Problem Statement —

Create customer profiles by conducting customer segmentation based on age groups, genders, occupations, product categories, demographics, etc., to understand patterns and trends in how customers make purchasing decisions.

Report Structure —

1. Preparing Data
2. Basic statistics
3. Data Cleaning
4. Graphical Summary
5. Customer Profiling & Advance Statistics
6. Business Insights
7. Recommendations

Please note every section started from new page and, a section can span multiple pages.

Preparing Data

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv -O walmart.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv
To: /content/walmart.csv
100% 23.0M/23.0M [00:02<00:00, 9.73MB/s]
```

```
import re

import pandas as pd
from pandas.api.types import CategoricalDtype
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
walmart = pd.read_csv('walmart.csv')
```

Basic statistics

In this section, we'll try to identify —

1. Number of rows and columns of data.
2. Data type of each column, non-null values in each column and memory usage by the dataset.
3. How data looks like, by taking sample of 5 rows out of it.
4. Distinct values in each column.
5. If dataset contains duplicated rows.

```
# Number of rows and columns of data.
walmart.shape
```

```
(550068, 10)
```

```
# Data type of each column, non-null values in each column and memory usage by the dataset.
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

```
# How data looks like, by taking sample out of it.
walmart.sample(5)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
256977	1003626	P00066642	M	26-35	17	B	3	0	1	11574
126141	1001447	P00152242	M	18-25	4	A	0	0	14	14664
276685	1000678	P00367142	M	26-35	0	A	0	1	11	5910
329364	1002777	P00138542	M	18-25	4	B	2	0	5	8715
56286	1002679	P00110842	F	18-25	4	A	2	0	1	19544

```
# Distinct values in each column
walmart.nunique().rename('Distinct Values').reset_index().T.style.hide(axis='columns')
```

index	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
Distinct Values	5891	3631	2	7	21	3	5	2	20	18105

```
# If dataset contains duplicated rows.
walmart.duplicated().sum()
```

```
0
```

Statistical summary

Show code

```
# For numerical column.
describe = statistics[['purchase']].describe().T
describe['range'] = describe['max'] - describe['min']
describe['mode'] = statistics[['purchase']].mode().iloc[0]
describe['IQR'] = describe['75%'] - describe['25%']
describe[ [ 'count', 'range', 'mean', 'mode', 'std', 'min', '25%', '50%', '75%', 'max', 'IQR'] ].style.format('{:.2f}')
```

	count	range	mean	mode	std	min	25%	50%	75%	max	IQR
purchase	550068.00	23949.00	9263.97	7011.00	5023.07	12.00	5823.00	8047.00	12054.00	23961.00	6231.00

```
# For categorical columns.
statistics = statistics.select_dtypes(exclude='int64').describe().T
statistics['percentage'] = 100 * statistics['freq'] / statistics['count']
statistics.style.format({'percentage': '{:.2f}%'})
```

	count	unique	top	freq	percentage
user_id	550068	5891	1001680	1026	0.19%
product_id	550068	3631	P00265242	1880	0.34%
gender	550068	2	male	414259	75.31%
age	550068	7	26-35	219587	39.92%
occupation	550068	21	o_4	72308	13.15%
city_category	550068	3	B	231173	42.03%
stay_in_current_city_years	550068	5	1	193821	35.24%
marital_status	550068	2	single	324731	59.03%
product_category	550068	20	pc_5	150933	27.44%

Summary

- The dataset is huge and contains only two numerical columns, **Purchase** and **Stay in current city (in years)**.
- There are no null or missing values, and no duplicate rows.

Data Cleaning

In this section —

- Find categorical and numerical columns and convert them.
- Renaming numerics to more readable values in categorical columns.
- Detecting and Treating Outliers.

```
# Renaming the columns in small caps.
walmart.columns = walmart.columns.str.lower()
```

```
dedup = walmart[['user_id', 'age', 'marital_status']].drop_duplicates()
dedup[dedup['age'] == '0-17'][['age', 'marital_status']].value_counts()
```

age	marital_status	count
0-17	**0**	218

dtype: int64

```
# Nominal Variables — User ID, Occupation, Marital Status, Product Category
nominal_columns = ['user_id', 'marital_status']

for column in nominal_columns:
  walmart[column] = walmart[column].astype('category')

# Ordinal Variables — Age, Stay in Current City(Years)
ordinal_columns = ['age', 'city_category']

for column in ordinal_columns:
  walmart[column] = walmart[column].astype(pd.CategoricalDtype(categories=walmart[column].sort_values().unique(), ordered=True))
walmart.gender = walmart.gender.replace('F', 'female').replace('M', 'male')
walmart.marital_status = walmart.marital_status.cat.rename_categories({0: 'single', 1: 'married'})
```

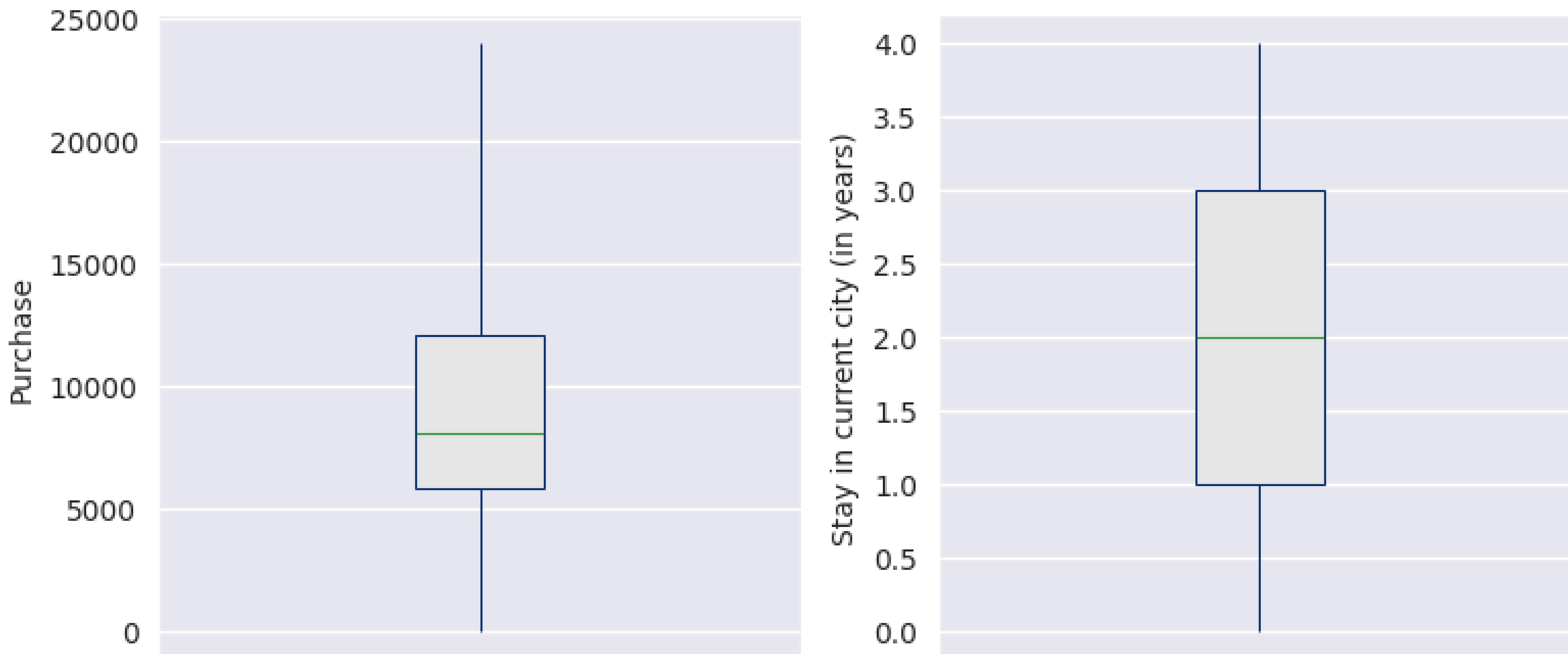
```python
# Other Nominal variable for which order can be helpful.

def extract_number(pc_string):
  return int(re.search(r'\d+', pc_string).group())

# Renaming Occupation column for better readability.
walmart.occupation = 'o_' + walmart.occupation.astype(str)

# Renaming Product Category column for better readability.
walmart.product_category = 'pc_' + walmart.product_category.astype(str)

columns = ['occupation', 'product_category']
for column in columns:
  walmart[column] = walmart[column].astype(pd.CategoricalDtype(categories=sorted(walmart[column].unique(), key=extract_number), ordered=True))
```

```python
# Stay in current city (in years) - at least n years in the city.
walmart.stay_in_current_city_years = walmart.stay_in_current_city_years.replace('4+', '4').astype('int8')
```

```python
# Verifying the changes.
walmart.sample(5)
```

	user_id	product_id	gender	age	occupation	city_category	stay_in_current_city_years	marital_status	product_category	purchase
172435	1002670	P00064042	male	36-45	o_6	B	1	single	pc_3	8071
82594	1000757	P00147742	male	26-35	o_12	A	1	single	pc_1	12126
329413	1002783	P00003242	male	46-50	o_17	C	0	married	pc_8	6074
196705	1000352	P00192042	male	18-25	o_4	A	0	single	pc_5	5330
320756	1001404	P00178942	male	51-55	o_16	B	1	married	pc_5	5398

Outlier Detection and Treatment

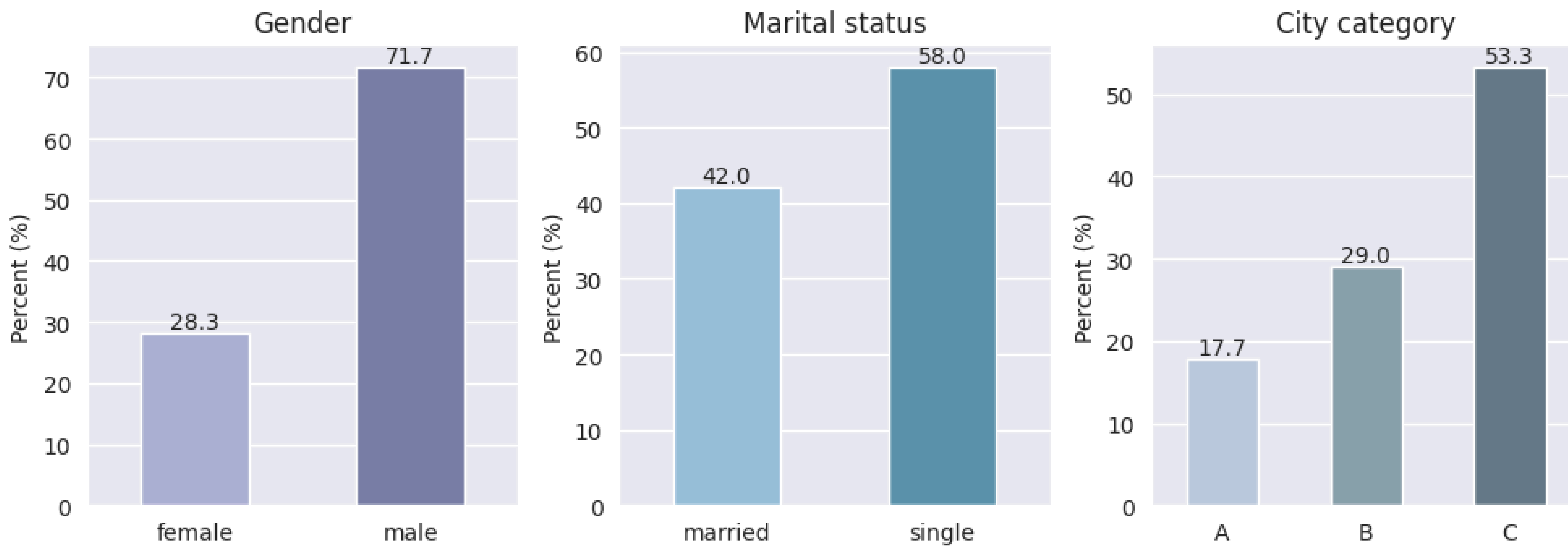


Summary

- Identified and converted categorical and numerical columns.
- There aren't any outliers in *Purchase* and *Stay in current city*, whiskers are extended to 3^{rd} standard deviation.

Graphical Summary

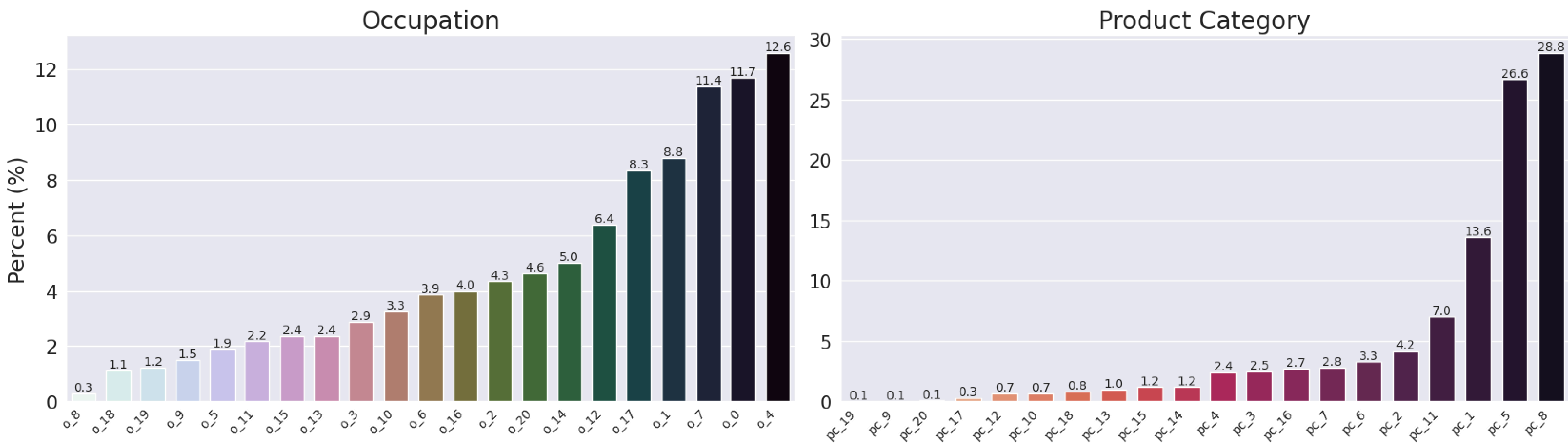
Uni-variate Analysis



- The dataset is heavily male-dominated, with 72% males and 28% females.
- There are 58% more married individuals and 42% single individuals.
- 53% of the individuals are from city C.

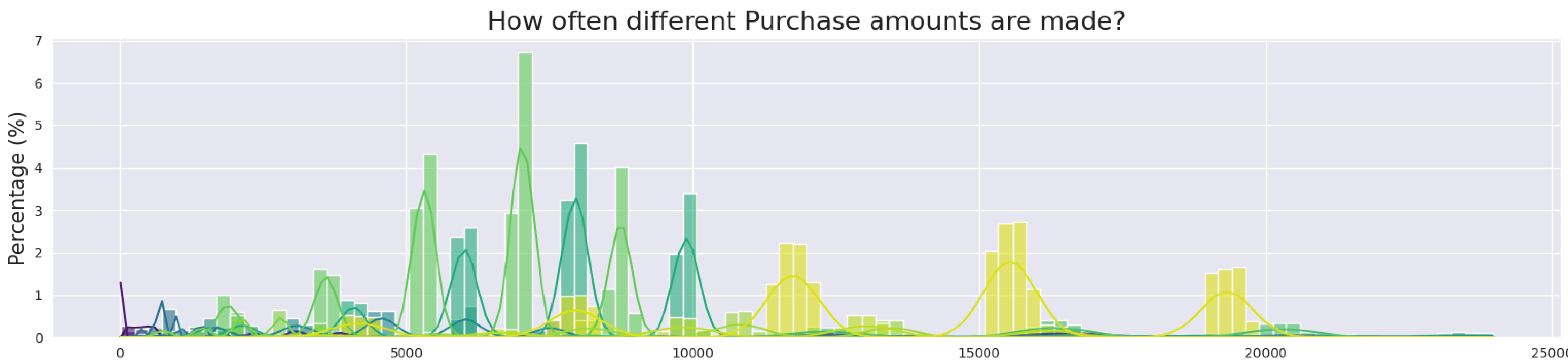
- A staying period of 1 to 3 years is ideal, accounting for 55% of the population.
- 72% of the population are in the age range of 18 to 45.

Please note, in first graph n indicates that a person stayed atleat n year(s) in the city.



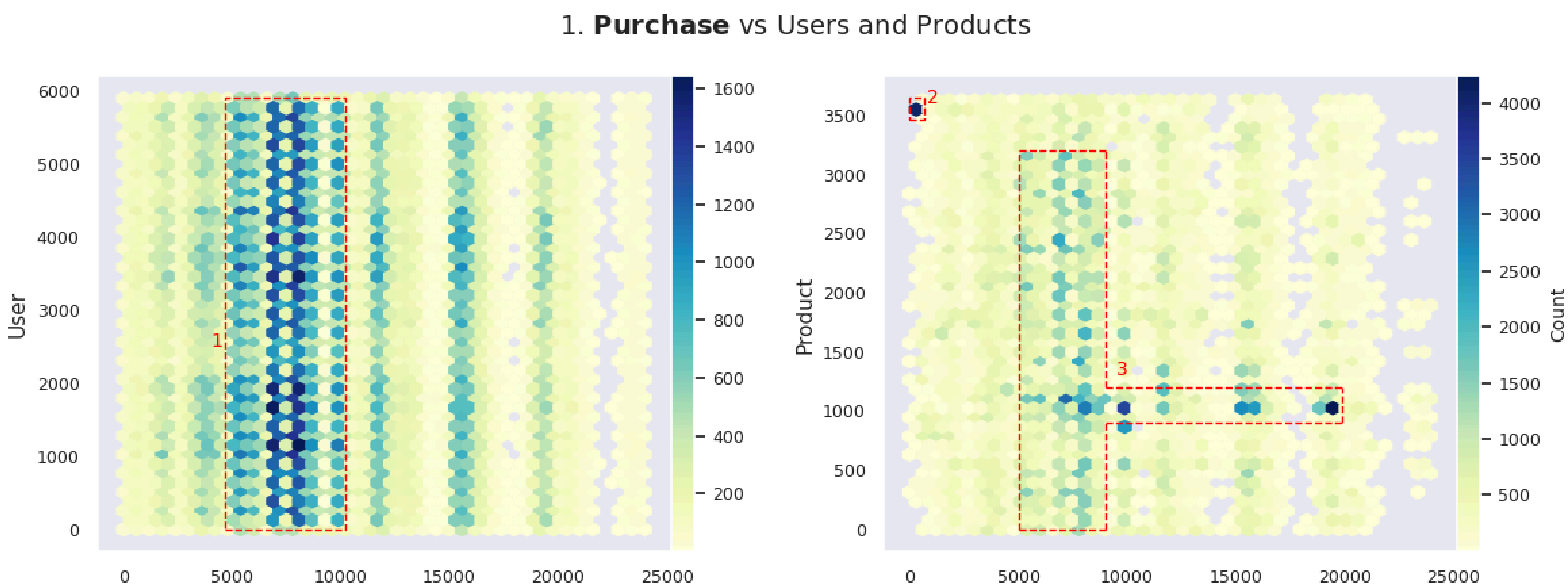
1. Occupation —
   - Top 5 occupations: `o_4`, `o_0`, `o_7`, `o_1`, and `o_17`.
   - The top 5 occupations account for 50% of the customers.
2. Product Category —
   - Top 3 product categories: `pc_8`, `pc_5`, and `pc_1`.
   - The top 3 product categories account for 73% of the products.
   - Later we'll see whether this affects the *sales* of product categories.

Sales Distribution



- The total sales of the given dataset is around 5.1 billion.
- 38% of the total revenue is in-between $5,000$ and $10,000$ purchase amount, to be exact 1.96 billion.
- The top 3 purchase amounts generated sales of around 4.5 million, with almost equal purchase amounts of approximately $16,000$.

Bi-variate Analysis

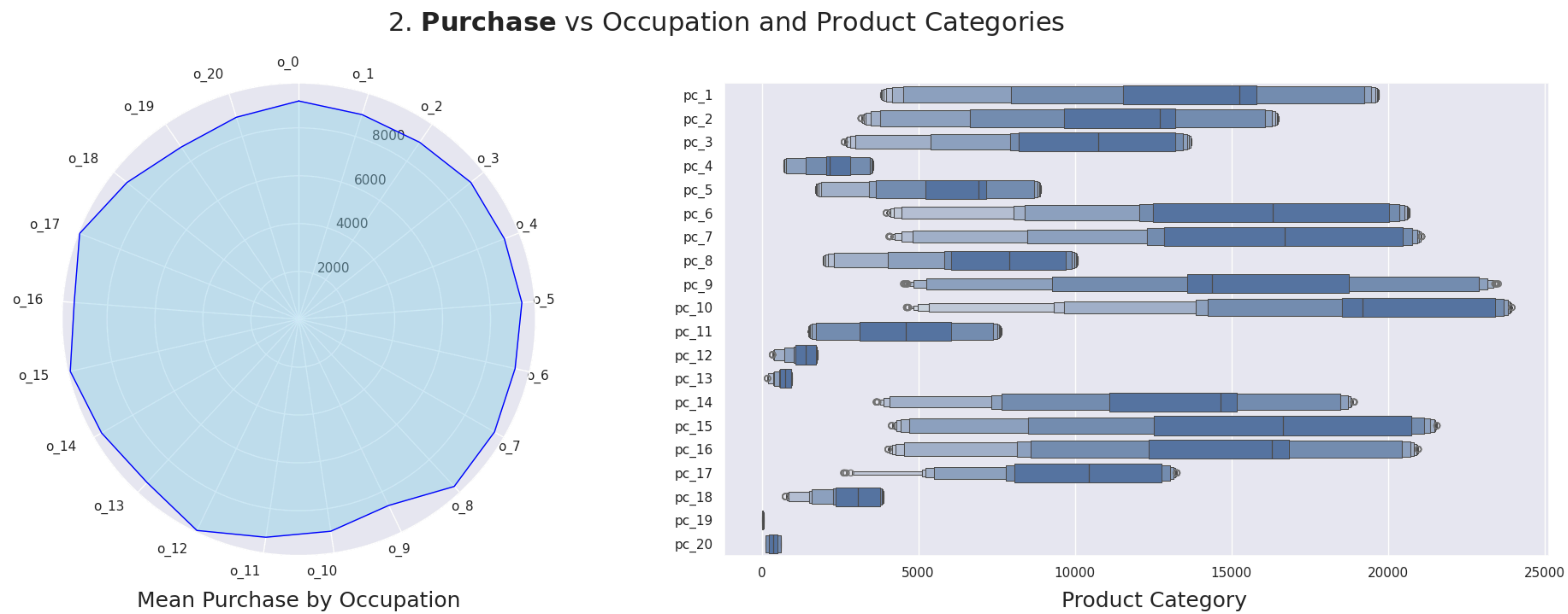


Observations —

The numbers reflect the outlined regions on the graph,

1. 48% of the purchase order is made in-between $5,000$ and $10,000$, generating revenue of 1.96 billions.
2. This *hexbin* aggregates approximately $4,000$ low-value purchase orders, with products from 10 to 12 out of the 20 product categories.
3. It contains 79% of the $3,631$ products from all categories, generating a revenue of 2.49 billion.

> The colors of both graphs are intentionally kept the same so that the difference in purchasing habits for users with respect to products is evident.




```
# Purchase (in millions) by Occupation
walmart.groupby('occupation', observed=False).purchase.sum().apply(lambda x: f"{x / 1_000_000:.1f}").reset_index().T\
  .rename(index={'occupation': 'Occupation', 'purchase': 'Purchase (in millions)'}).style.hide(axis='columns')
```

Occupation	o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_10	o_11	o_12	o_13	o_14	o_15	o_16	o_17	o_18	o_19	o_20
Purchase (in millions)	635.4	424.6	238.0	162.0	666.2	113.6	188.4	557.4	14.7	54.3	115.8	106.8	305.4	71.9	259.5	119.0	238.3	393.3	60.7	73.7	296.6

```
# Mean Purchase by Occupation
walmart.groupby('occupation', observed=False).purchase.mean().apply(lambda x: f"{x:.0f}").reset_index().T\
  .rename(index={'occupation': 'Occupation', 'purchase': 'Mean Purchase'}).style.hide(axis='columns')
```

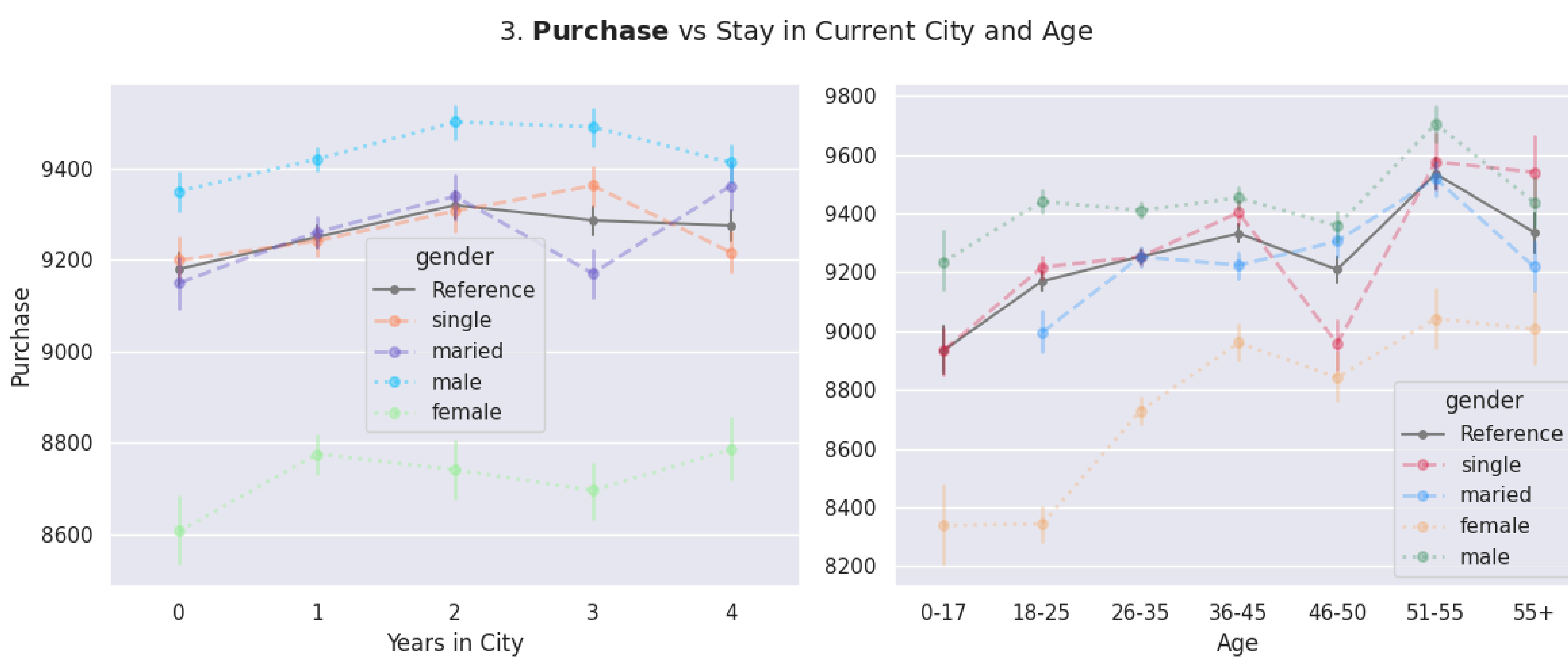
Occupation	o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_10	o_11	o_12	o_13	o_14	o_15	o_16	o_17	o_18	o_19	o_20
Mean Purchase	9124	8953	8952	9179	9214	9333	9257	9426	9533	8638	8959	9214	9797	9306	9501	9779	9394	9821	9170	8711	8836

```
# Number of products in each product category
walmart.groupby('product_category', observed=False).product_id.unique().apply(lambda x: len(x)).reset_index().T \
  .rename(index={'product_category': 'Product Category', 'product_id': '# of Products'}).style.hide(axis='columns')
```

Product Category	pc_1	pc_2	pc_3	pc_4	pc_5	pc_6	pc_7	pc_8	pc_9	pc_10	pc_11	pc_12	pc_13	pc_14	pc_15	pc_16	pc_17	pc_18	pc_19	pc_20
# of Products	493	152	90	88	967	119	102	1047	2	25	254	25	35	44	44	98	11	30	2	3

Observations —

- **Purcase vs Product Categories** graph, along with the table, clearly shows that the number of products in a category does not affect the purchase orders; rather, the category itself does.



Observations —

- **Purchase vs Stay in Current city** graph, purchasing habits for married and single individuals are, on average, the same for the first two years. However, differences can be observed in the 3^{rd} and 4^{th} years, with married individuals spending 194 million less than single individuals in the 3^{rd} year.
- **Purchase vs Age** graph, on average, married individuals spend less than single individuals at any age, except for those aged 46 to 50, where they spend 193 million more than single individuals.

Since the dataset is heavily male-dominated, there is a considerable difference between the average spending for males and females in both categories.

Customer Profiling & Advance Statistics

In this section —

We'll try to answer the following question with a certain level of confidence using the hypothesis testing framework,

- **Gender:** Do women spend more money per transaction than men? Why or why not?
- **Marital Status:** Do married individuals spend more money per transaction than single individuals? Why or why not?
- **Age:** Do individuals in different age groups spend varying amounts of money per transaction? Why or why not?
- Do the confidence intervals of different categories of average spending overlap?

For the preceding categories, the following approach will be used:

1. **Demonstrating Central Limit Theorem:** *Assuming normality*, we will apply the Central Limit Theorem (CLT) to demonstrate it and calculate the population means.

2. **Testing Hypothesis:** We will use the means calculated in step 1 to test the null hypothesis.

3. **Calculating Confidence Intervals:** We will use the t-statistics calculated in step 2 to create confidence intervals at 90%, 95%, and 99% levels.

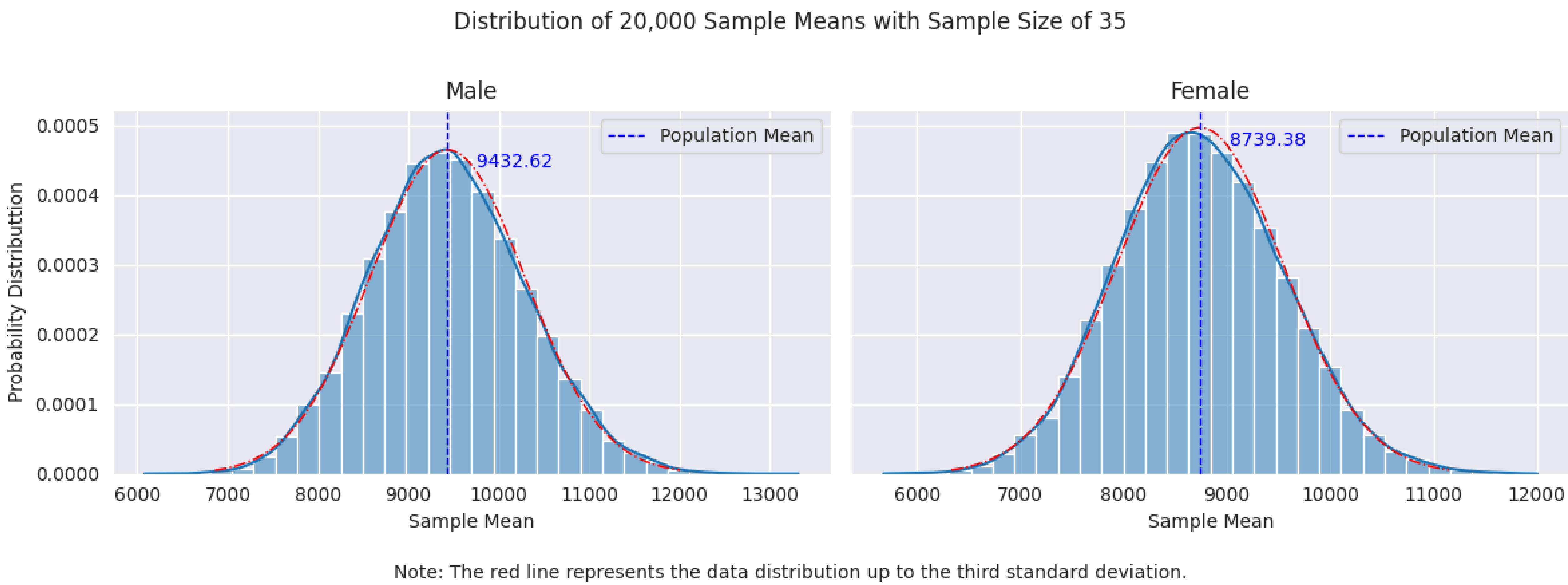
4. **Comparing Confidence Intervals:** We will compare the overlap within each category for different confidence intervals.

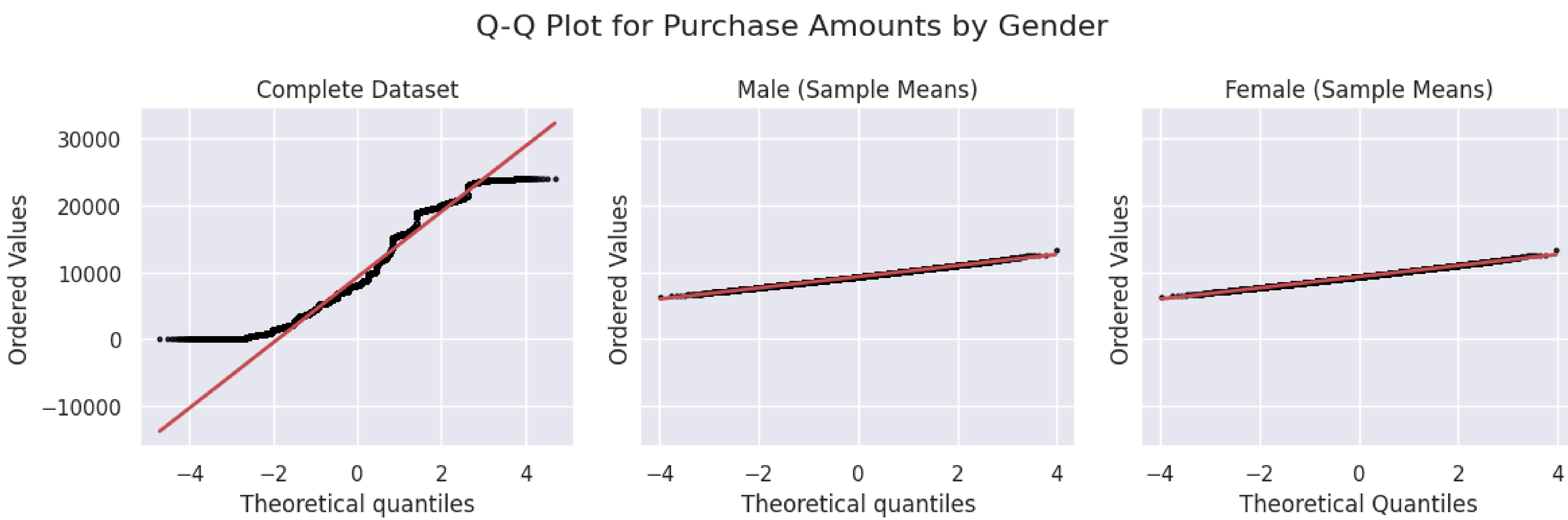
Common Functions

Show code

Profiling Based on **Gender**

1. Demonstrating Central Limit Theorem





Q-Q Plot for Purchase Amounts by Gender
Complete Dataset
Male (Sample Means)
Female (Sample Means)
Ordered Values
30000
20000
10000
0
−10000
−4
−2
0
2
4
Theoretical quantiles
Theoretical Quantiles

2. Testing Hypothesis

- **Null Hypothesis (H_0)** $-$ Women and men spend the same amount of money per transaction on average, i.e., $\mu_{\text{women}} = \mu_{\text{men}}$
- **Alternative Hypothesis (H_a)** $-$ Women spend more money per transaction than men on average, i.e., $\mu_{\text{women}} \neq \mu_{\text{men}}$

Significance Level $(\alpha) - 0.05$

```
# Two-tailed test

male = walmart[walmart.gender == 'male']
female = walmart[walmart.gender == 'female']

alpha = 0.05

t_stat, p_value = stats.ttest_ind(male.purchase, female.purchase, alternative='two-sided')

# Output the results
print("T-statistic:", t_stat.round(2))
print("P-value:", p_value)

if p_value < alpha: print("Women and men don't spend the same amount of money per transaction on average.")
else: print("Women and men spend the same amount of money per transaction on average.")
```

```
T-statistic: 44.84
P-value: 0.0
Women and men don't spend the same amount of money per transaction on average.
```

- **Null Hypothesis (H_0)** — Women spend more money per transaction on average, i.e., $\mu_{\text{men}} > \mu_{\text{women}}$
- **Alternative Hypothesis (H_a)** — Women spend less money per transaction than men on average, i.e., $\mu_{\text{men}} < \mu_{\text{women}}$

Significance Level (α) — 0.05

```
# Left-tailed test

male = walmart[walmart.gender == 'male']
female = walmart[walmart.gender == 'female']

alpha = 0.05

t_stat, p_value = stats.ttest_ind(male.purchase, female.purchase, alternative='less')

# Output the results
print("T-statistic:", t_stat.round(2))
print("P-value:", p_value)

if p_value < alpha: print("Women spend less money per transaction than men on average.")
else: print("Women spend more money per transaction on average.")
```

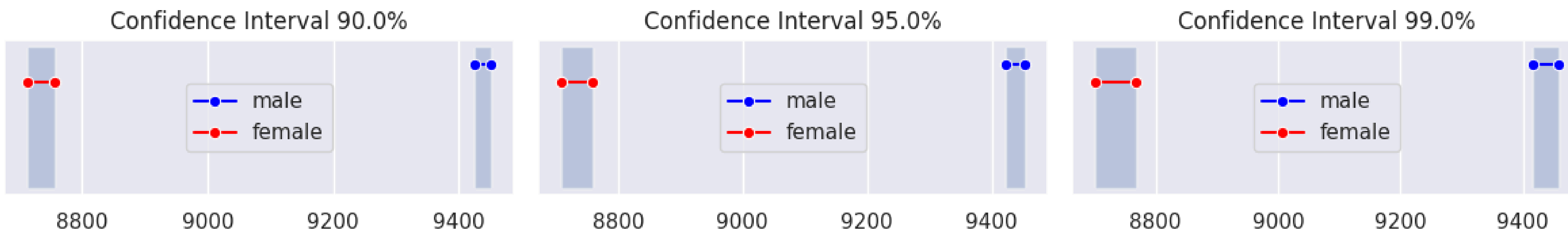
```
T-statistic: 44.84
P-value: 1.0
Women spend more money per transaction on average.
```

3. Calculating Confidence Intervals

```
# Calculated in Step 1.
means = {
    'male': np.mean(male.purchase),
    'female': np.mean(female.purchase)
}

# Calculate and print confidence intervals for Male
print("Confidence Intervals for Male:")
for conf in confidence_levels:
    ci = calc_conf_interval(male.purchase, conf, means['male'])
    print(f"{int(conf*100)}% Confidence Interval: {ci}")

# Calculate and print confidence intervals for Female
print("\nConfidence Intervals for Female:")
for conf in confidence_levels:
    ci = calc_conf_interval(female.purchase, conf, means['female'])
    print(f"{int(conf*100)}% Confidence Interval: {ci}")
```

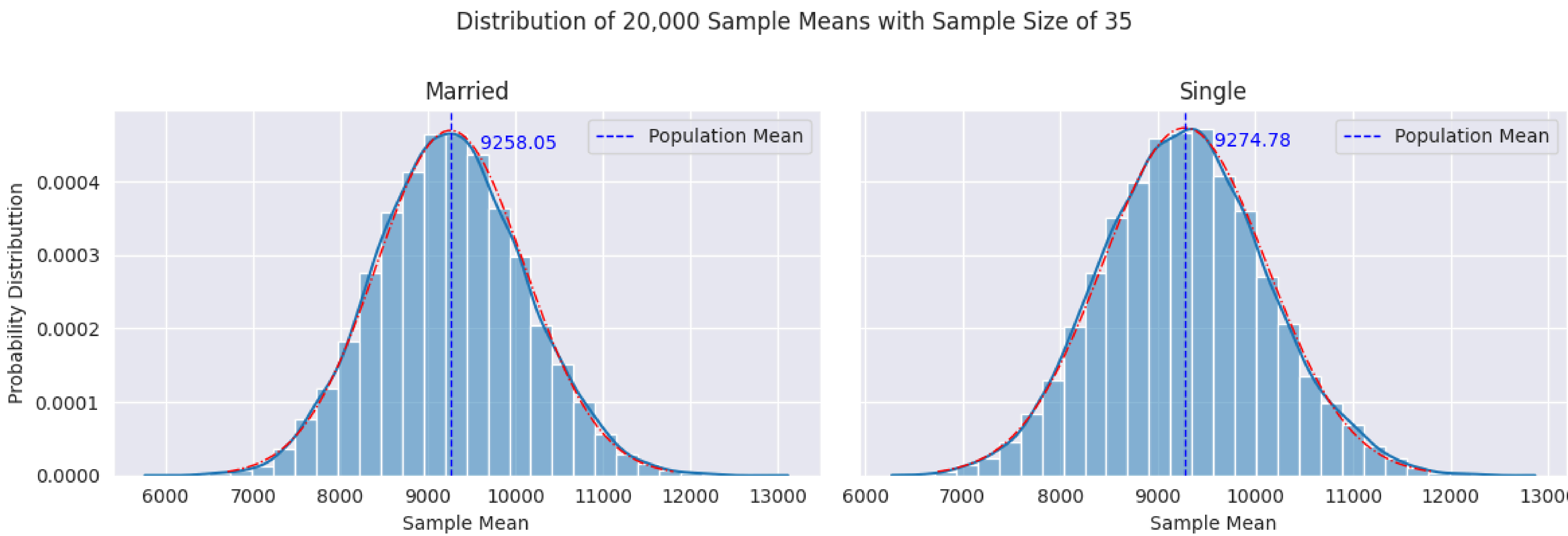
```
Confidence Intervals for Male:
90% Confidence Interval: (9424.51, 9450.54)
95% Confidence Interval: (9422.02, 9453.03)
99% Confidence Interval: (9417.15, 9457.91)

Confidence Intervals for Female:
90% Confidence Interval: (8713.29, 8755.84)
95% Confidence Interval: (8709.21, 8759.92)
99% Confidence Interval: (8701.24, 8767.89)
```

4. Comparing Confidence Intervals

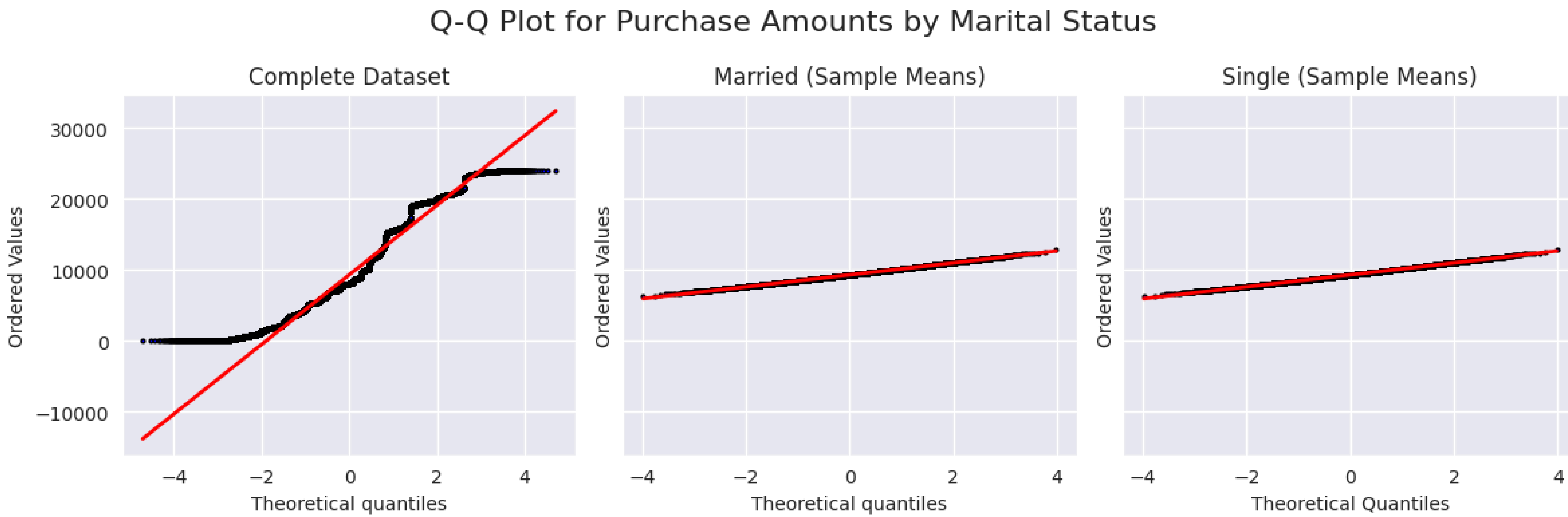


Profiling Based on **Marital Status**

1. Demonstrating Central Limit Theorem



Note: The red line represents the data distribution up to the third standard deviation.


Q-Q Plot for Purchase Amounts by Marital Status
Complete Dataset
Married (Sample Means)
Single (Sample Means)
Ordered Values
Theoretical quantiles
Theoretical Quantiles
30000
20000
10000
0
−10000
−4
−2
0
2
4

2. Testing Hypothesis

- **Null Hypothesis (H_0)** — Married individuals and single individuals spend the same amount of money per transaction on average, i.e., $\mu_{\text{married}} = \mu_{\text{single}}$
- **Alternative Hypothesis (H_a)** — Married individuals spend more money per transaction than single individuals on average, i.e., $\mu_{\text{married}} > \mu_{\text{single}}$

Significance Level (α) $- 0.05$

```
married = walmart[walmart.marital_status == 'married']
single = walmart[walmart.marital_status == 'single']

alpha = 0.05

t_stat, p_value = stats.ttest_ind(married.purchase, single.purchase, alternative='less')

# Output the results
print("T-statistic:", t_stat)
print("P-value:", p_value)

if p_value < alpha: print('Married individuals spend more money per transaction than single individuals on average.')
else: print("Married individuals and single individuals spend the same amount of money per transaction on average.")
```

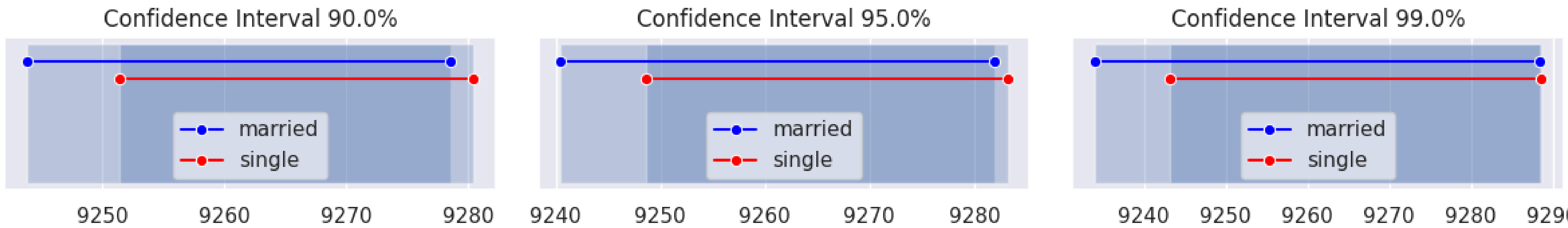
```
T-statistic: -0.3436698055440526
P-value: 0.3655473762879158
Married individuals and single individuals spend the same amount of money per transaction on average.
```

3. Calculating Confidence Intervals

```
# Calculated in Step 1.
means = {
    'married': np.mean(married.purchase),
    'single': np.mean(single.purchase)
}


# Calculate and print confidence intervals for married
print("Confidence Intervals for Married:")
for conf in confidence_levels:
    ci = calc_conf_interval(married.purchase, conf, means['married'])
    print(f"{int(conf*100)}% Confidence Interval: {ci}")

# Calculate and print confidence intervals for single
print("\nConfidence Intervals for Single:")
for conf in confidence_levels:
    ci = calc_conf_interval(single.purchase, conf, means['single'])
    print(f"{int(conf*100)}% Confidence Interval: {ci}")
```

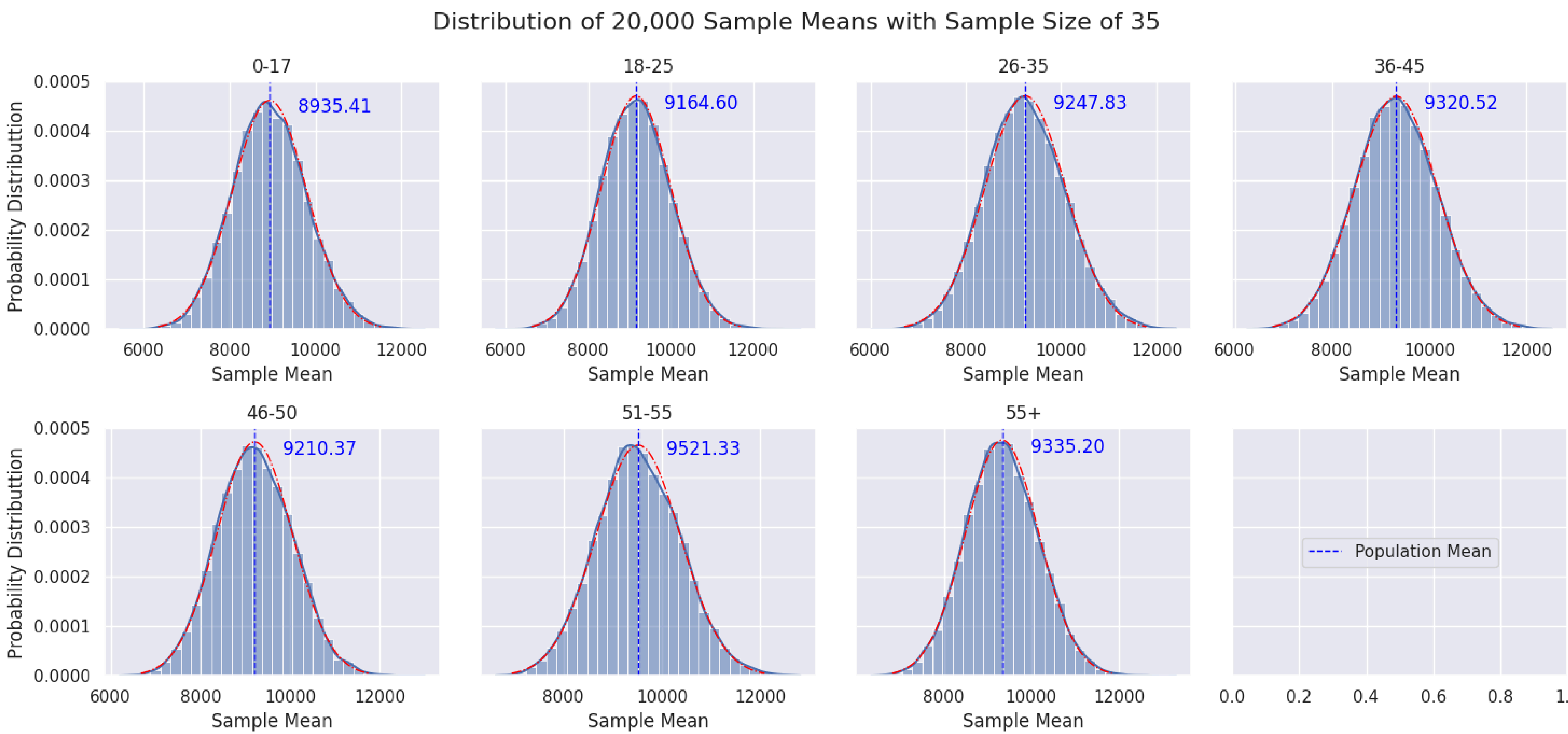
```
Confidence Intervals for Married:
90% Confidence Interval: (9243.79, 9278.56)
95% Confidence Interval: (9240.46, 9281.89)
99% Confidence Interval: (9233.95, 9288.4)

Confidence Intervals for Single:
90% Confidence Interval: (9251.4, 9280.42)
95% Confidence Interval: (9248.62, 9283.2)
99% Confidence Interval: (9243.18, 9288.63)
```

4. Comparing Confidence Intervals

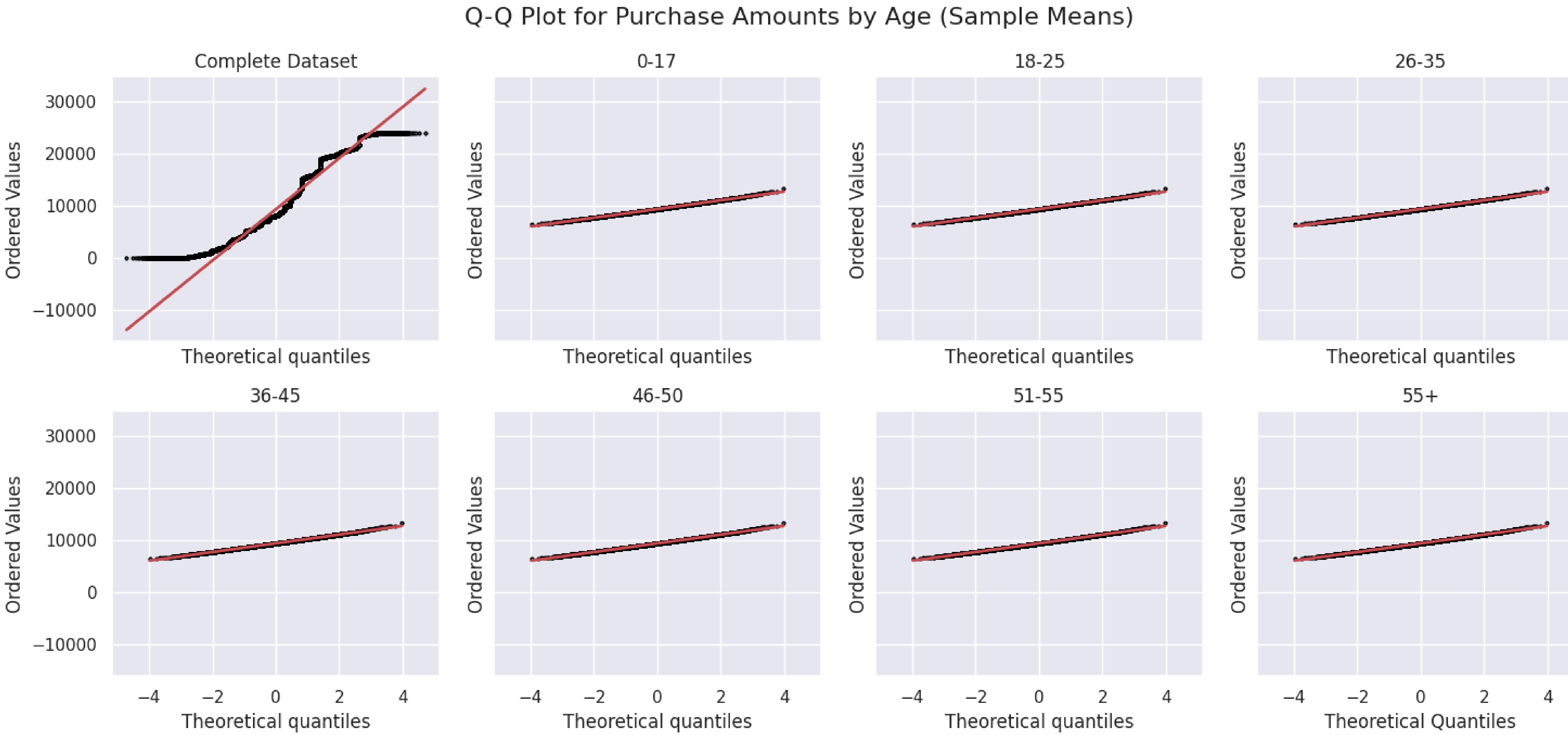


Profiling Based on **Age**

1. Demonstrating Central Limit Theorem



Note: The red line represents the data distribution up to the third standard deviation.


Q-Q Plot for Purchase Amounts by Age (Sample Means)
Complete Dataset
0-17
18-25
26-35
36-45
46-50
51-55
55+
Ordered Values
Theoretical quantiles
Theoretical Quantiles
30000
20000
10000
0
−10000
−4
−2
0
2
4

2. Testing Hypothesis

- **Null Hypothesis (H_0)** — All age groups spend the same amount of money per transaction on average.
- **Alternative Hypothesis (H_a)** — There is a significant association between age categories and purchase categories.

Significance Level $(\alpha) - 0.05$

```
# Null Hypothesis (H0): The data from the age category comes from a normal distribution
# Alternative Hypothesis (Ha):The data from the age category does not come from a normal distribution

# Perform Kolmogorov-Smirnov test for the entire numerical column
stat, p_value = stats.kstest(walmart.purchase, 'norm', args=(walmart.purchase.mean(), walmart.purchase.std()))
print("\nKolmogorov-Smirnov Test for the entire dataset:")
print("Statistic:", stat)
print("P-value:", p_value)

if p_value < alpha: print("The data from the age category does not come from a normal distribution")
else: print("The data from the age category comes from a normal distribution")
```

```
Kolmogorov-Smirnov Test for the entire dataset:
Statistic: 0.12964936863633847
P-value: 0.0
The data from the age category does not come from a normal distribution
```

```
# Null Hypothesis: Levene's test assumes that the variances of the groups are equal (homoscedasticity).
# Alternative Hypothesis: The variances of the groups are not equal (heteroscedasticity).

# Test for Homogeneity of Variances using Levene's Test
levene_stat, levene_p_value = stats.levene(
    walmart[walmart.age == '0-17'].purchase,
    walmart[walmart.age == '18-25'].purchase,
    walmart[walmart.age == '26-35'].purchase,
    walmart[walmart.age == '36-45'].purchase,
    walmart[walmart.age == '46-50'].purchase,
    walmart[walmart.age == '51-55'].purchase,
    walmart[walmart.age == '55+'].purchase
)
```

```
print("\nLevene's Test for Homogeneity of Variances:")
print("Statistic:", levene_stat)
print("P-value:", p_value)

if p_value < alpha: print("The variances of the groups are not equal (heteroscedasticity).")
else: print("Levene's test assumes that the variances of the groups are equal (homoscedasticity).")
```

```
Levene's Test for Homogeneity of Variances:
Statistic: 11.500322127282507
P-value: 0.0
The variances of the groups are not equal (heteroscedasticity).
```

- Since the Normality test and Levene's test failed we cannot apply ANOVA.
- Now, we'll apply Non-parametric test namely, $Kruskal - Wallis$.

```
# Null Hypothesis: There is no significant association between age groups and purchase.
# Alternative Hypothesis: There is a significant association between age groups and purchase.



# Perform Kruskal-Wallis
kruskal_stat, p_value = stats.kruskal(
    *[walmart[walmart.age == category].purchase for category in walmart.age.unique()]
)

print("Kruskal-Wallis statistic:", kruskal_stat)
print("P-value:", p_value)

if p_value < alpha: print("There is a significant association between age groups and purchase.")
else: print("There is no significant association between age groups and purchase.")
```

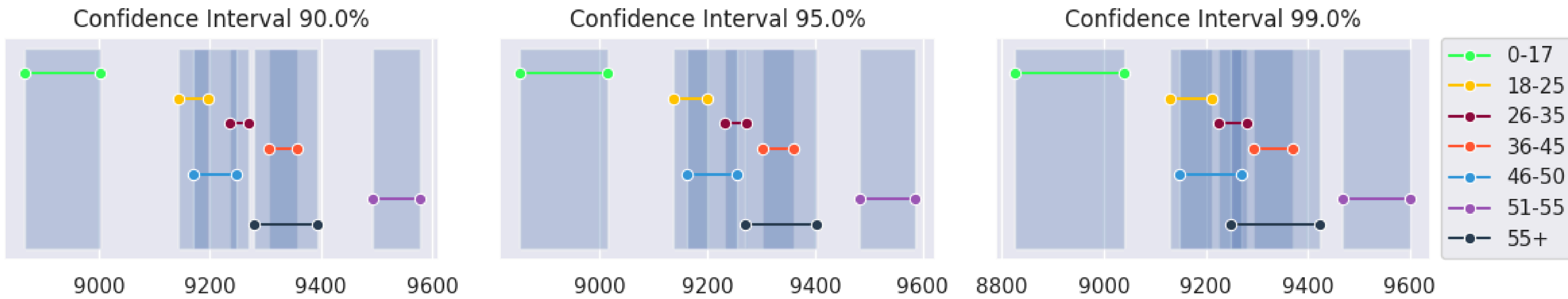
```
Kruskal-Wallis statistic: 315.65242682849174
P-value: 3.612251655399266e-65
There is a significant association between age groups and purchase.
```

3. Calculating Confidence Intervals

```
# List of age categories
age_categories = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

# Calculate and print confidence intervals for each age category
print("\nConfidence Intervals for Age Categories:")
for category in age_categories:
    data = walmart[walmart.age == category].purchase
    mean = np.mean(data)
    print(f"{category} –", end = "")
    for conf in confidence_levels:
        ci = calc_conf_interval(data, conf, mean)
        print(f" {int(conf*100)}% CI: {ci}", end = " |")
    print()
```

```
Confidence Intervals for Age Categories:
0-17 – 90% CI: (8865.05, 9001.88) | 95% CI: (8851.94, 9014.99) | 99% CI: (8826.32, 9040.61) |
18-25 – 90% CI: (9143.43, 9195.89) | 95% CI: (9138.41, 9200.92) | 99% CI: (9128.59, 9210.74) |
26-35 – 90% CI: (9235.1, 9270.28) | 95% CI: (9231.73, 9273.65) | 99% CI: (9225.15, 9280.23) |
36-45 – 90% CI: (9306.44, 9356.26) | 95% CI: (9301.67, 9361.03) | 99% CI: (9292.34, 9370.36) |
46-50 – 90% CI: (9170.41, 9246.85) | 95% CI: (9163.08, 9254.17) | 99% CI: (9148.77, 9268.48) |
51-55 – 90% CI: (9492.16, 9577.46) | 95% CI: (9483.99, 9585.63) | 99% CI: (9468.02, 9601.6) |
55+ – 90% CI: (9280.07, 9392.5) | 95% CI: (9269.3, 9403.27) | 99% CI: (9248.24, 9424.32) |
```

4. Comparing Confidence Intervals



Business Insights

Analytical Results —

1. 55% of the population has been staying in the current city for 1 to 3 years.
2. 72% of the population is in the age range of 18 to 45.
3. 38% of the total revenue is from purchase amounts between $5,000$ and $10,000$, totaling exactly 1.96 billion.
4. The top 20 products account for 445.9 million, which is 8.75% of the total revenue.
5. The top product category accounts for 1.91 billion, which is 37.48% of the total revenue.
6. The top 20 customers account for 131.68 million, which is 2.58% of the total revenue.
7. The top occupation category accounts for 666 million, which is 13.07% of the total revenue.

Results from Hypothesis Testing —

1. "From gender-based hypothesis testing, we find that **women spend more money per transaction than men on average**. However, the data is heavily male-dominated, so the result may seem counterintuitive.
2. From marital status-based hypothesis testing, we find that **married individuals and single individuals spend the same amount of money per transaction on average**.
3. From age-based hypothesis testing, we find a **significant association between age groups and purchase amounts**. Individuals from the age group 51 to 55 spend more money than any other age group in a single purchase, teenagers spend the least, and for the rest of the age groups, $9,250$ is the sweet spot.

Recommendations

1. **Marketing Strategy** —
    - Product displays for less popular products can help customers notice these items.
    - Offering sales/discounts for less popular products.
2. **Customer Retention** —
    - Organize lucky draw offers for all customers from time to time.
    - Provide special discount offers for the top 20 to 50 customers.
3. **Product Range** —
    - For less popular product categories, offering the same type of product from different companies can give customers options and can boost product category sales.