLT CI: Age Group 51-55, with a range of 163.53.
- Widest Bootstrap CI: Age Group 51-55, with a range of 163.97.
- The age group 51-55 has the widest confidence intervals. This indicates more variability in the amount spent by individuals in this age group compared to others.

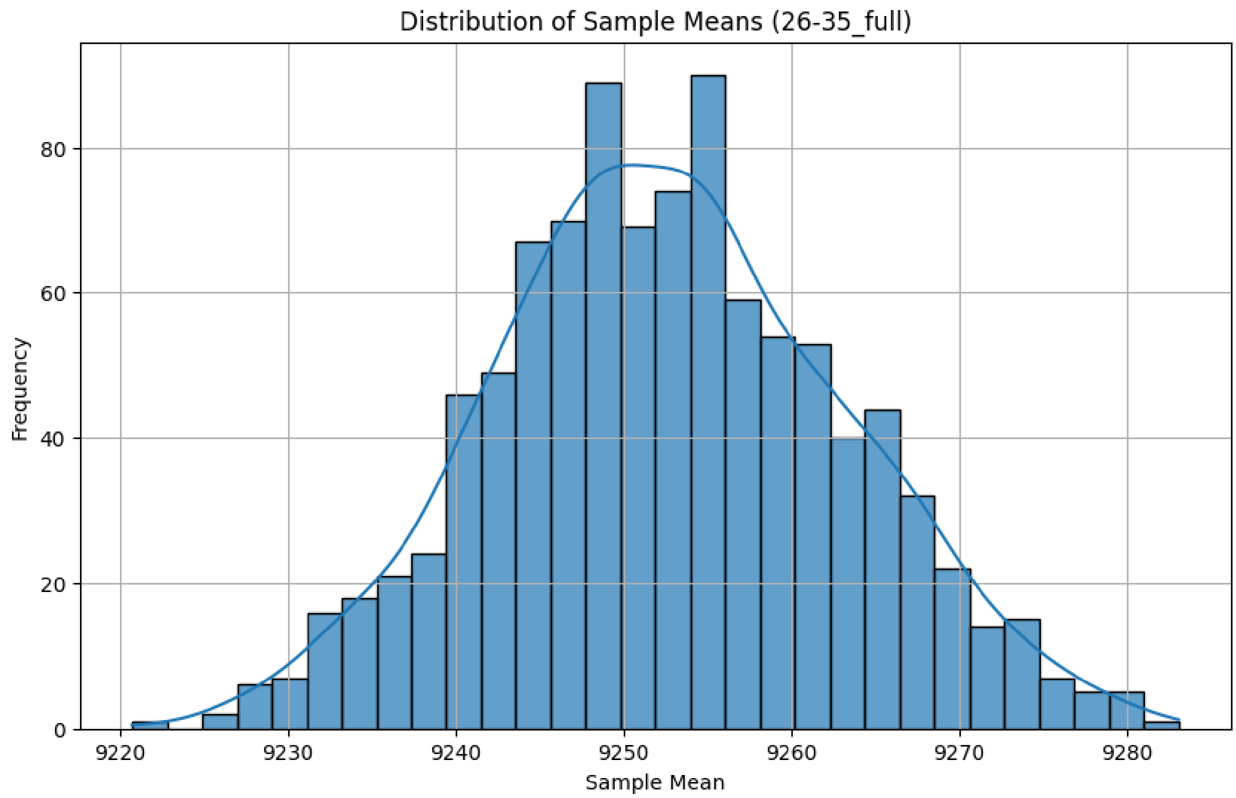
```python
for key, means in all_means.items():
    plt.figure(figsize=(10, 6))
    sns.histplot(means, bins=30, kde=True, edgecolor='k', alpha=0.7)
    plt.title(f'Distribution of Sample Means ({key})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()
```

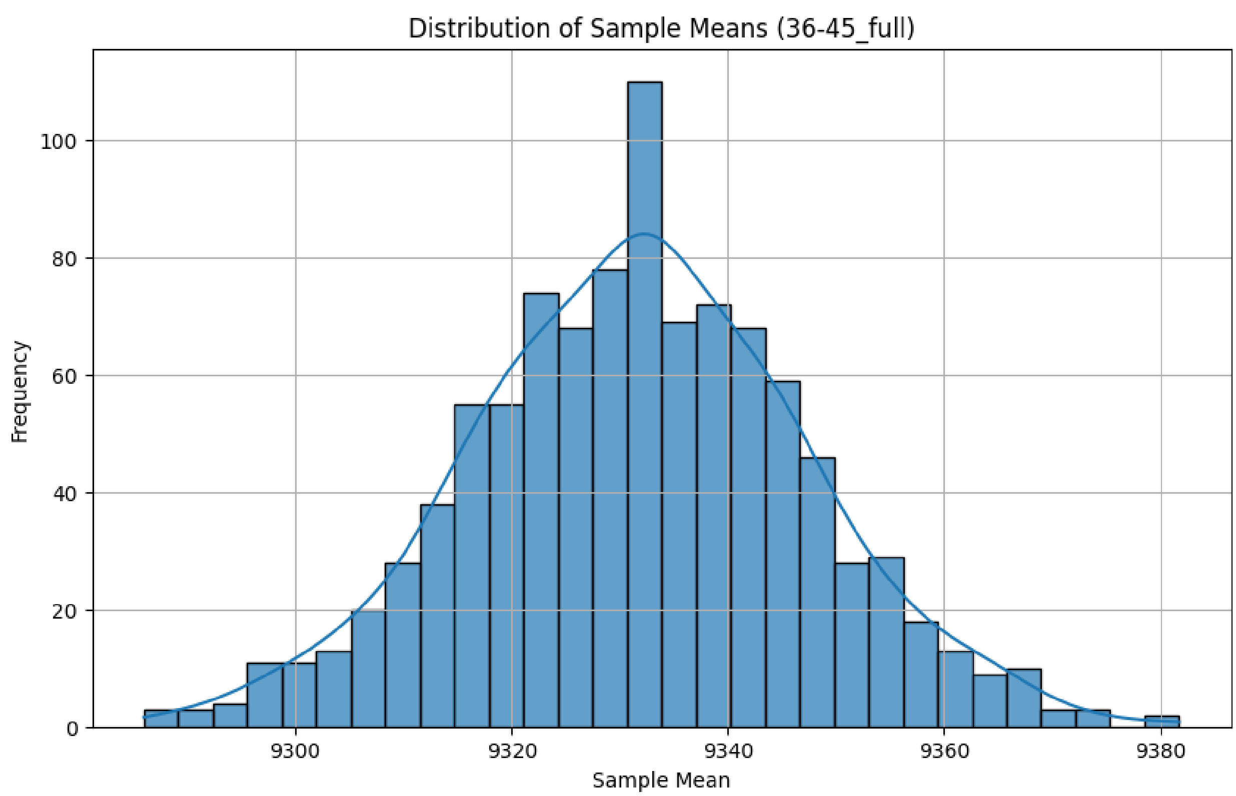


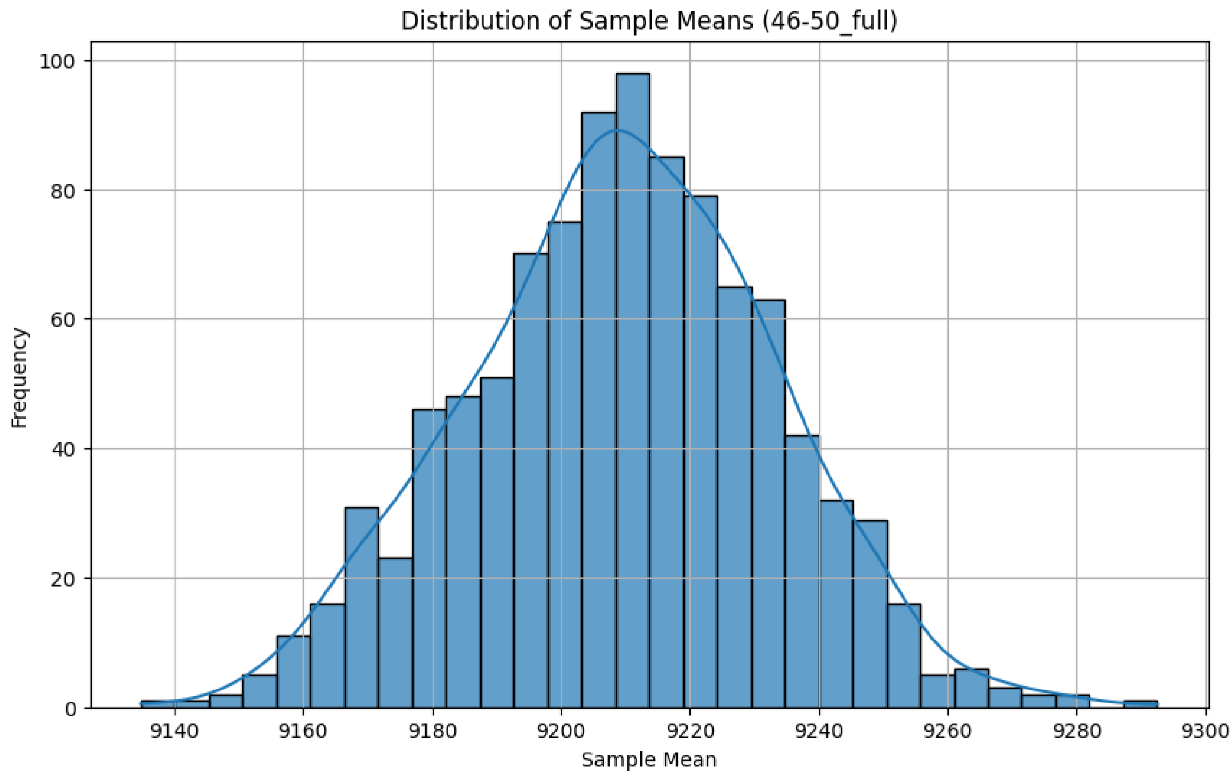
Distribution of Sample Means (0-17_full)
Frequency
Sample Mean
0
20
40
60
80
8800
8850
8900
8950
9000
9050



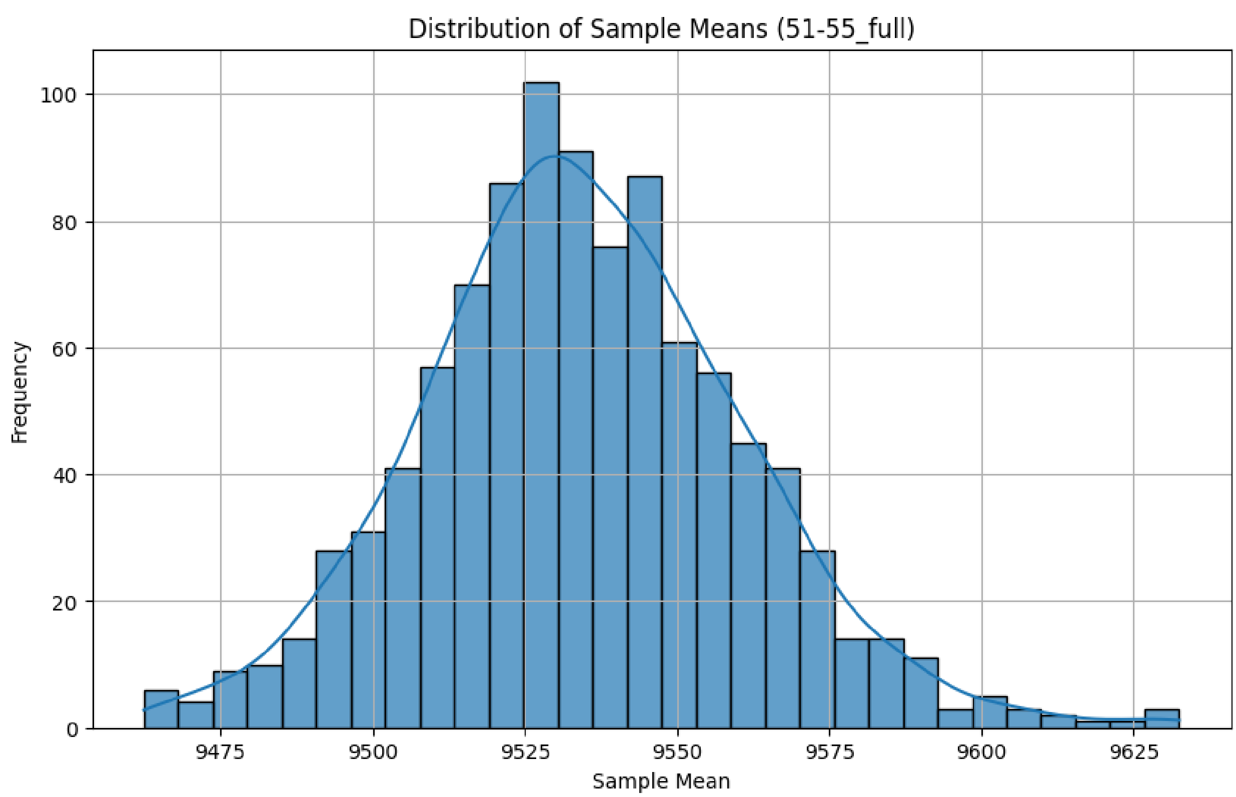
Distribution of Sample Means (18-25_full)
Frequency
Sample Mean
0
10
20
30
40
50
60
70
80
9120
9140
9160
9180
9200
9220


Distribution of Sample Means (26-35_full)
Frequency
0
20
40
60
80
9220
9230
9240
9250
9260
9270
9280
Sample Mean

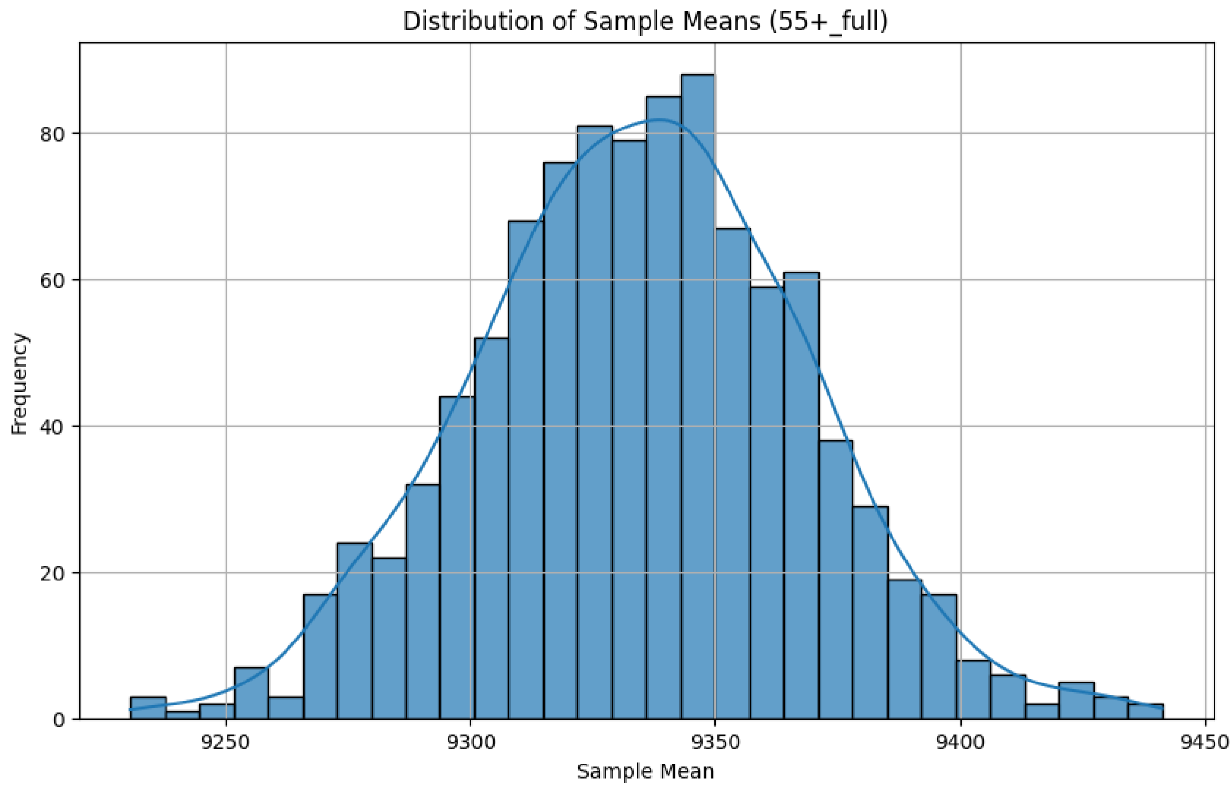

Distribution of Sample Means (36-45_full)
Frequency
0
20
40
60
80
100
9300
9320
9340
9360
9380
Sample Mean

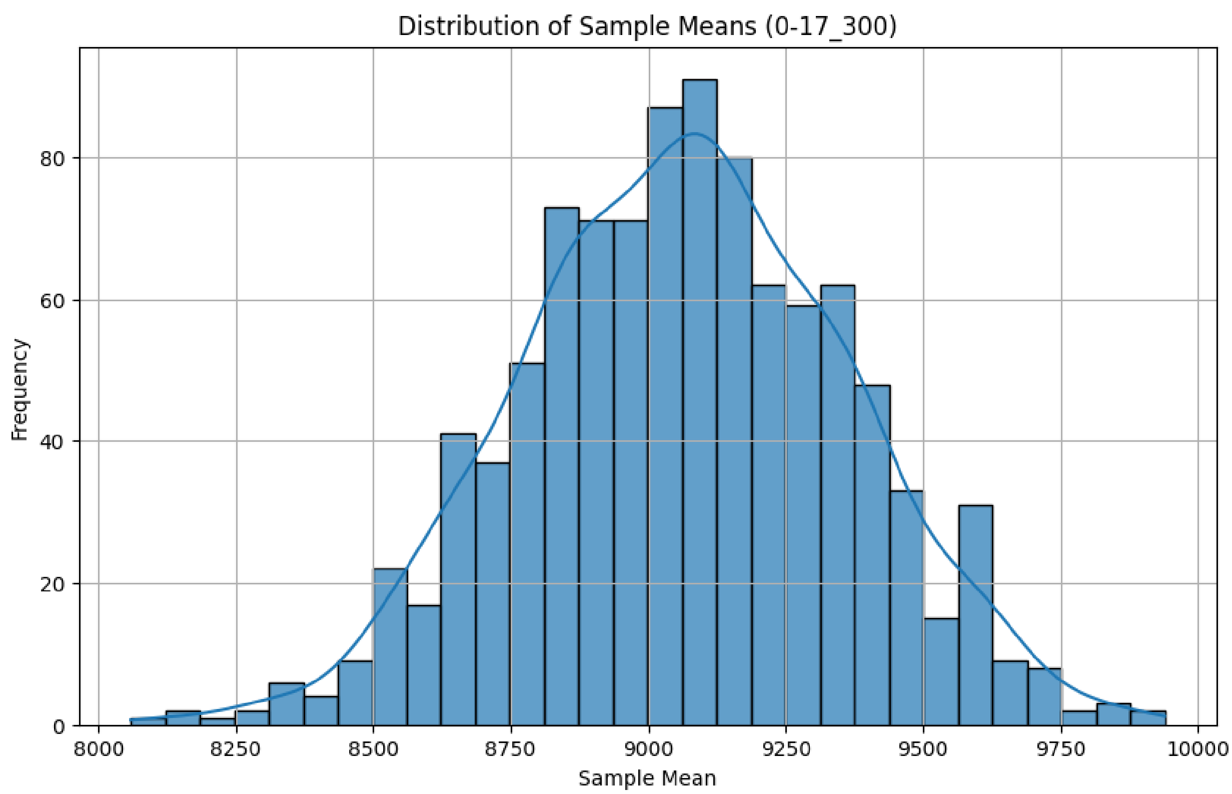


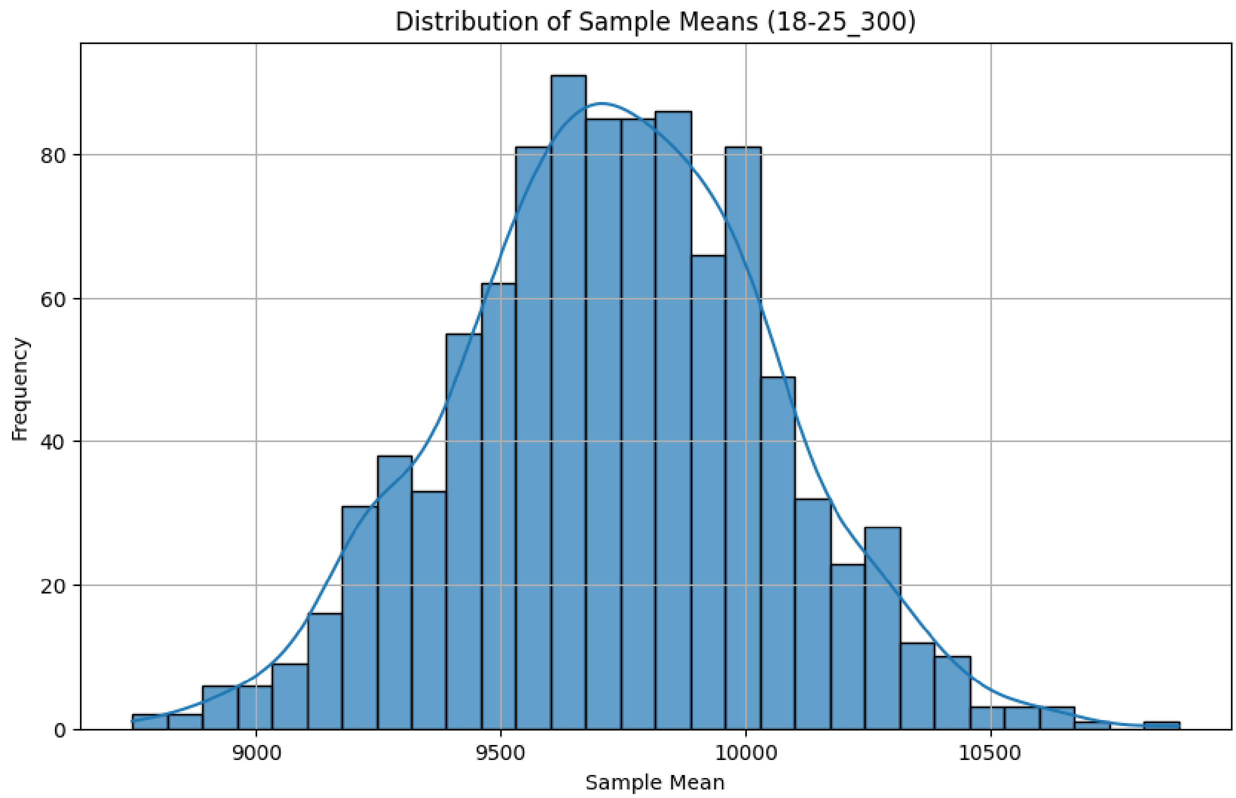
Distribution of Sample Means (46-50_full)
Frequency
Sample Mean
0
20
40
60
80
100
9140
9160
9180
9200
9220
9240
9260
9280
9300

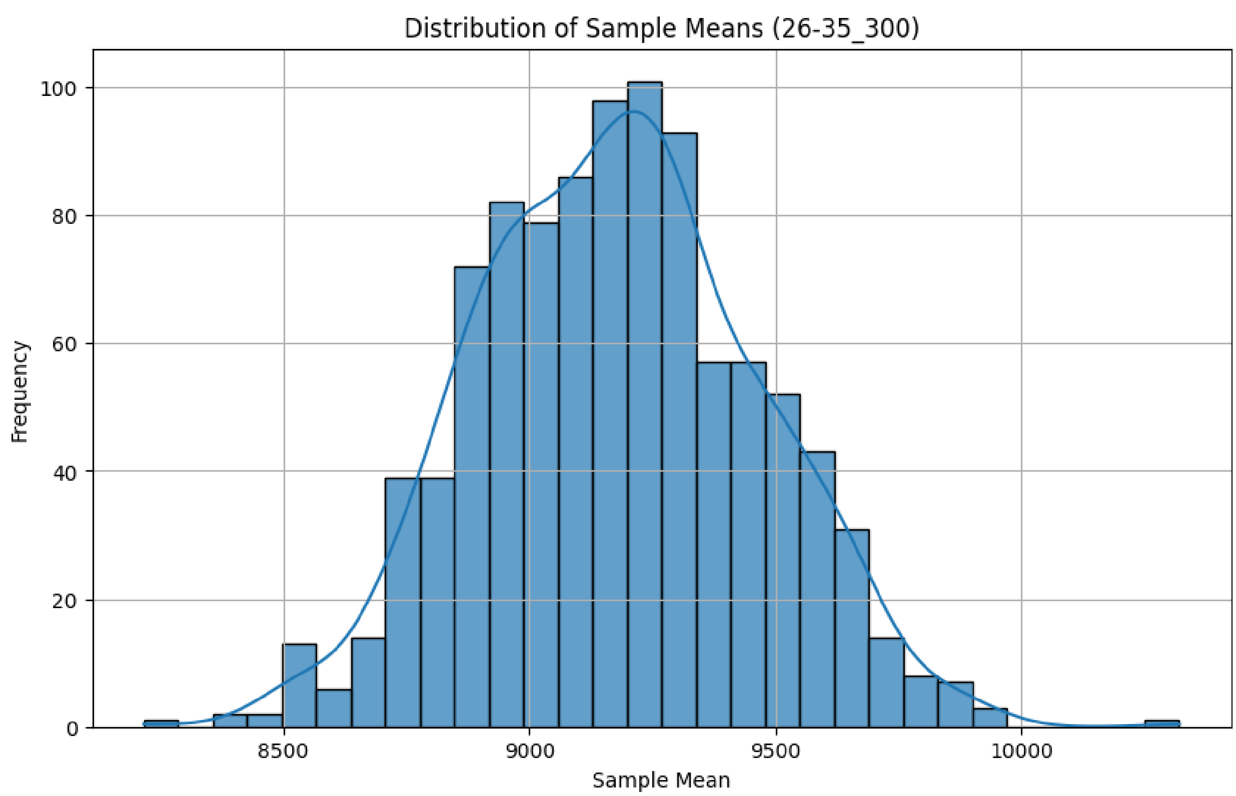


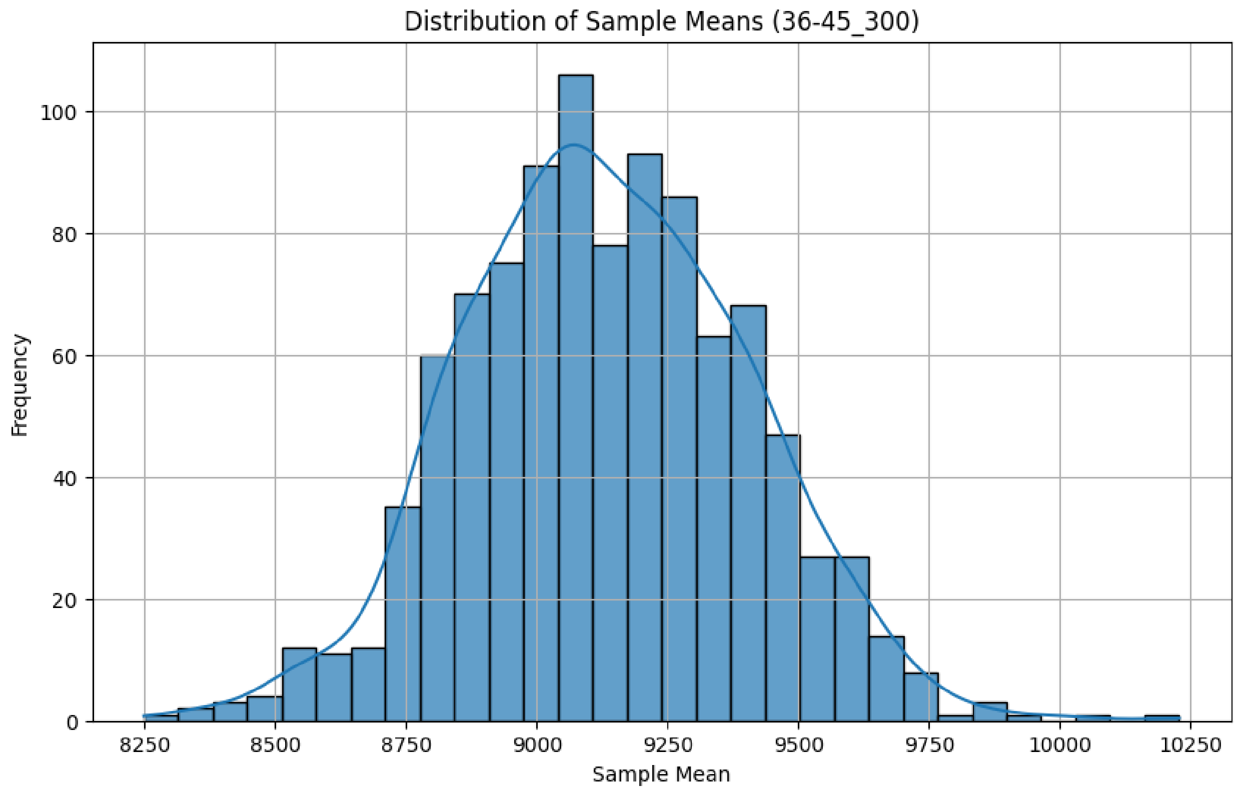
Distribution of Sample Means (51-55_full)
Frequency
Sample Mean
0
20
40
60
80
100
9475
9500
9525
9550
9575
9600
9625


Distribution of Sample Means (55+_full)
Frequency
Sample Mean
0
20
40
60
80
9250
9300
9350
9400
9450

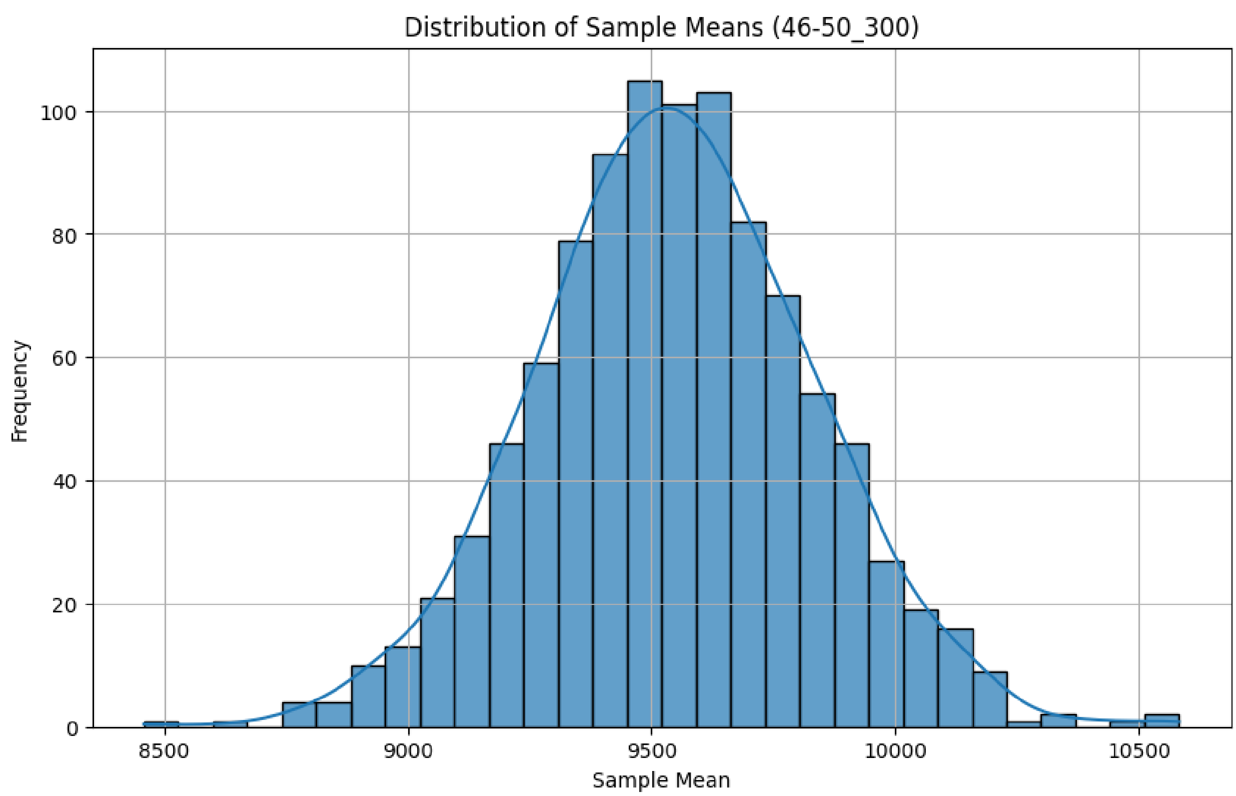

Distribution of Sample Means (0-17_300)
Frequency
Sample Mean
0
20
40
60
80
8000
8250
8500
8750
9000
9250
9500
9750
10000


Distribution of Sample Means (18-25_300)
Frequency
Sample Mean
0
20
40
60
80
9000
9500
10000
10500

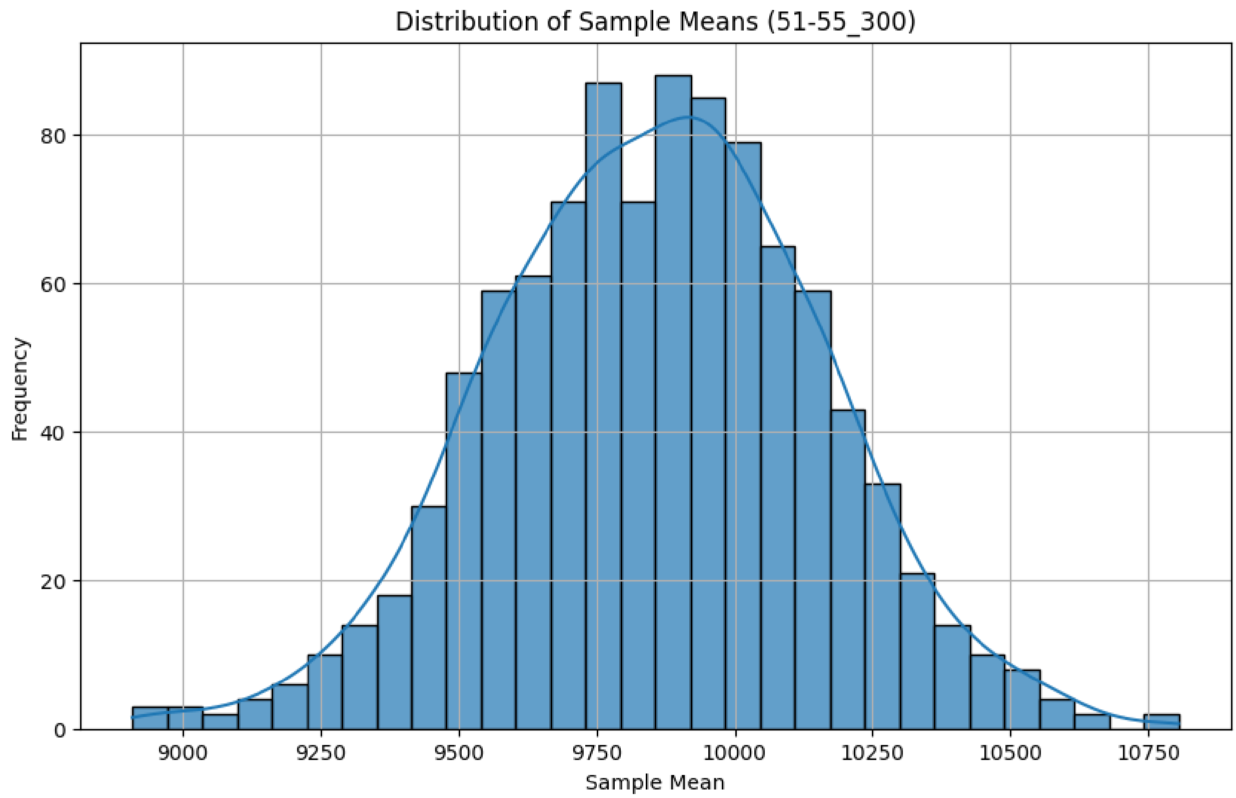

Distribution of Sample Means (26-35_300)
Frequency
Sample Mean
0
20
40
60
80
100
8500
9000
9500
10000

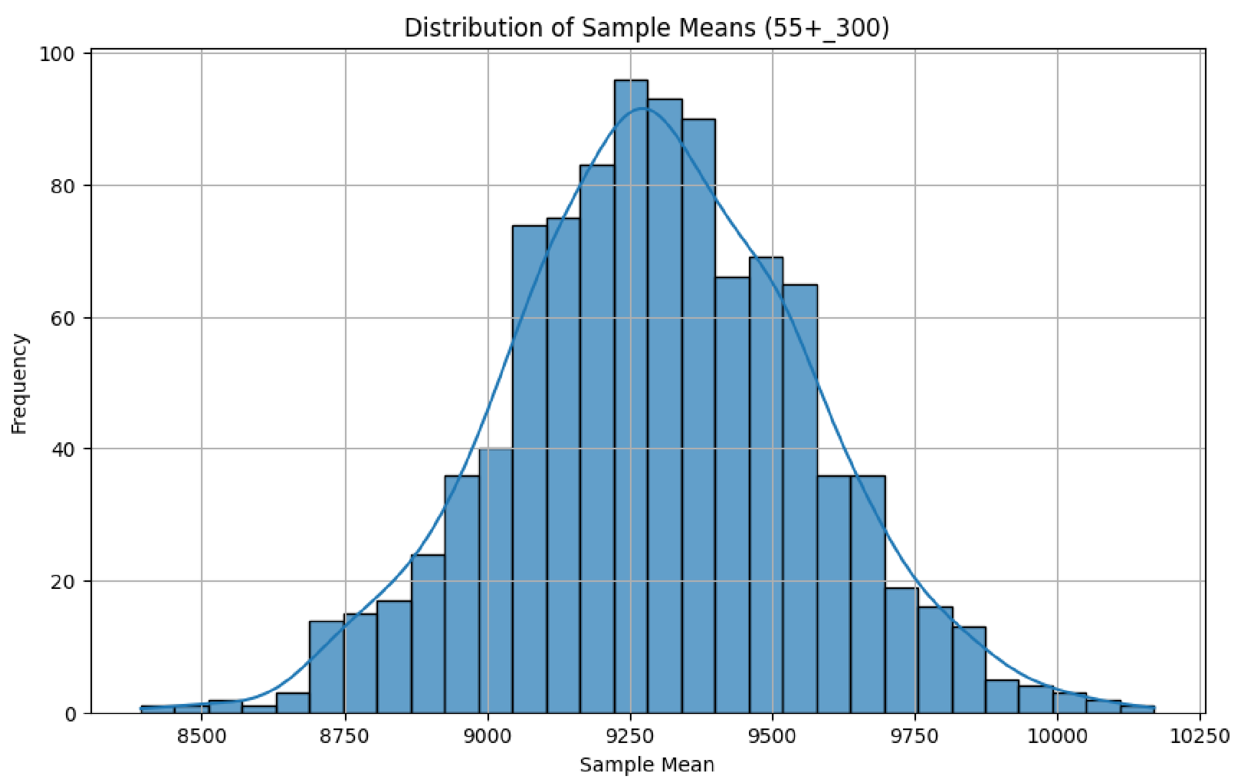


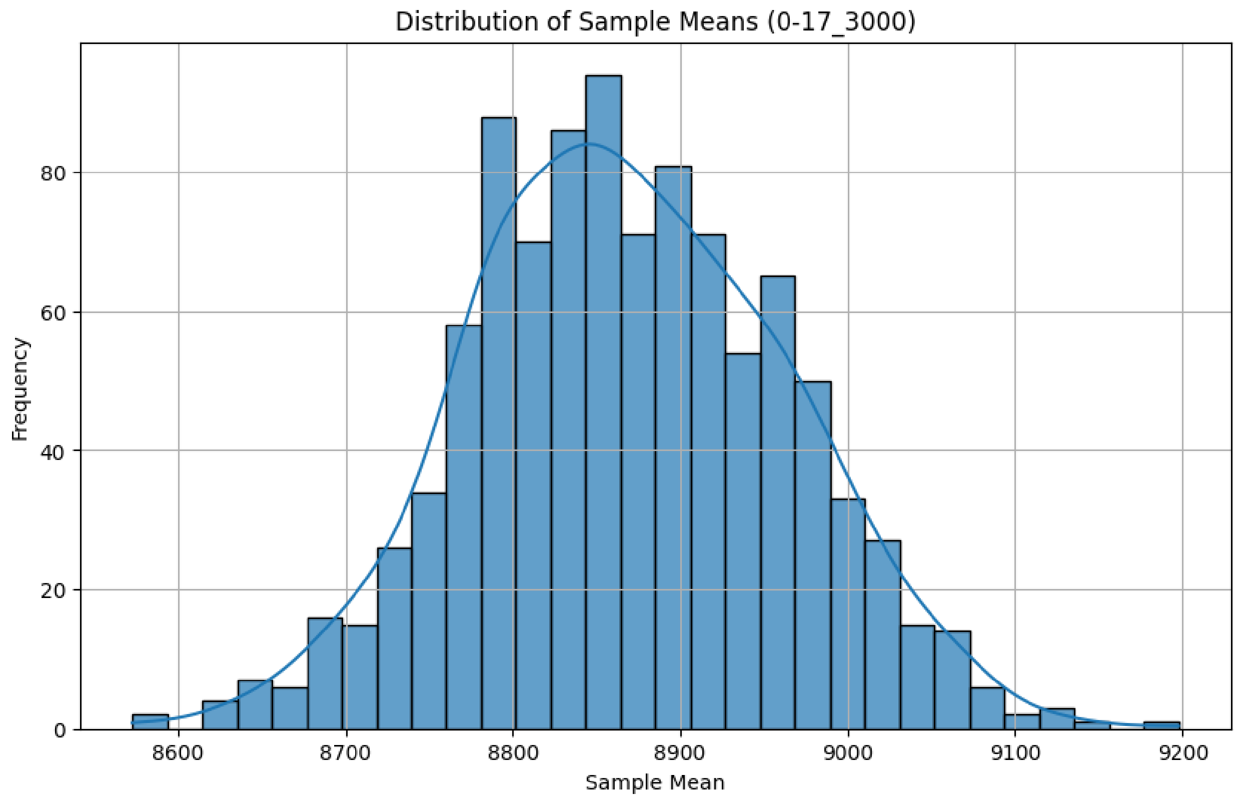
Distribution of Sample Means (36-45_300)
Frequency
Sample Mean
0
20
40
60
80
100
8250
8500
8750
9000
9250
9500
9750
10000
10250



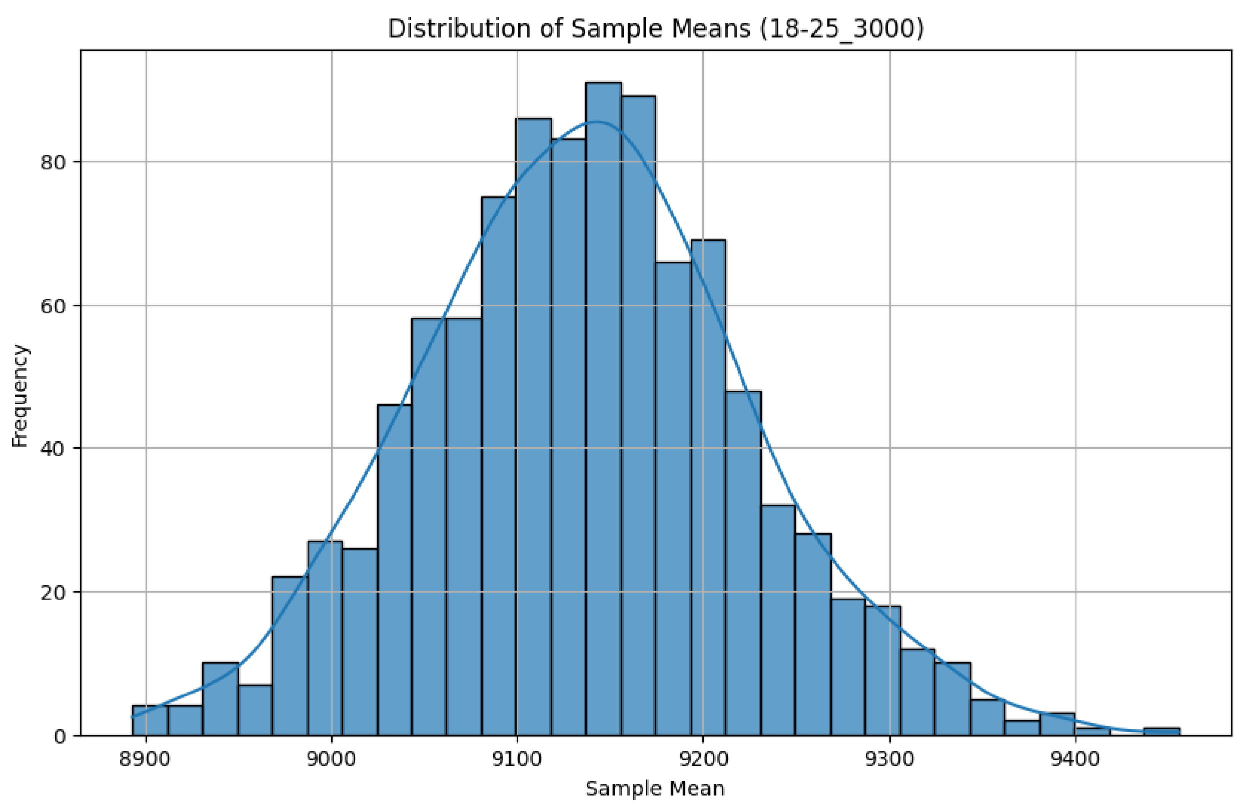
Distribution of Sample Means (46-50_300)
Frequency
Sample Mean
0
20
40
60
80
100
8500
9000
9500
10000
10500


Distribution of Sample Means (51-55_300)
Frequency
0
20
40
60
80
9000
9250
9500
9750
10000
10250
10500
10750
Sample Mean

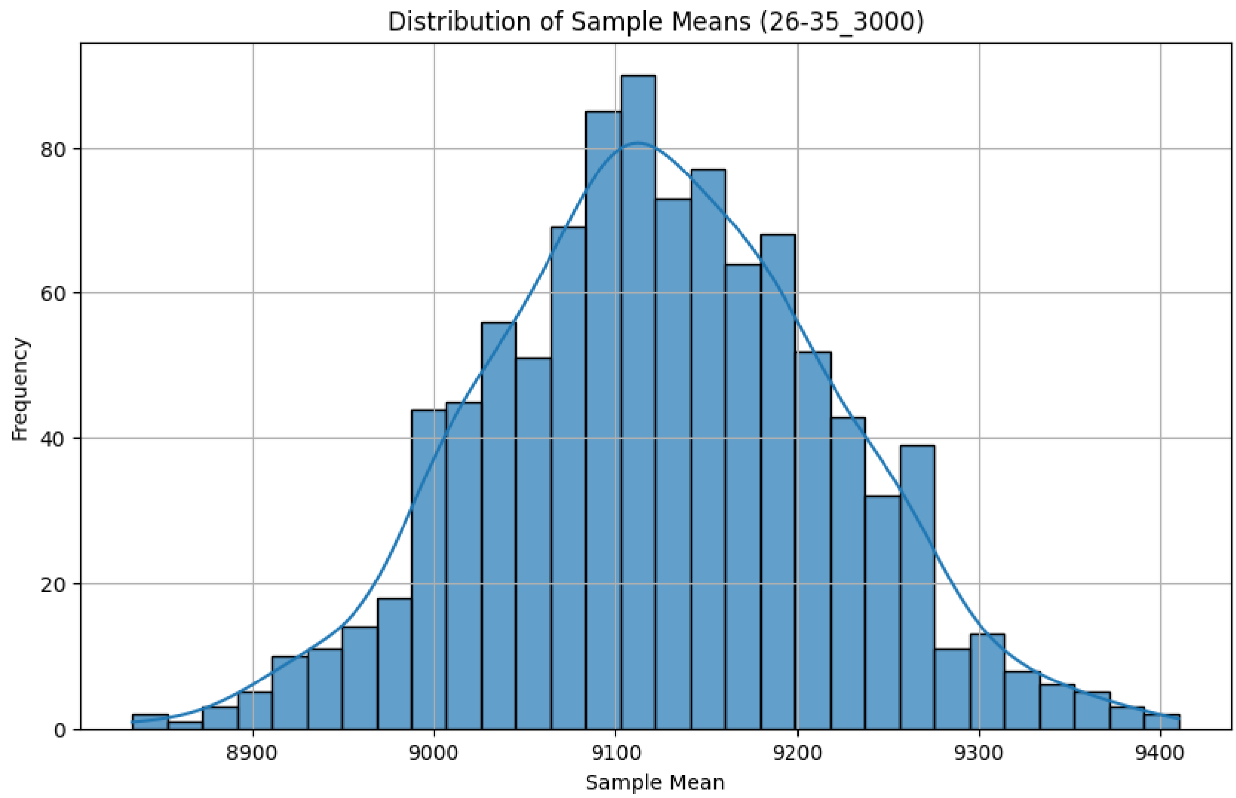

Distribution of Sample Means (55+_300)
Frequency
0
20
40
60
80
100
8500
8750
9000
9250
9500
9750
10000
10250
Sample Mean

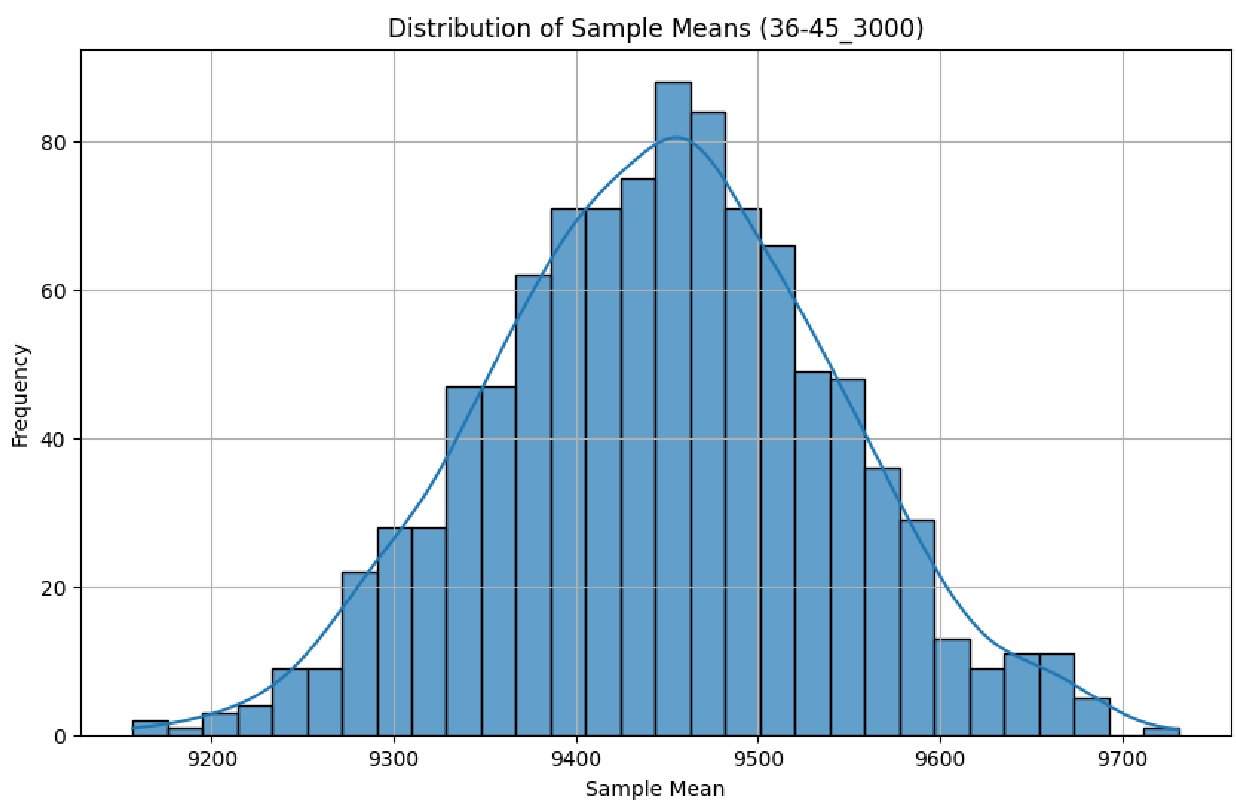


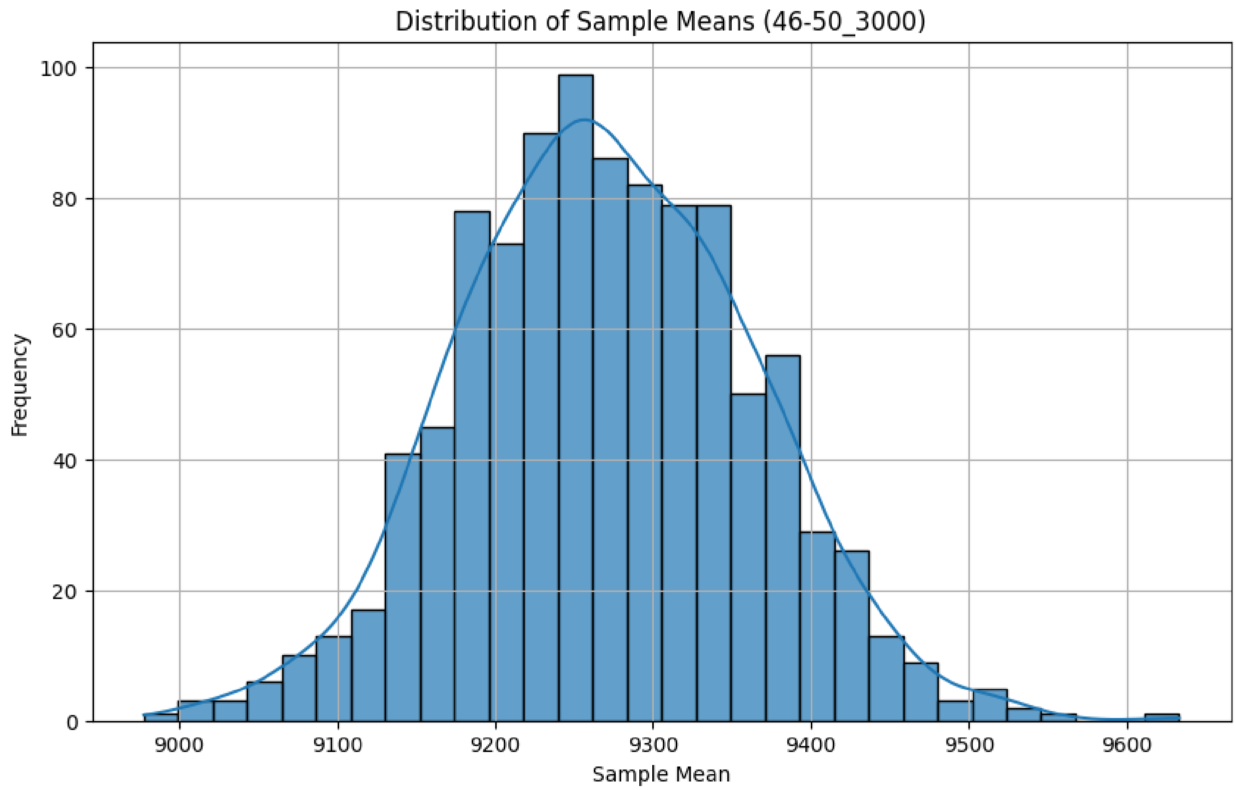
Distribution of Sample Means (0-17_3000)
Frequency
0
20
40
60
80
8600
8700
8800
8900
9000
9100
9200
Sample Mean

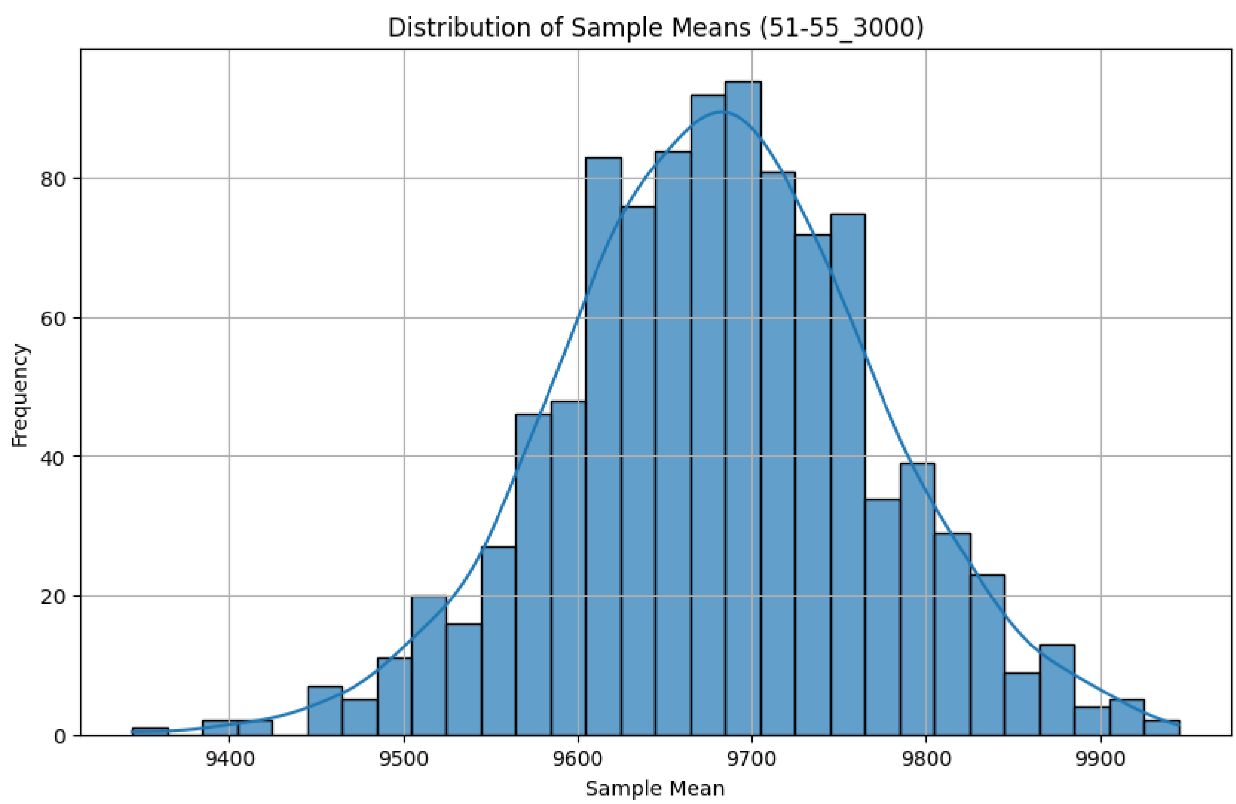


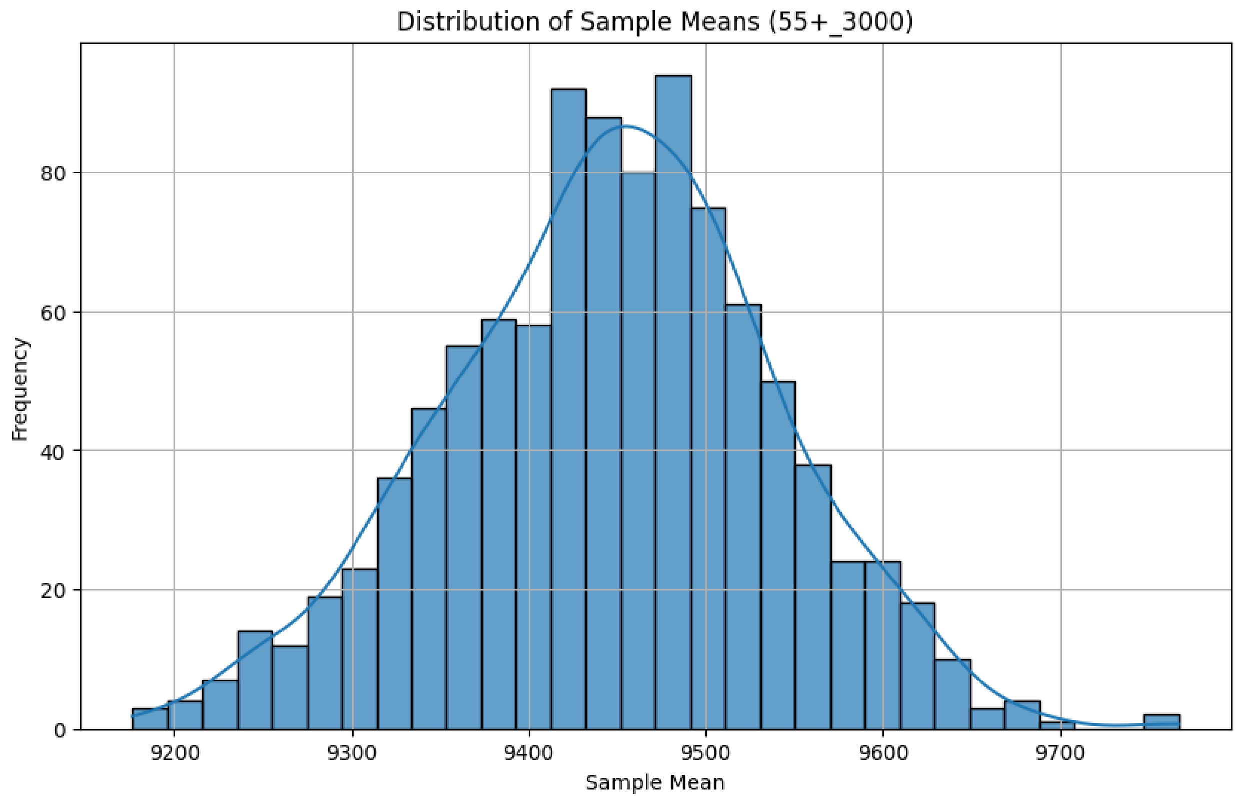
Distribution of Sample Means (18-25_3000)
Frequency
0
20
40
60
80
8900
9000
9100
9200
9300
9400
Sample Mean


Distribution of Sample Means (26-35_3000)
Frequency
Sample Mean
0
20
40
60
80
8900
9000
9100
9200
9300
9400

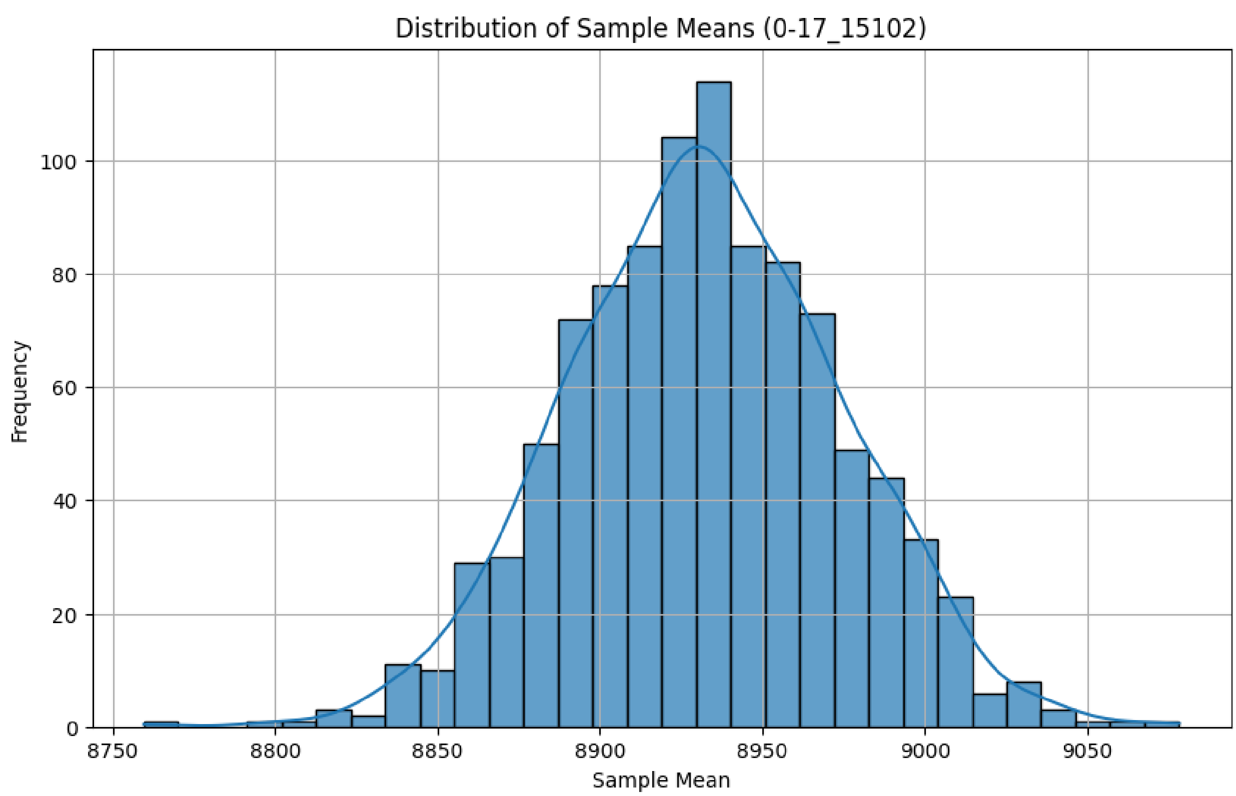

Distribution of Sample Means (36-45_3000)
Frequency
Sample Mean
0
20
40
60
80
9200
9300
9400
9500
9600
9700


Distribution of Sample Means (46-50_3000)
Frequency
Sample Mean
0
20
40
60
80
100
9000
9100
9200
9300
9400
9500
9600

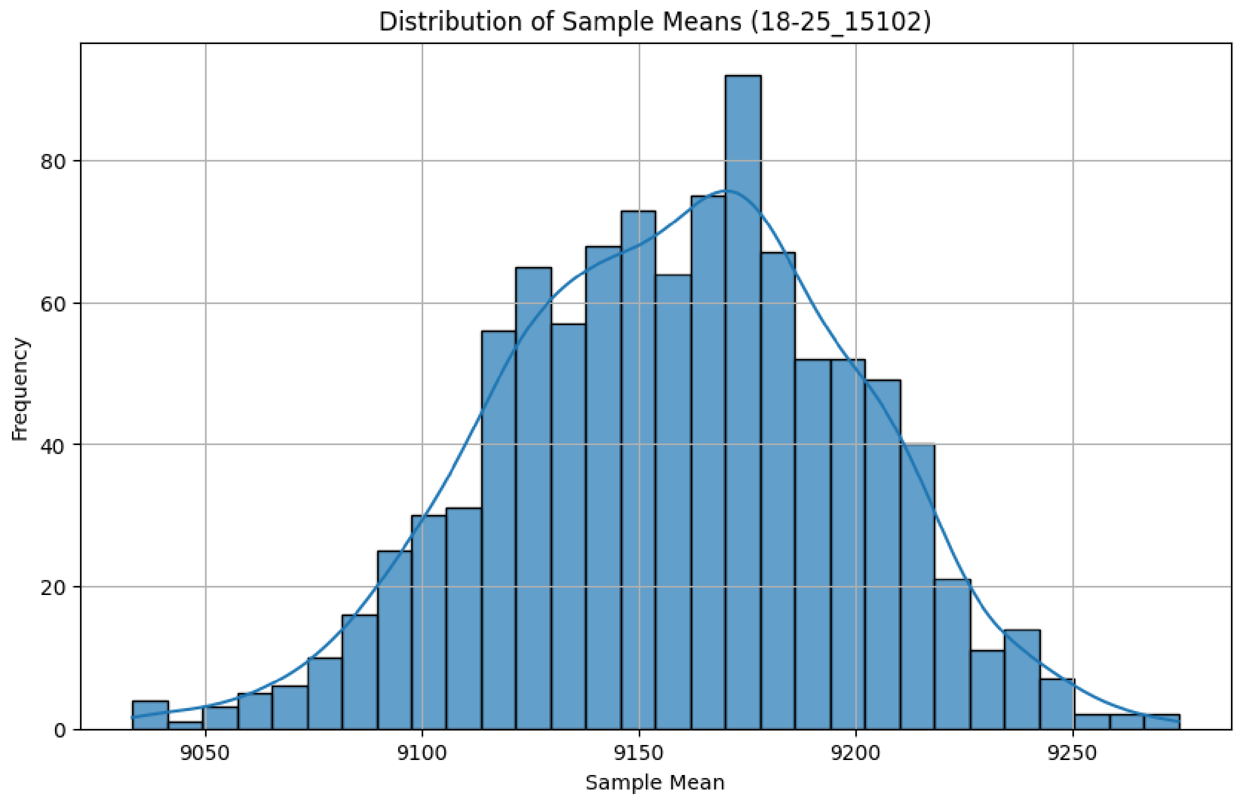

Distribution of Sample Means (51-55_3000)
Frequency
Sample Mean
0
20
40
60
80
9400
9500
9600
9700
9800
9900

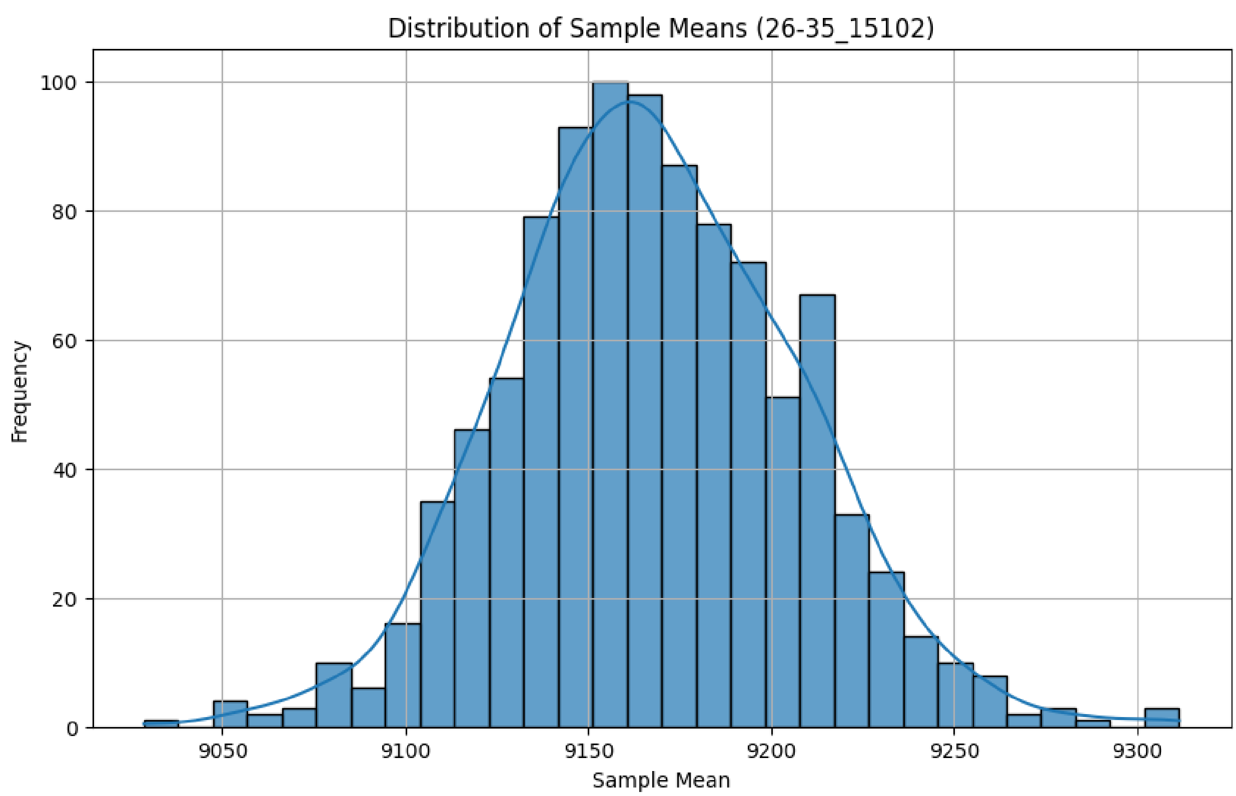


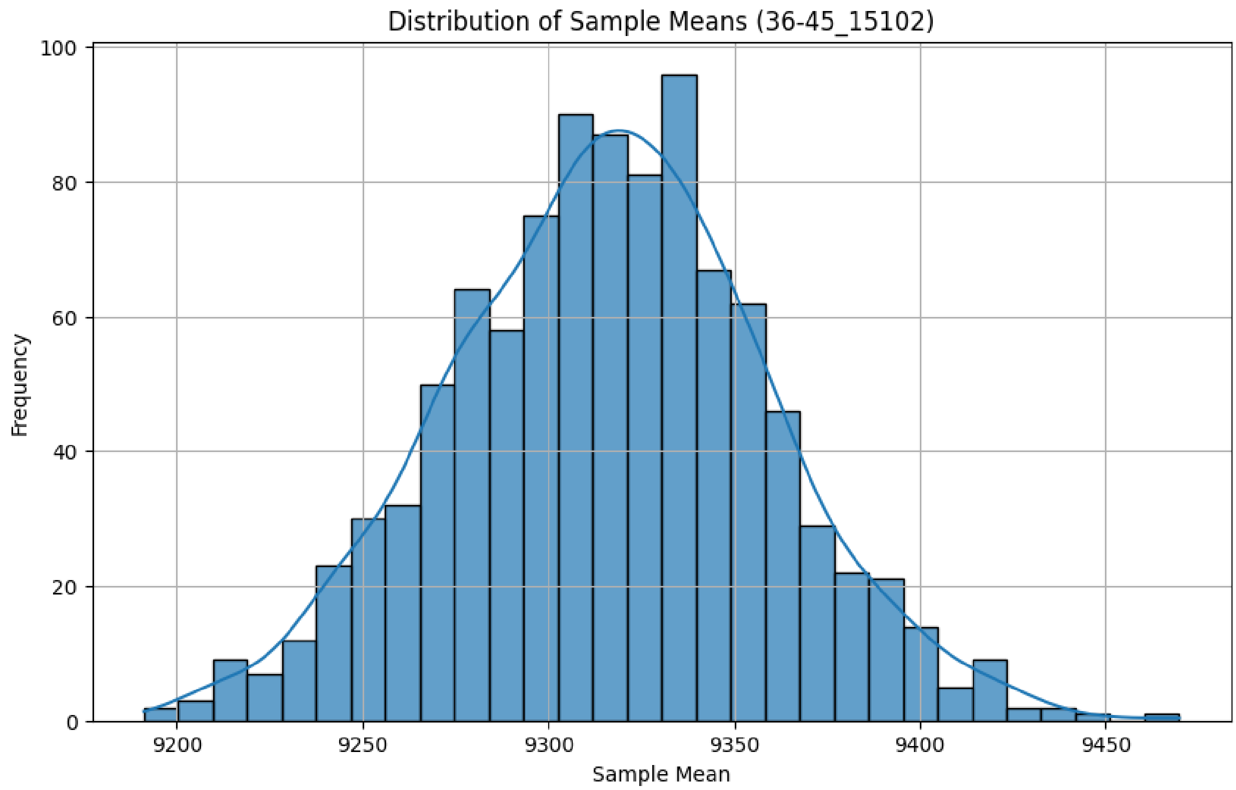
Distribution of Sample Means (55+_3000)
Frequency
Sample Mean
0
20
40
60
80
9200
9300
9400
9500
9600
9700

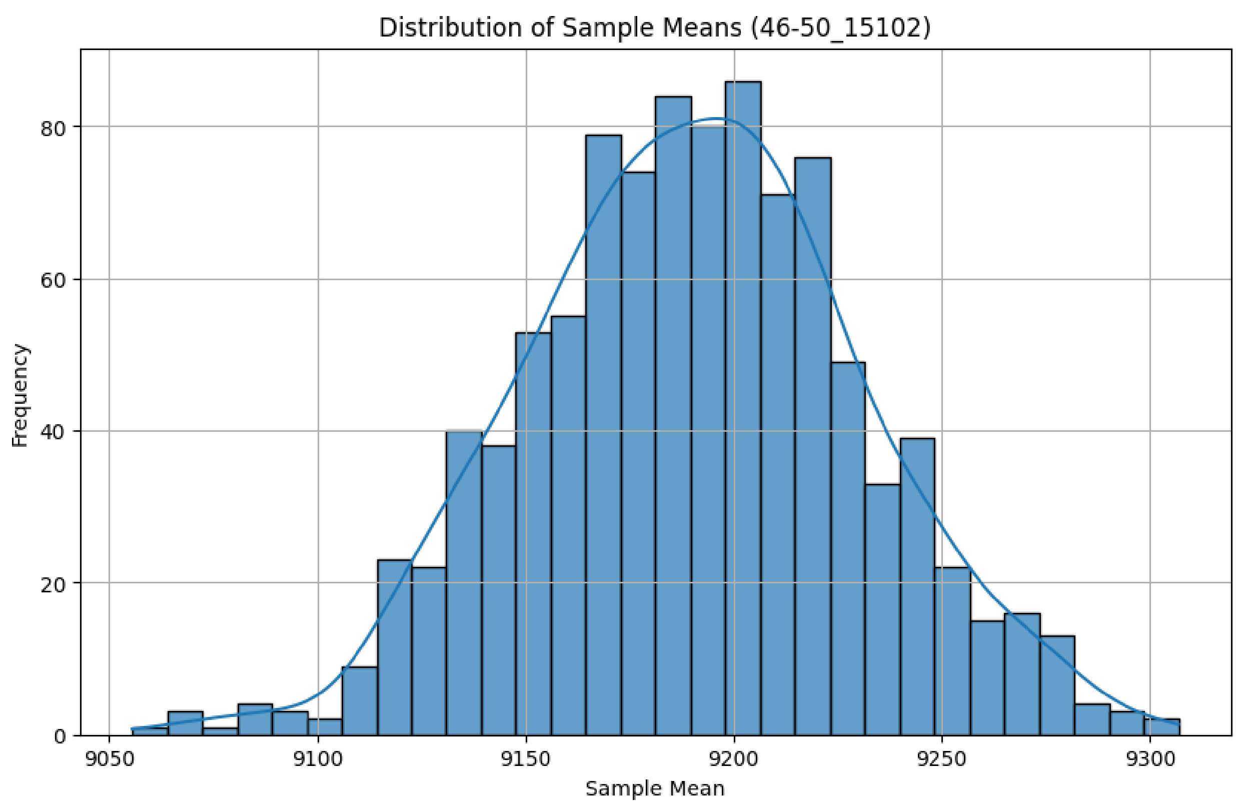


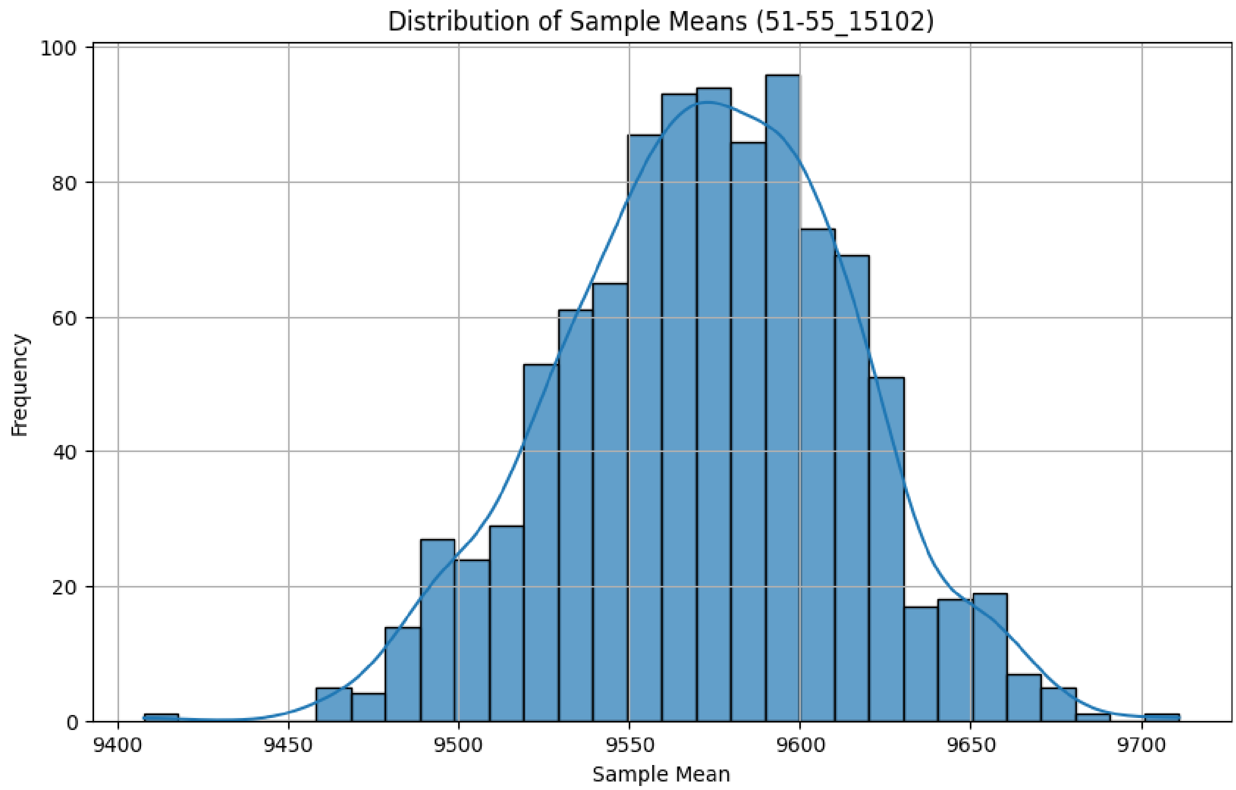
Distribution of Sample Means (0-17_15102)
Frequency
Sample Mean
0
20
40
60
80
100
8750
8800
8850
8900
8950
9000
9050

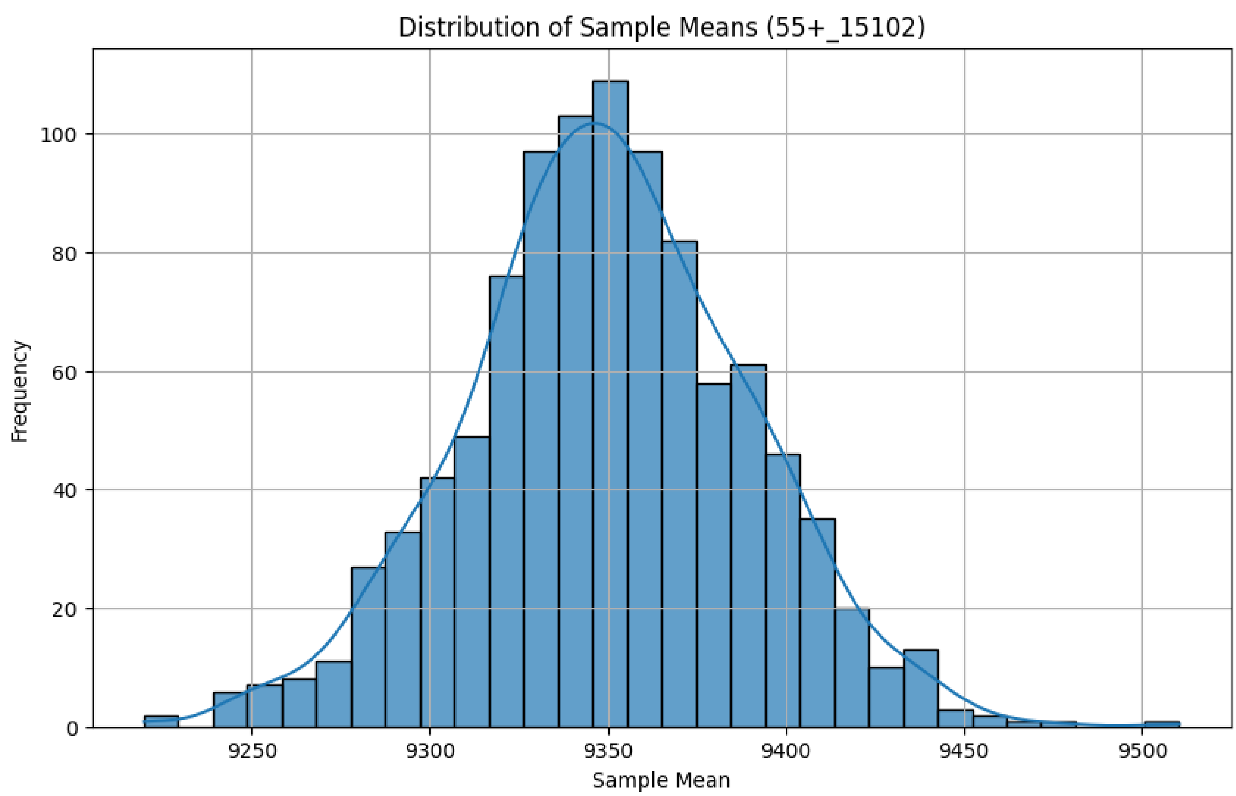

Distribution of Sample Means (18-25_15102)
Frequency
0
20
40
60
80
9050
9100
9150
9200
9250
Sample Mean


Distribution of Sample Means (26-35_15102)
Frequency
0
20
40
60
80
100
9050
9100
9150
9200
9250
9300
Sample Mean


Distribution of Sample Means (36-45_15102)
Frequency
Sample Mean
0
20
40
60
80
100
9200
9250
9300
9350
9400
9450


Distribution of Sample Means (46-50_15102)
Frequency
Sample Mean
0
20
40
60
80
9050
9100
9150
9200
9250
9300


Distribution of Sample Means (51-55_15102)
Frequency
Sample Mean
0
20
40
60
80
100
9400
9450
9500
9550
9600
9650
9700


Distribution of Sample Means (55+_15102)
Frequency
Sample Mean
0
20
40
60
80
100
9250
9300
9350
9400
9450
9500

7. Report

- The average purchase for females is 8734.56.
- The average purchase for males is 9437.52.
- Males tend to spend more on average than females.
- Male customers have slightly higher quartile values than female customers.
- Higher quartile values and more transactions lead to a higher average for male customers.
- The 51-55 age group has the highest average purchase, followed by the 55+ age group.
- The 26-35 age group makes the most purchases across different product categories.
- Both the 51-55 age group and unmarried individuals have a high average purchase.
- Product Category 1 is the most popular among males.
- Product Category 5 is the most popular among females.
- Female customers show more variability in spending, indicating more outliers.
- The average purchase of married and unmarried customers is similar.

8. Final Insights

- Focus on promoting higher-priced products and bundles to male customers, who have a higher average purchase.
- Create personalized offers and discounts for female customers to encourage higher spending and reduce variability.
- For males: Emphasize Product Category 1 in marketing campaigns, ads, and promotions.
- For females: Highlight Product Category 5, incorporating it into targeted advertisements and offers.
- Age Group 51-55: Tailor premium products and exclusive offers to this age group, as they have the highest average purchase. -Age Group 55+: Focus on products that cater to the needs and interests of this age group, such as health-related or leisure products.
- Age Group 26-35: Promote diverse product categories through cross-selling and up-selling strategies to capitalize on their frequent purchases across different categories.
- Ensure adequate stock levels for Product Category 1 and Product Category 5 to meet the demand from male and female customers, respectively.
- Use the insights to tailor email marketing campaigns and online advertisements to specific customer segments, focusing on their spending patterns and product preferences.