walmart_solution

August 1, 2022

1 Business Case: Walmart - Confidence Interval and CLT

1.1 About Walmart

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.

1.2 Business Problem

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).

1.3 Dataset

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

- 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

1.4 What good looks like?

1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.
2. Detect Null values & Outliers (using boxplot, "describe" method by checking the difference between mean and median, isnull etc.)
3. Do some data exploration steps like:

- Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male customers, calculate the average, and conclude the results.
- Inference after computing the average female and male expenses.
- Use the sample average to find out an interval within which the population average will lie. Using the sample of female customers you will calculate the interval within which the average spending of 50 million male and female customers may lie.

4. Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers.
   - The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.
5. Conclude the results and check if the confidence intervals of average male and female spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?
6. Perform the same activity for Married vs Unmarried and Age
   - For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.
7. Give recommendations and action items to Walmart.

1.5 Solution

1.5.1 Read data and analyze basic metrics

[194]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('data/walmart_data.txt')
df_original = df.copy()

df.head()
```

[194]:

	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

[195]:
```
df.shape
```

[195]: (550068, 10)

[196]:
```
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
```

[197]:
```
for col in df.columns:
    print(f'{col}: {df[col].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
```

[198]:
```
for col in ['Gender', 'Age', 'Occupation', 'City_Category',␣
↪'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']:
    print(df[col].value_counts())
    print('\n')
```

```
M    414259
F    135809
Name: Gender, dtype: int64


26-35    219587
36-45    110013
```

```
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64


4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64


B    231173
C    171175
A    147720
Name: City_Category, dtype: int64


1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64


0    324731
1    225337
```

```
Name: Marital_Status, dtype: int64
```

```
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
Name: Product_Category, dtype: int64
```

Observations

1. Total number of records = 550068. Unique number of users = 5891. unique number of products = 3631
2. *Purchase* is a continuous variable
3. *User_ID, Product_ID, Occupation, Product_Category* are nominal categorical variables.
4. *Gender, Marital_Status* are dichotomous variables.
5. *Age* (binned) and *Stay_In_Current_City_Years* can be considered ordinal categorical variables.
6. *City_Category*, in the absence of any additional details, can be considered a nominal categorical variable.

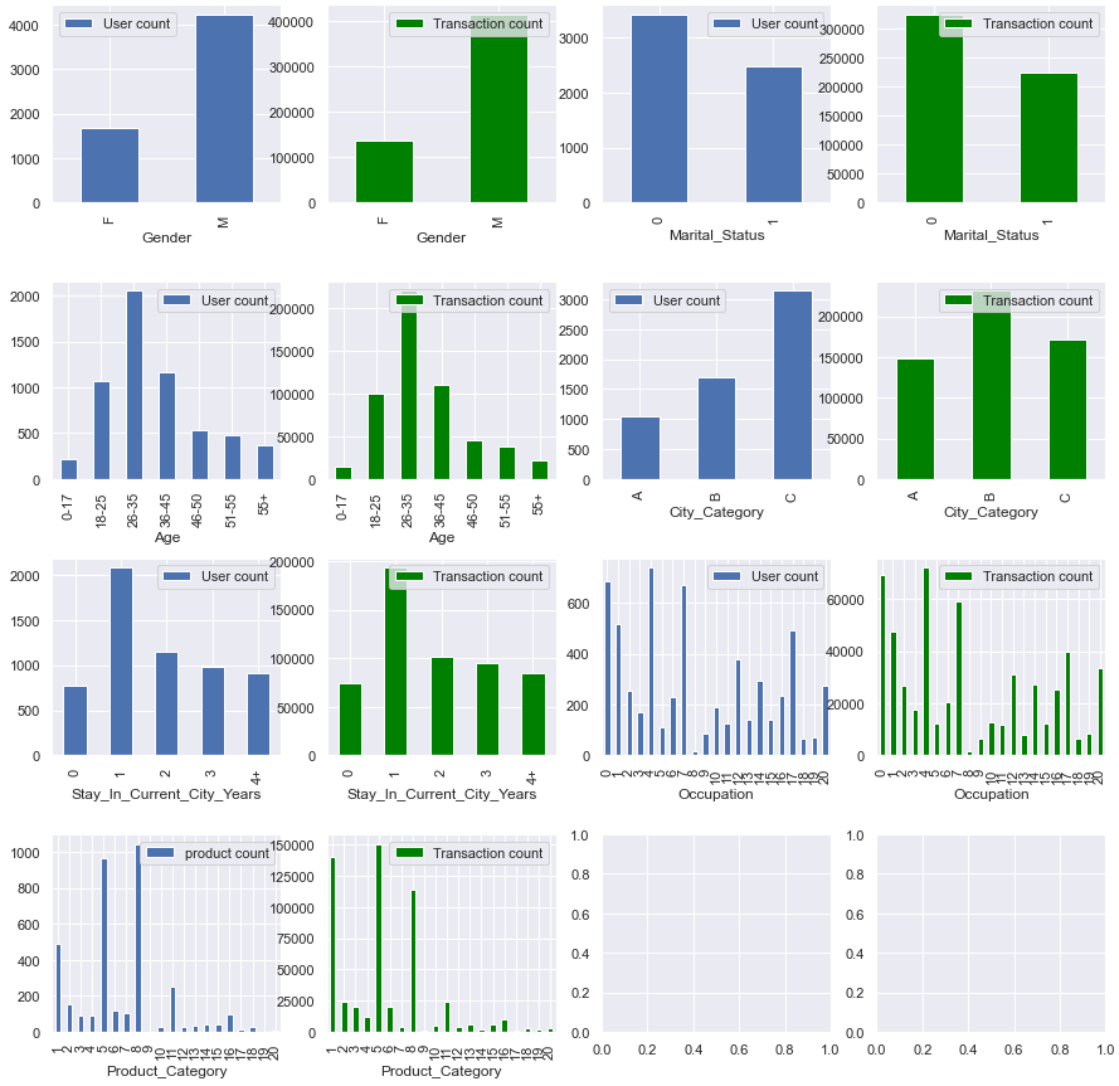
1.6 Univariate analysis

1.6.1 Categorical variables

Unique user/product count per variable

[199]:
```
#univariate analysis for categorical variables
```

```
fig, ax = plt.subplots(4, 4, figsize=(15, 15))

df.groupby('Gender')['User_ID'].nunique().to_frame().rename(columns={'User_ID':␣
↪'User count'}).plot(kind='bar', ax= ax[0][0])
df.groupby('Gender')['User_ID'].count().to_frame().rename(columns={'User_ID':␣
↪'Transaction count'}).plot(kind='bar',color='green', ax= ax[0][1])
df.groupby('Marital_Status')['User_ID'].nunique().to_frame().
↪rename(columns={'User_ID': 'User count'}).plot(kind='bar', ax= ax[0][2])
df.groupby('Marital_Status')['User_ID'].count().to_frame().
↪rename(columns={'User_ID': 'Transaction count'}).
↪plot(kind='bar',color='green', ax= ax[0][3])

df.groupby('Age')['User_ID'].nunique().to_frame().rename(columns={'User_ID':␣
↪'User count'}).plot(kind='bar', ax= ax[1][0])
df.groupby('Age')['User_ID'].count().to_frame().rename(columns={'User_ID':␣
↪'Transaction count'}).plot(kind='bar',color='green', ax= ax[1][1])
df.groupby('City_Category')['User_ID'].nunique().to_frame().
↪rename(columns={'User_ID': 'User count'}).plot(kind='bar', ax= ax[1][2])
df.groupby('City_Category')['User_ID'].count().to_frame().
↪rename(columns={'User_ID': 'Transaction count'}).
↪plot(kind='bar',color='green', ax= ax[1][3])

df.groupby('Stay_In_Current_City_Years')['User_ID'].nunique().to_frame().
↪rename(columns={'User_ID': 'User count'}).plot(kind='bar',ax= ax[2][0])
df.groupby('Stay_In_Current_City_Years')['User_ID'].count().to_frame().
↪rename(columns={'User_ID': 'Transaction count'}).
↪plot(kind='bar',color='green',ax= ax[2][1])
df.groupby('Occupation')['User_ID'].nunique().to_frame().
↪rename(columns={'User_ID': 'User count'}).plot(kind='bar', ax= ax[2][2])
df.groupby('Occupation')['User_ID'].count().to_frame().
↪rename(columns={'User_ID': 'Transaction count'}).
↪plot(kind='bar',color='green', ax= ax[2][3])

df.groupby('Product_Category')['Product_ID'].nunique().to_frame().
↪rename(columns={'Product_ID': 'product count'}).plot(kind='bar', ax=␣
↪ax[3][0])
df.groupby('Product_Category')['Product_ID'].count().to_frame().
↪rename(columns={'Product_ID': 'Transaction count'}).plot(kind='bar',␣
↪color='green',ax= ax[3][1])

plt.subplots_adjust(hspace=0.4)
plt.show()
```



Observations

1. Number of unique male customers (4225) are more than twice the number of unique female customers (1666).
2. Total number of transactions by male customers (414259) are more than thrice the number of transactions by female customers (135809)
3. Number of unique customers with Marital Status zero (3417) are more than the number of customers with Martial Status one (2474).
4. Total number of transactions by customers with Marital Status zero (324731) are more than the number of transactions by customers with Marital Status one (225337)
5. Unique customers in the age bracket 26-35 (2053) are highest followed by those in age brackets 36-45 (1167) and 18-25 (1069).

6. Transactions by customers in the age bracket 26-35 (219587) are highest followed by those in age brackets 36-45 (110013) and 18-25 (99660).
7. Unique customers belonging to city_category C (3139) are highest, followed by B (1707) and A (1045)
8. Transactions by customers belonging to city_category B (231173) are highest, followed by C (171175) and A (147720). This shows that customers from city category B has the highest number of average transactions per user.
9. Number of unique customers as well as total number of transations are highest for consumers staying in the city for 1 year followed by 2 and 3 years.
10. Number of unique customers as well as total number of transations are highest for consumers with occupation 4 followed by 0 and 7.
11. The highest number of unique products are from category 8, 5 and 1.
12. The highest number of transactions are executed for products from category 5, 1 and 8

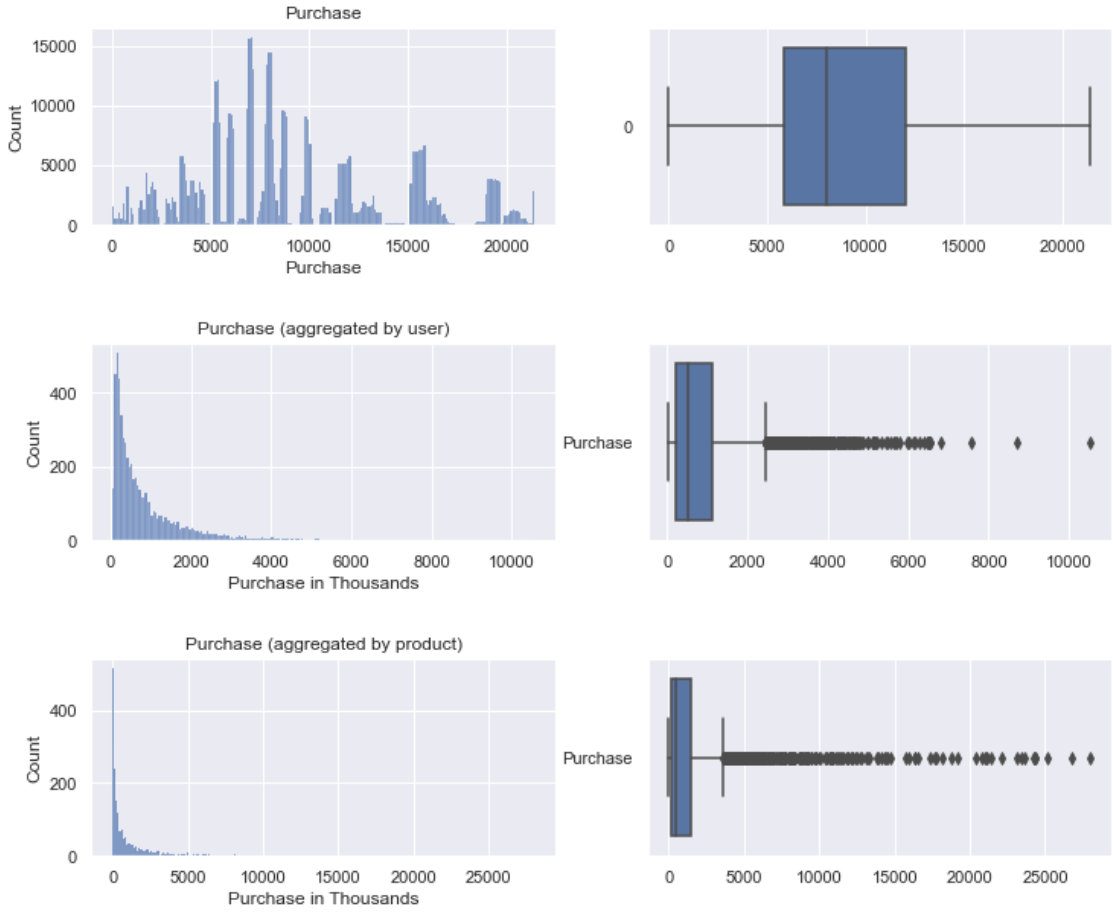
1.6.2 Continuous variable - Purchase

[200]:
```
df['Purchase'].describe()
```

[200]:
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

[246]:
```
#univariate analysis for purchase variable

fig, ax = plt.subplots(3, 2, figsize=(12, 10))

ax[0][0].title.set_text('Purchase')
sns.histplot(data=df['Purchase'], binwidth=100, ax=ax[0][0])
sns.boxplot(data=df['Purchase'], orient="horizontal", ax=ax[0][1])

ax[1][0].title.set_text('Purchase (aggregated by user)')
data = df.groupby('User_ID')['Purchase'].sum().div(1000)
p1 = sns.histplot(data=data, binwidth=50, ax=ax[1][0])
p1.set(xlabel = "Purchase in Thousands")
sns.boxplot(data=data.to_frame(), orient="horizontal", ax=ax[1][1])
purchase_by_user_skew = data.skew()

ax[2][0].title.set_text('Purchase (aggregated by product)')
```

```
data = df.groupby('Product_ID')['Purchase'].sum().div(1000)
p2 = sns.histplot(data=df.groupby('Product_ID')['Purchase'].sum().div(1000),␣
 ↪binwidth=50,  ax=ax[2][0])
p2.set(xlabel = "Purchase in Thousands")
sns.boxplot(data=data.to_frame(), orient="horizontal", ax=ax[2][1])
purchase_by_prod_skew = data.skew()

plt.subplots_adjust(hspace=0.6)

plt.show()

print(f'skew for Purchase by user = {purchase_by_user_skew}')
print(f'skew for Purchase by prod = {purchase_by_prod_skew}')
```




```
skew for Purchase by user = 2.428250092257213
skew for Purchase by prod = 4.43956792557916
```

Observations

1. The distribution of Purchase variable has several peaks and it doesn't follow normal distribu-

tion. The median Purchase value is 8047. It has several outlers (2677). See outliers section for more details.

2. The distributions of total Purchase by user is positively skewed (skew = 2.43)
3. The distributions of total Purchase by product is positively skewed (skew = 4.44)

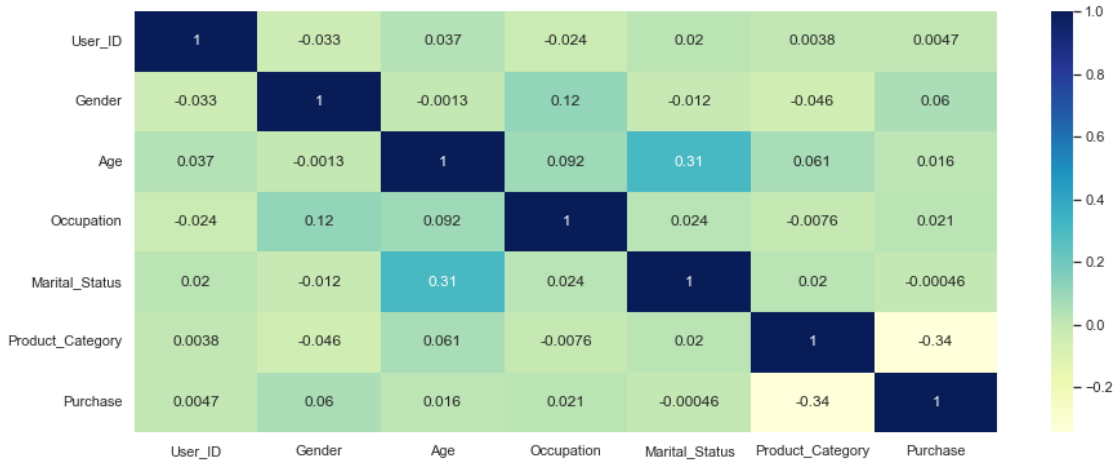
1.7 Correlation between Purchase and various variables

In the given dataset, Purchase is the only continous variable. We can attempt to compute correlation of Purchase with various dichotomous variables (Gender, Marital_Status) and ordinal categorical variables (Age and Stay_In_Current_City_Years).

[202]:
```
df_copy = df.copy()

df_copy['Gender'].replace(['M', 'F'], [1, 0], inplace=True)
df_copy['Marital_Status'].replace(['Single', 'Partnered'], [0, 1], inplace=True)
df_copy['Age'].replace(['0-17', '18-25', '26-35', '36-45',␣
 ↪'46-50','51-55','55+'], [9, 22, 31, 41, 48, 53, 65], inplace=True)
df_copy['Stay_In_Current_City_Years'].replace(['1', '2', '3', '4', '5+'], [1,␣
 ↪2, 3, 4, 5], inplace=True)

df_copy.corr()
```

[202]:
```
                    User_ID    Gender       Age  Occupation  Marital_Status  \
User_ID            1.000000 -0.033474  0.036640   -0.023971        0.020443
Gender            -0.033474  1.000000 -0.001272    0.117291       -0.011603
Age                0.036640 -0.001272  1.000000    0.091689        0.308139
Occupation        -0.023971  0.117291  0.091689    1.000000        0.024280
Marital_Status     0.020443 -0.011603  0.308139    0.024280        1.000000
Product_Category   0.003825 -0.045594  0.060828   -0.007618        0.019888
Purchase           0.004716  0.060346  0.015837    0.020833       -0.000463

                   Product_Category  Purchase
User_ID                    0.003825  0.004716
Gender                    -0.045594  0.060346
Age                        0.060828  0.015837
Occupation                -0.007618  0.020833
Marital_Status             0.019888 -0.000463
Product_Category           1.000000 -0.343703
Purchase                  -0.343703  1.000000
```

[203]:
```
plt.figure(figsize=(15,6))
sns.heatmap(df_copy.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



Observations

We do not see correlation between any pair of variables.

1.8 Handling missing values and outliers

[204]:
```
#detecting null/missing values
df.isnull().sum()
```

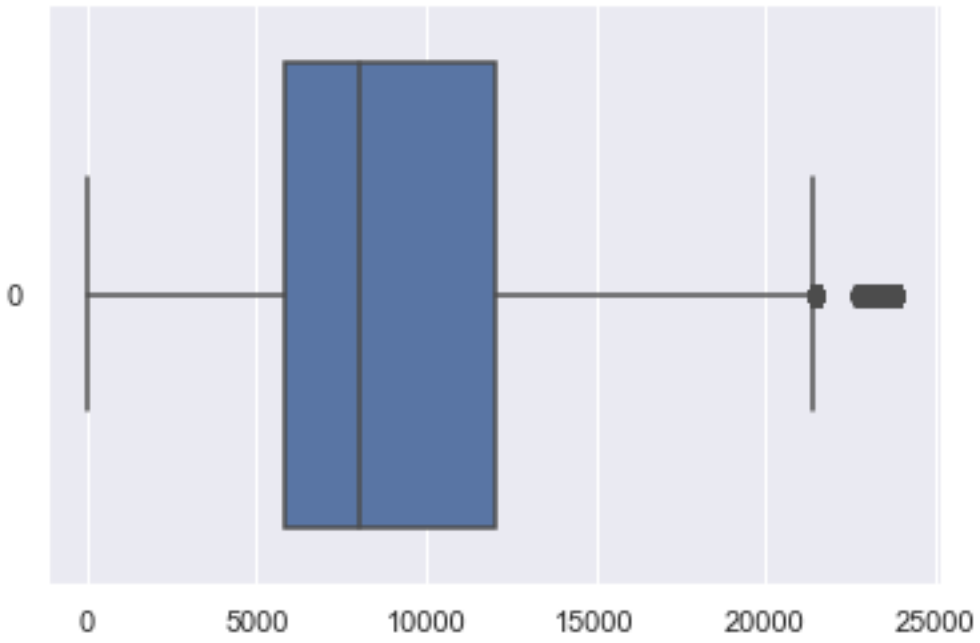
[204]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category            0
Purchase                    0
dtype: int64
```

Observation

No missing values.

[205]:
```
#outlier detection for purchase
sns.boxplot(data=df['Purchase'], orient='horizontal')
```

[205]:
```
<AxesSubplot:>
```



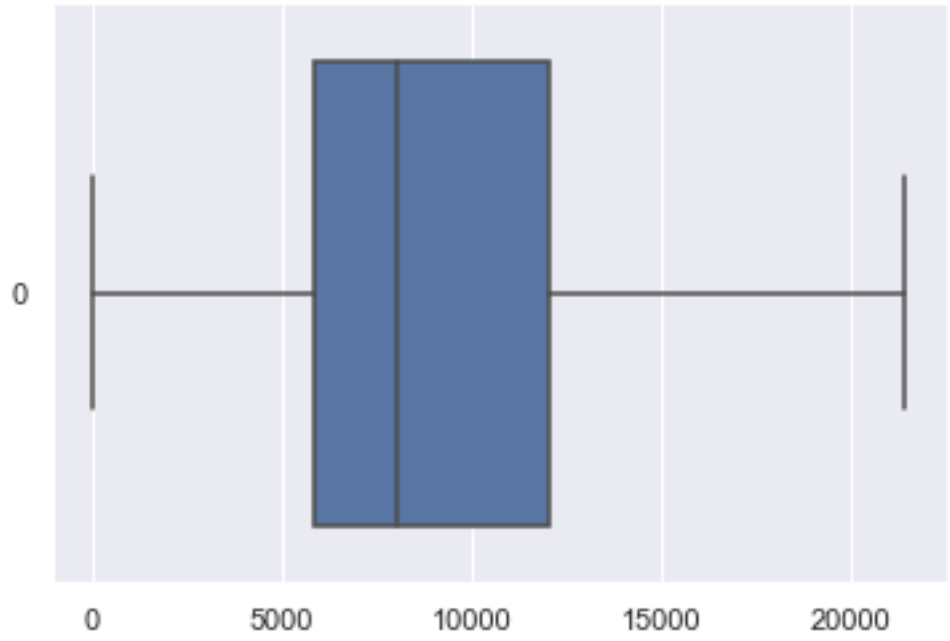
[206]:
```
#outlier values handling

max_purchase = df_original['Purchase'].max()
min_purchase = df_original['Purchase'].min()

p25, p50, p75 = df_original['Purchase'].quantile([0.25, 0.5, 0.75]).values
max_box_val = p75 + (p75-p25) * 1.5
no_outliers = (df_original['Purchase'] > max_box_val).sum()

print(f'maximum purchase value is: {max_purchase}')
print(f"Box plot's upper limit is {max_box_val}. The number of outliers =␣
↪{no_outliers}")

#We observe that outliers are NOT significantly more than maximum box plot␣
↪value.
#It should be safe to clip all the outliers to max box plot value.

df['Purchase'] = np.clip(df_original['Purchase'], min_purchase, max_box_val)
```

```
maximum purchase value is: 23961
Box plot's upper limit is 21400.5. The number of outliers = 2677
```

After replacing outliers with maximum Box plot values, we get the below box plot

[207]:
```
sns.boxplot(data=df['Purchase'], orient='horizontal')
```

[207]: <AxesSubplot:>



1.9 Data Exploration

In the given business problem, the population consists of 50 million male and female consumers. We are given a dataset consisting of around 550068 transactions (5891 unique users)**. From the given dataset, we create** samples of various sizes; n=5000, 10000, 20000, 100000, and 550068 (the full data set).**

For the combination of each **sample** and **categorical variable values** (such as 'M' and 'F' for gender, 0 and 1 for marital_status), we **estimate Purchase mean value** and also compute **Confidence interval** around the estimated value (**for confidence levels 90, 95 and 99**). We observe that the confidence interval gets narrower as the sample size increases. Also, as the confidence level increases, the confidence interval widens.

We compute confidence interval using two approaches. First, using the **standard formula**, and second, using the **bootstrapping** approach (using seaborn pointplot function).

1.9.1 Create samples and define helper functions

[208]:
```
sample_sizes = [5000, 10000, 20000, 100000, df.shape[0]]
samples = []
sample_num = 0
for size in sample_sizes:
    sample_num += 1
    sample = df.sample(n=size)
    sample['Sample'] = f'sample-{sample_num} ({size})'
```

```
    samples.append(sample)

combined_samples = pd.DataFrame([]).append(other=samples)

#define helper functions

# calculates confidence_interval
def confidence_interval(sample, cv):
    n = sample.size
    sample_mean = sample.mean()

    sample_var = ((sample - sample_mean)**2).sum() / (n-1)
    sample_std = sample_var**0.5
    standard_error = sample_std / n**0.5
    zscore = stats.norm.ppf(1 - (1-cv)/2)
    error_margin = zscore * standard_error

    return (sample_mean - error_margin, sample_mean + error_margin, sample_mean)

# returns CI data in convenient dataframe form
def getCIDataFrame(col, samples, clevels):
    ci_data = []
    for sample in samples:
        #group sample by categorical column
        sample_name = sample['Sample'].iloc[0]
        groups = sample.groupby(col)['Purchase'].groups

        #for each group, calculate CI
        for grp_key in groups.keys():
            grp_obj = groups[grp_key]
            grp_rows_indexes = grp_obj.values #indexes of the rows␣
↪corresponding to this group
            grp_rows = sample.loc[grp_rows_indexes]['Purchase']

            #for each CI level
            for clevel in clevels:
                ci_lower, ci_upper, ci_mean = confidence_interval(grp_rows,␣
↪clevel/100)
                ci_data.append([sample_name, grp_key, clevel, np.
↪round(ci_mean,2), np.round(ci_lower,2), np.round(ci_upper,2)])

    ci_df = pd.DataFrame(ci_data, columns=['sample', 'value',␣
↪'confidence-level', ' mean' ,'ci-lower', 'ci-upper'])
    ci_df = ci_df.pivot(index = ['sample', 'value'], columns =␣
↪['confidence-level'], values = [' mean' ,'ci-lower', 'ci-upper'])
    ci_df = ci_df.swaplevel(0,1,axis=1).sort_index(axis=1)
```

```
    return ci_df
```

1.9.2 Spending by Gender

Average spending per transaction by gender for the given dataset

[209]:
```
grps = df.groupby('Gender')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print(total / cnt)
```

```
Gender
F    8726.256327
M    9428.373455
dtype: float64
```

Observations

1. Average spending per transaction for female customers is less than that of male customers by around 7.4%.

Estimating average spending per transaction by Gender for samples and calculating CI

[210]:
```
# Calculating CI using formula
res_df = getCIDataFrame('Gender', samples, [90, 95, 99])

res_df
```

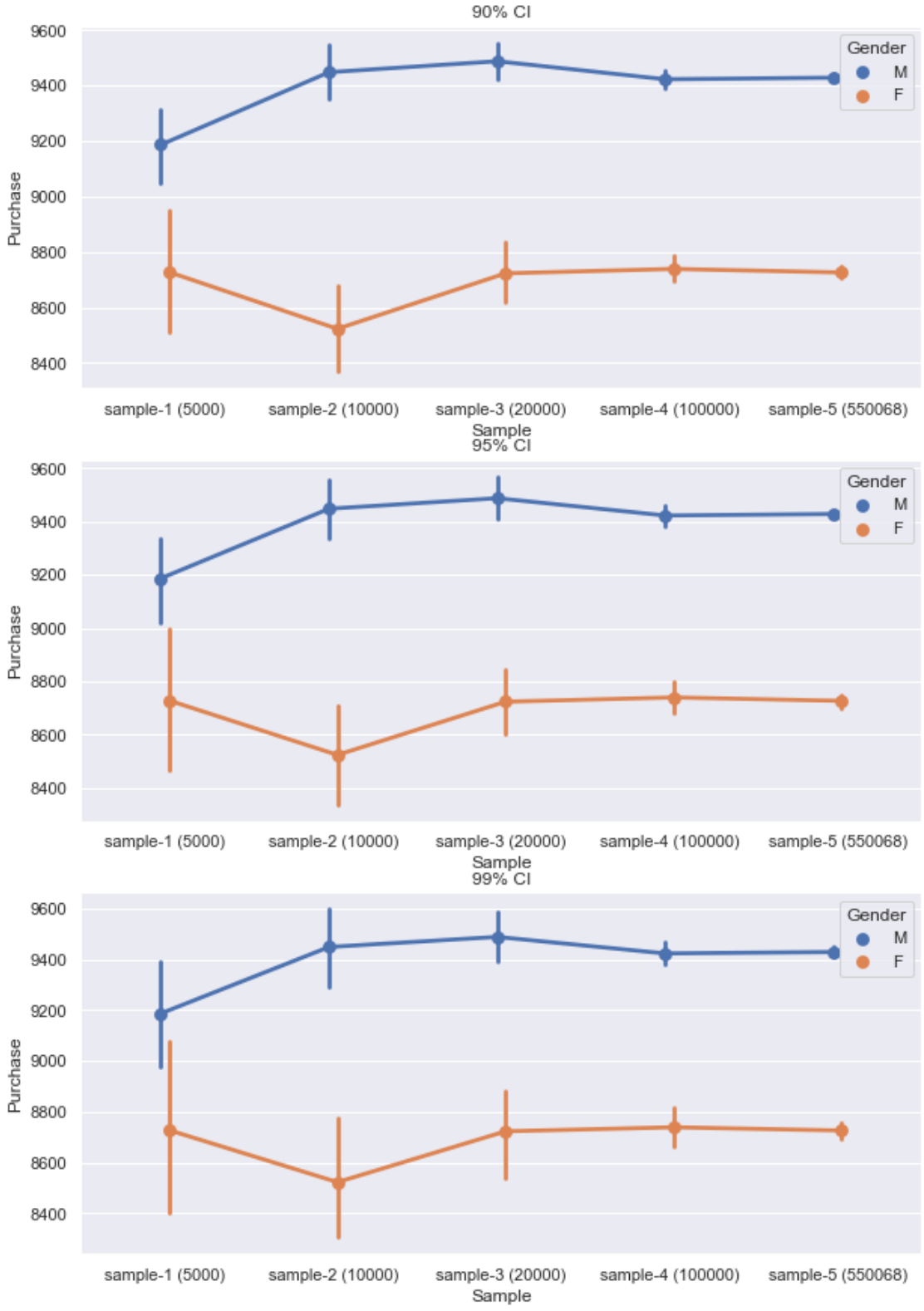
[210]:

confidence-level		90			95			\
		mean	ci-lower	ci-upper	mean	ci-lower	ci-upper	
sample	value							
sample-1 (5000)	F	8727.37	8509.47	8945.26	8727.37	8467.73	8987.01	
	M	9186.64	9053.03	9320.24	9186.64	9027.44	9345.83	
sample-2 (10000)	F	8523.13	8367.95	8678.31	8523.13	8338.22	8708.04	
	M	9447.99	9351.75	9544.23	9447.99	9333.31	9562.67	
sample-3 (20000)	F	8723.16	8614.23	8832.08	8723.16	8593.37	8852.95	
	M	9487.62	9419.41	9555.84	9487.62	9406.34	9568.91	
sample-4 (100000)	F	8739.10	8689.57	8788.64	8739.10	8680.08	8798.13	
	M	9422.65	9392.21	9453.08	9422.65	9386.38	9458.91	
sample-5 (550068)	F	8726.26	8705.09	8747.43	8726.26	8701.03	8751.48	
	M	9428.37	9415.42	9441.33	9428.37	9412.94	9443.81	

confidence-level		99		
		mean	ci-lower	ci-upper
sample	value			
sample-1 (5000)	F	8727.37	8386.14	9068.59
	M	9186.64	8977.41	9395.86
sample-2 (10000)	F	8523.13	8280.11	8766.15

```
                   M      9447.99  9297.27  9598.70
sample-3 (20000)   F      8723.16  8552.58  8893.73
                   M      9487.62  9380.80  9594.45
sample-4 (100000)  F      8739.10  8661.53  8816.67
                   M      9422.65  9374.99  9470.30
sample-5 (550068)  F      8726.26  8693.10  8759.41
                   M      9428.37  9408.09  9448.66
```

[211]:
```
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")

fig, ax = plt.subplots(3, 1, figsize=(10, 15))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="Gender", x='Sample', ci=90,␣
 ↪data=combined_samples, dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="Gender", x='Sample', ci=95,␣
 ↪data=combined_samples, dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="Gender", x='Sample', ci=99,␣
 ↪data=combined_samples, dodge=True, ax=ax[2])

plt.show()
```


90% CI
95% CI
99% CI
Gender
M
F
Purchase
9600
9400
9200
9000
8800
8600
8400
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample

Observations

1. Average spending per transaction for female customers is less than that of male customers by around 7.4%.
2. The estimated mean values and their corresponding **confidence intervals are non-overlapping** for male and female customers.
3. For the given confidence level, as the sample size increases, confidence interval gets narrower and we usually get better estimate of the mean value.
4. For the given sample size, as the confidence level increases, confidence interval widens.

Recommendations

1. Average spending per transaction for female customers is less than that of male customers by around 7.4%. There is a potential to incentivize female customers to spend more money per transaction. Business can consider running special promotions/offers for female constomers to encourage them to purchase more stuff in each transaction.

1.9.3 Spending by Marital Status

Average spending per transaction by Marital Status for the given dataset

[247]:
```
grps = df.groupby('Marital_Status')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print(total / cnt)

df['Purchase'].sum()/df['Purchase'].size
```

```
Marital_Status
0    9257.517959
1    9251.430702
dtype: float64
```

[247]:
```
9255.02429608703
```

Observations

1. Average spending per transaction for married/unmarried customers is very close.

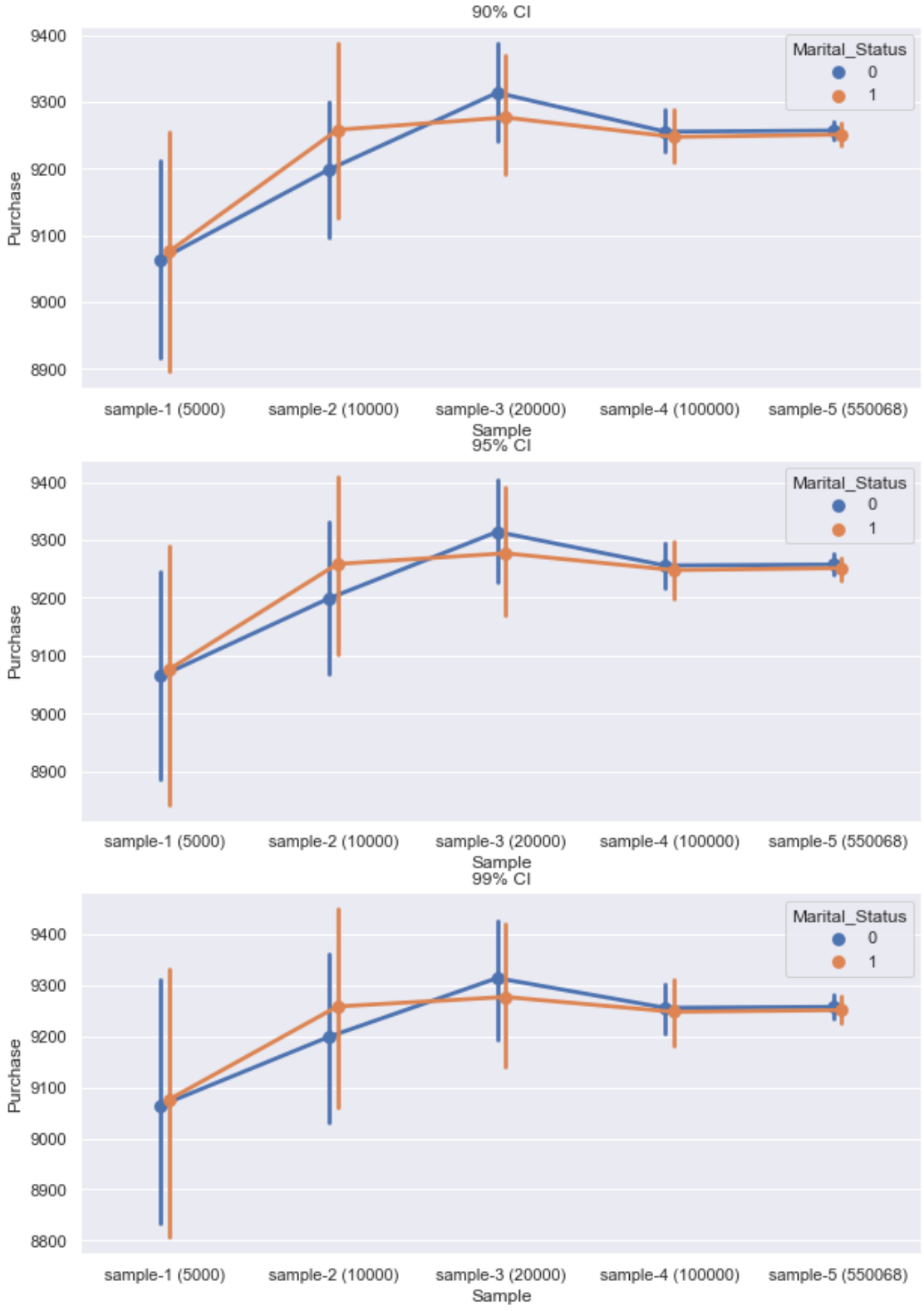
Estimating average spending by Marital Status for samples and calculating CI

[213]:
```
# Calculating CI using formula
res_df = getCIDataFrame('Marital_Status', samples, [90, 95, 99])

res_df
```

[213]:
```
confidence-level              90                           95                      \
                             mean ci-lower ci-upper     mean ci-lower ci-upper
sample            value
```

```
sample-1 (5000)   0      9064.47  8914.40  9214.54  9064.47  8885.66  9243.29
                  1      9076.33  8900.72  9251.93  9076.33  8867.08  9285.58
sample-2 (10000)  0      9199.03  9092.68  9305.37  9199.03  9072.31  9325.75
                  1      9258.51  9128.59  9388.42  9258.51  9103.71  9413.30
sample-3 (20000)  0      9314.11  9238.27  9389.95  9314.11  9223.74  9404.48
                  1      9276.99  9186.55  9367.43  9276.99  9169.22  9384.75
sample-4 (100000) 0      9255.82  9221.96  9289.68  9255.82  9215.47  9296.17
                  1      9247.94  9207.30  9288.58  9247.94  9199.52  9296.37
sample-5 (550068) 0      9257.52  9243.07  9271.97  9257.52  9240.30  9274.73
                  1      9251.43  9234.14  9268.73  9251.43  9230.82  9272.04

confidence-level             99
                           mean ci-lower ci-upper
sample            value
sample-1 (5000)   0      9064.47  8829.47  9299.48
                  1      9076.33  8801.33  9351.33
sample-2 (10000)  0      9199.03  9032.49  9365.56
                  1      9258.51  9055.06  9461.95
sample-3 (20000)  0      9314.11  9195.34  9432.88
                  1      9276.99  9135.36  9418.62
sample-4 (100000) 0      9255.82  9202.79  9308.85
                  1      9247.94  9184.30  9311.58
sample-5 (550068) 0      9257.52  9234.89  9280.14
                  1      9251.43  9224.35  9278.51
```

[214]:
```python
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")

fig, ax = plt.subplots(3, 1, figsize=(10, 15))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="Marital_Status", x='Sample', ci=90,␣
 ↪data=combined_samples, dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="Marital_Status", x='Sample', ci=95,␣
 ↪data=combined_samples, dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="Marital_Status", x='Sample', ci=99,␣
 ↪data=combined_samples, dodge=True, ax=ax[2])

plt.show()
```


90% CI
Marital_Status
0
1
Purchase
9400
9300
9200
9100
9000
8900
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample
95% CI
Marital_Status
0
1
Purchase
9400
9300
9200
9100
9000
8900
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample
99% CI
Marital_Status
0
1
Purchase
9400
9300
9200
9100
9000
8900
8800
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample

Observations

1. Average spending per transaction for married and unmarried customers very close.
2. The estimated mean values and their corresponding **confidence intervals are highly overlapping** for married and unmarried customers.
3. For the given confidence level, as the sample size increases, confidence interval gets narrower and we usually get better estimate of the mean value.
4. For the given sample size, as the confidence level increases, confidence interval widens.

Recommendations

1. Customer's marital status does not impact their average spending per transaction. The business therefore should provide equal focus to both married and unmarried customers in their strategy.

1.9.4 Spending by Age

Average spending per transaction by Age for the given dataset

[215]:
```
grps = df.groupby('Age')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print(total / cnt)

df['Purchase'].sum()/df['Purchase'].size
```

```
Age
0-17     8925.539597
18-25    9164.189554
26-35    9244.947060
36-45    9320.888550
46-50    9198.531093
51-55    9519.560427
55+      9319.768741
dtype: float64
```

[215]:
```
9255.02429608703
```

Estimating average spending per transaction by Age for samples and calculating CI

[225]:
```
# Calculating CI using formula
res_df = getCIDataFrame('Age', samples, [90, 95, 99])

res_df
```

[225]:
```
confidence-level              90                            95           \
                            mean ci-lower  ci-upper     mean ci-lower
sample            value
```

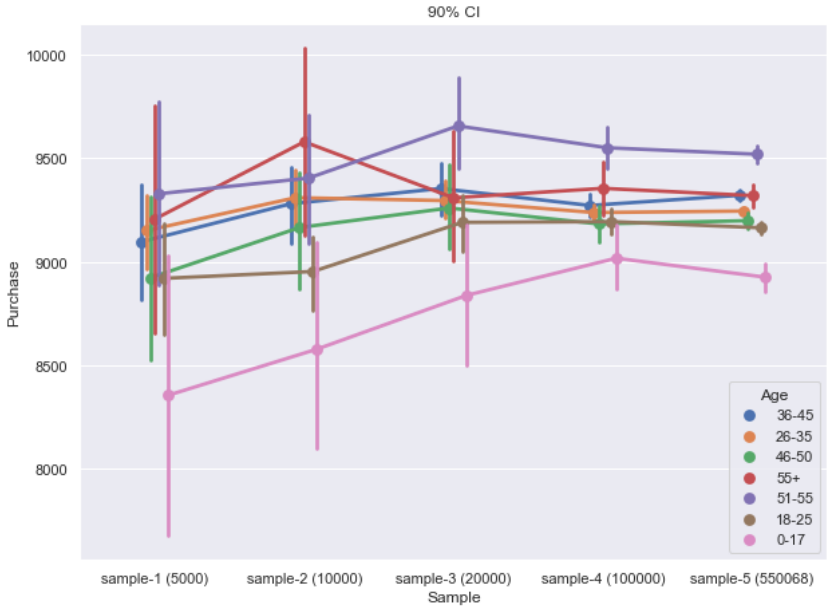
sample-1 (5000)	0-17	8354.76	7662.70	9046.83	8354.76	7530.12
	18-25	8919.83	8659.43	9180.23	8919.83	8609.54
	26-35	9150.29	8969.98	9330.59	9150.29	8935.44
	36-45	9097.05	8834.94	9359.17	9097.05	8784.72
	46-50	8923.17	8528.69	9317.64	8923.17	8453.12
	51-55	9328.61	8890.41	9766.80	9328.61	8806.47
	55+	9206.21	8641.11	9771.32	9206.21	8532.85
sample-2 (10000)	0-17	8578.54	8083.17	9073.91	8578.54	7988.27
	18-25	8953.15	8765.07	9141.23	8953.15	8729.04
	26-35	9309.60	9178.75	9440.46	9309.60	9153.68
	36-45	9280.61	9096.16	9465.07	9280.61	9060.82
	46-50	9165.75	8877.09	9454.41	9165.75	8821.79
	51-55	9404.26	9091.42	9717.11	9404.26	9031.48
	55+	9579.56	9153.89	10005.23	9579.56	9072.34
sample-3 (20000)	0-17	8838.01	8488.00	9188.02	8838.01	8420.95
	18-25	9190.35	9055.64	9325.07	9190.35	9029.83
	26-35	9295.51	9203.64	9387.39	9295.51	9186.03
	36-45	9353.67	9222.62	9484.73	9353.67	9197.52
	46-50	9259.76	9056.58	9462.95	9259.76	9017.66
	51-55	9656.25	9438.68	9873.81	9656.25	9397.00
	55+	9308.92	9007.95	9609.89	9308.92	8950.30
sample-4 (100000)	0-17	9017.37	8857.36	9177.37	9017.37	8826.71
	18-25	9194.01	9132.15	9255.87	9194.01	9120.30
	26-35	9237.68	9196.67	9278.70	9237.68	9188.81
	36-45	9272.09	9214.11	9330.07	9272.09	9203.00
	46-50	9182.95	9094.10	9271.80	9182.95	9077.08
	51-55	9550.61	9451.31	9649.92	9550.61	9432.28
	55+	9354.73	9223.23	9486.23	9354.73	9198.04
sample-5 (550068)	0-17	8925.54	8857.41	8993.67	8925.54	8844.36
	18-25	9164.19	9138.03	9190.34	9164.19	9133.02
	26-35	9244.95	9227.43	9262.46	9244.95	9224.08
	36-45	9320.89	9296.12	9345.66	9320.89	9291.37
	46-50	9198.53	9160.52	9236.54	9198.53	9153.24
	51-55	9519.56	9477.24	9561.88	9519.56	9469.13
	55+	9319.77	9264.04	9375.49	9319.77	9253.37

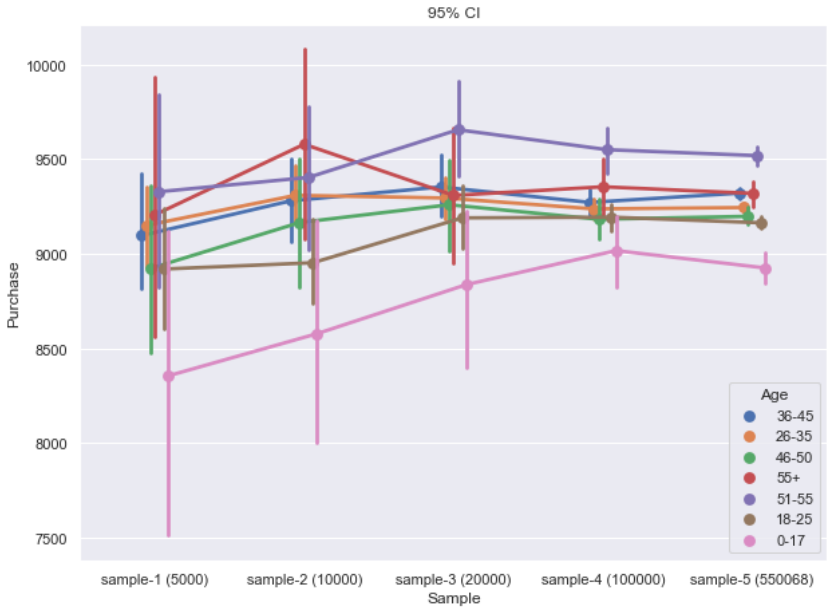
confidence-level		99			
		ci-upper	mean	ci-lower	ci-upper
sample	value				
sample-1 (5000)	0-17	9179.41	8354.76	7271.00	9438.53
	18-25	9230.11	8919.83	8512.04	9327.61
	26-35	9365.14	9150.29	8867.93	9432.65
	36-45	9409.38	9097.05	8686.58	9507.52
	46-50	9393.21	8923.17	8305.43	9540.91
	51-55	9850.75	9328.61	8642.40	10014.82
	55+	9879.58	9206.21	8321.27	10091.16
sample-2 (10000)	0-17	9168.81	8578.54	7802.80	9354.29

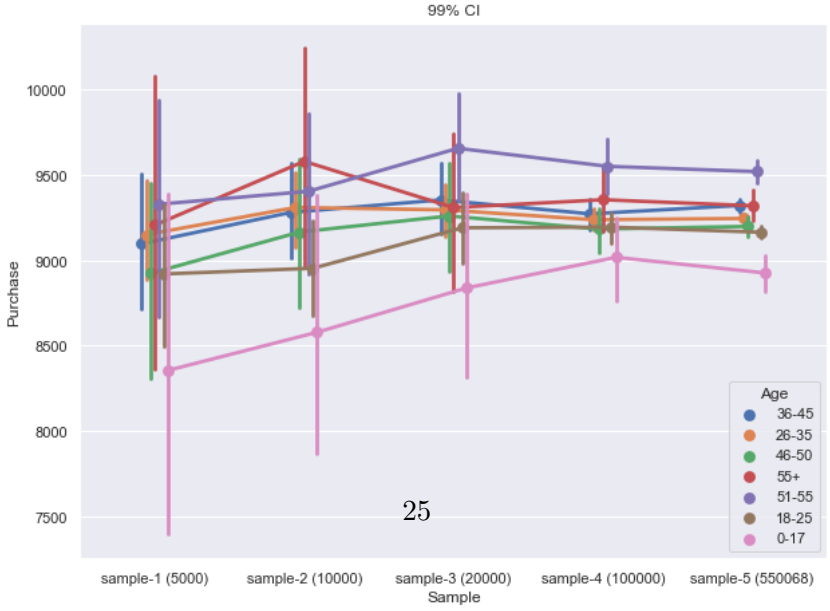
```
                    18-25   9177.26  8953.15  8658.62   9247.69
                    26-35   9465.52  9309.60  9104.69   9514.52
                    36-45   9500.40  9280.61  8991.76   9569.47
                    46-50   9509.71  9165.75  8713.71   9617.79
                    51-55   9777.04  9404.26  8914.35   9894.18
                    55+    10086.78  9579.56  8912.96  10246.16
sample-3 (20000)    0-17    9255.07  8838.01  8289.90   9386.12
                    18-25   9350.87  9190.35  8979.39   9401.31
                    26-35   9404.99  9295.51  9151.63   9439.39
                    36-45   9509.83  9353.67  9148.45   9558.90
                    46-50   9501.87  9259.76  8941.58   9577.95
                    51-55   9915.49  9656.25  9315.54   9996.96
                    55+     9667.55  9308.92  8837.61   9780.24
sample-4 (100000)   0-17    9208.03  9017.37  8766.80   9267.94
                    18-25   9267.72  9194.01  9097.14   9290.89
                    26-35   9286.55  9237.68  9173.46   9301.91
                    36-45   9341.18  9272.09  9181.29   9362.89
                    46-50   9288.82  9182.95  9043.82   9322.09
                    51-55   9668.95  9550.61  9395.10   9706.13
                    55+     9511.42  9354.73  9148.81   9560.66
sample-5 (550068)   0-17    9006.72  8925.54  8818.85   9032.23
                    18-25   9195.36  9164.19  9123.23   9205.15
                    26-35   9265.82  9244.95  9217.52   9272.38
                    36-45   9350.41  9320.89  9282.09   9359.68
                    46-50   9243.82  9198.53  9139.01   9258.06
                    51-55   9569.99  9519.56  9453.29   9585.83
                    55+     9386.17  9319.77  9232.50   9407.03
```

[217]:
```
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")

fig, ax = plt.subplots(3, 1, figsize=(10, 25))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="Age", x='Sample', ci=90,␣
 ↪data=combined_samples, dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="Age", x='Sample', ci=95,␣
 ↪data=combined_samples, dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="Age", x='Sample', ci=99,␣
 ↪data=combined_samples, dodge=True, ax=ax[2])
```

```
plt.show()
```


90% CI
10000
9500
9000
8500
8000
Purchase
Age
36-45
26-35
46-50
55+
51-55
18-25
0-17
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample


95% CI
10000
9500
9000
8500
8000
7500
Purchase
Age
36-45
26-35
46-50
55+
51-55
18-25
0-17
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample


99% CI
10000
9500
9000
8500
8000
7500
Purchase
Age
36-45
26-35
46-50
55+
51-55
18-25
0-17
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample

Observations

1. Average spending per transaction is highest for customers in 51-55 age group (\$9519) and lowest in 0-17 age group (\$8925). Average spending for age groups 55+ and 36-45 are very close and second highest (around \$9320). Average spending value for age groups 26-35 (\$9244), 46-50 (\$9198), and 18-25 (\$9164) just below average spending value (\$9255)
2. The estimated mean values and their corresponding **confidence intervals are non-overlapping** for customers in various age groups.
3. For the given confidence level, as the sample size increases, confidence interval gets narrower and we usually get better estimate of the mean value.
4. For the given sample size, as the confidence level increases, confidence interval widens.

Recommendations

1. Customers in age group 51-55 have the highest average spending per transaction, followed by customers in age groups 55+ and 36-45. The other age groups except for 0-17 are have Business must continue to focus on customers in this age group as they bring maximum revenue.
2. Customers in age group 0-17 have the lowest spending per transaction, presumably because they are dependent or have lower purchasing power. Business can consider promoting affordable and yet attractive products for customers in this age group to widen the customer base. This will allow offseting lower transactional spending by increasing number of transactions (more customers increases number of transactions).

1.10 Average spending per transaction by City category

[218]:
```
grps = df.groupby('City_Category')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print(total / cnt)

df['Purchase'].sum()/df['Purchase'].size
```

```
City_Category
A    8903.502451
B    9142.912942
C    9709.786528
dtype: float64
```

[218]: 9255.02429608703

Estimating average spending per transaction by City_Category for samples and calculating CI

[219]:
```
# Calculating CI using formula
res_df = getCIDataFrame('City_Category', samples, [90, 95, 99])

res_df
```

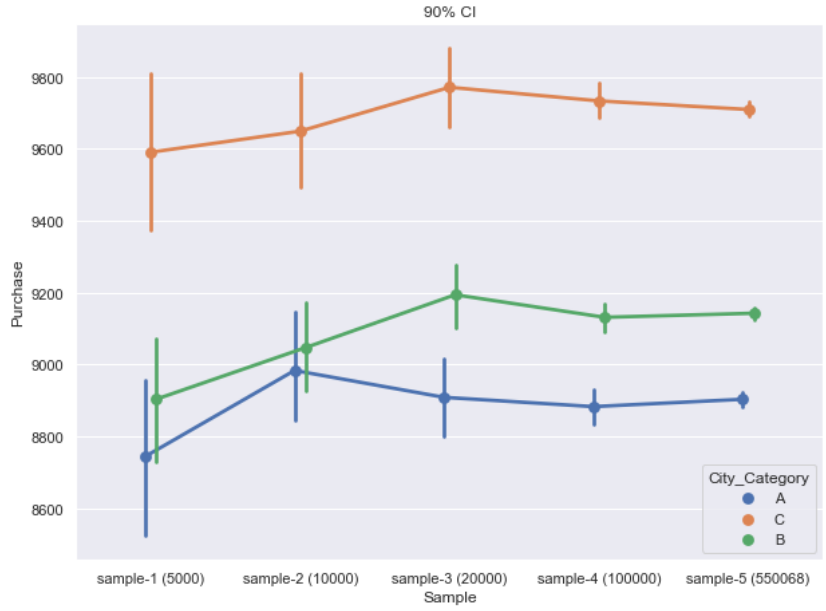
[219]:

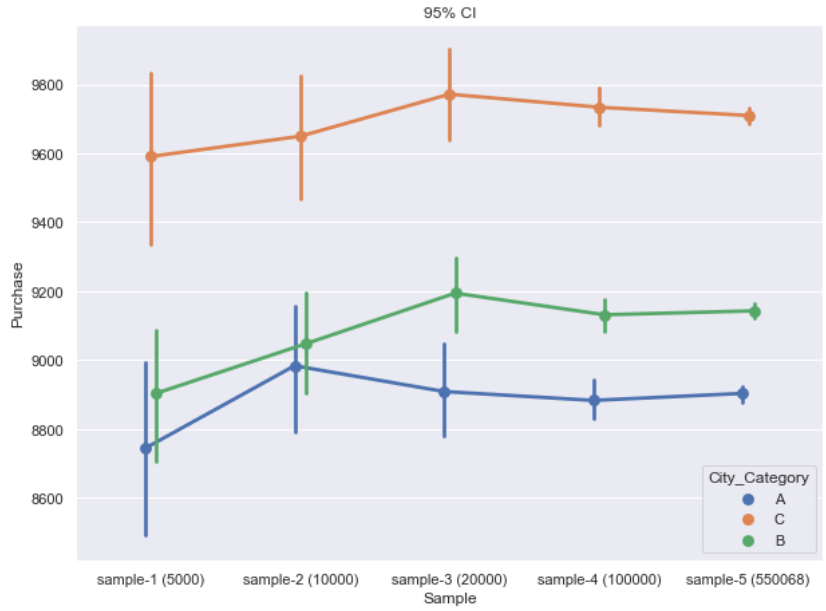
confidence-level		90			95			\
		mean	ci-lower	ci-upper	mean	ci-lower	ci-upper	
sample	value							
sample-1 (5000)	A	8744.32	8531.90	8956.75	8744.32	8491.21	8997.44	
	B	8903.87	8731.78	9075.95	8903.87	8698.82	9108.92	
	C	9590.93	9375.64	9806.23	9590.93	9334.39	9847.47	
sample-2 (10000)	A	8983.68	8827.64	9139.73	8983.68	8797.74	9169.62	
	B	9047.47	8923.88	9171.06	9047.47	8900.21	9194.73	
	C	9649.48	9496.06	9802.89	9649.48	9466.67	9832.28	
sample-3 (20000)	A	8908.93	8800.98	9016.88	8908.93	8780.30	9037.57	
	B	9194.29	9105.12	9283.46	9194.29	9088.04	9300.54	
	C	9771.60	9664.26	9878.93	9771.60	9643.70	9899.49	
sample-4 (100000)	A	8883.09	8834.05	8932.13	8883.09	8824.65	8941.52	
	B	9131.57	9092.18	9170.96	9131.57	9084.63	9178.50	
	C	9733.73	9685.40	9782.06	9733.73	9676.14	9791.32	
sample-5 (550068)	A	8903.50	8882.67	8924.34	8903.50	8878.67	8928.33	
	B	9142.91	9126.04	9159.79	9142.91	9122.80	9163.02	
	C	9709.79	9689.25	9730.32	9709.79	9685.32	9734.25	

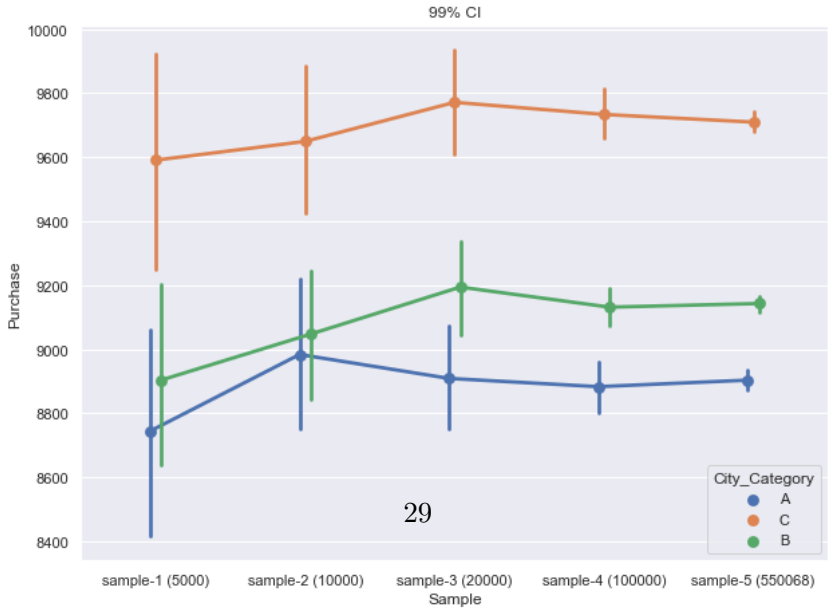
confidence-level		99		
		mean	ci-lower	ci-upper
sample	value			
sample-1 (5000)	A	8744.32	8411.67	9076.98
	B	8903.87	8634.38	9173.35
	C	9590.93	9253.78	9928.09
sample-2 (10000)	A	8983.68	8739.32	9228.05
	B	9047.47	8853.93	9241.00
	C	9649.48	9409.23	9889.73
sample-3 (20000)	A	8908.93	8739.88	9077.99
	B	9194.29	9054.66	9333.93
	C	9771.60	9603.51	9939.68
sample-4 (100000)	A	8883.09	8806.29	8959.88
	B	9131.57	9069.89	9193.25
	C	9733.73	9658.04	9809.42
sample-5 (550068)	A	8903.50	8870.87	8936.13
	B	9142.91	9116.49	9169.34
	C	9709.79	9677.63	9741.94

[226]:
```
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")
```

```
fig, ax = plt.subplots(3, 1, figsize=(10, 25))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="City_Category", x='Sample', ci=90,␣
 ↪data=combined_samples, dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="City_Category", x='Sample', ci=95,␣
 ↪data=combined_samples, dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="City_Category", x='Sample', ci=99,␣
 ↪data=combined_samples, dodge=True, ax=ax[2])

plt.show()
```


90% CI
9800
9600
9400
9200
9000
8800
8600
Purchase
City_Category
A
C
B
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample


95% CI
9800
9600
9400
9200
9000
8800
8600
Purchase
City_Category
A
C
B
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample


99% CI
10000
9800
9600
9400
9200
9000
8800
8600
8400
Purchase
City_Category
A
C
B
sample-1 (5000)
sample-2 (10000)
sample-3 (20000)
sample-4 (100000)
sample-5 (550068)
Sample

Observations

1. Average spending per transaction is highest for city category C (\$9709), followed by city category B (\$9142) and city category A (\$8903)
2. The estimated mean values and their corresponding **confidence intervals are non-overlapping** for the all three city categories.
3. For the given confidence level, as the sample size increases, confidence interval gets narrower and we usually get better estimate of the mean value.
4. For the given sample size, as the confidence level increases, confidence interval widens.

Recommendations

1. Customers from city category C have considerably higher spending per transaction than B and A. Business must continue to focus on customers from city category C as they bring maximum revenue.
2. Customers in city category B has slightly below average spending per transaction. Customers in category C have lowest average spending per transaction. Business needs to revisit their strategy for customers from these city categories and run special promotions/offers as necessary.

1.11 Average spending per transaction by Stay_In_Current_City_Years

[221]:
```
grps = df.groupby('Stay_In_Current_City_Years')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print((total / cnt).sort_values())

df['Purchase'].sum()/df['Purchase'].size
```

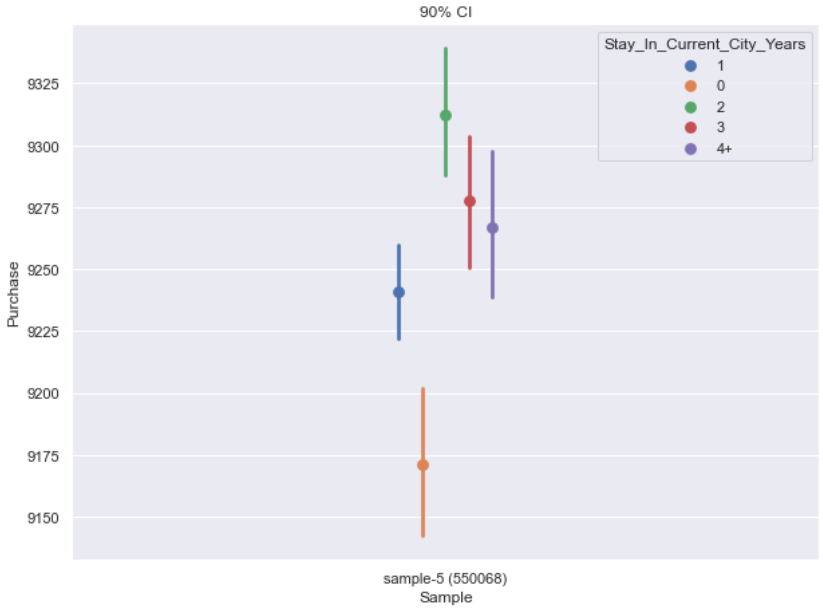
```
Stay_In_Current_City_Years
0     9171.127235
1     9240.816795
4+    9266.971544
3     9277.461211
2     9312.422730
dtype: float64
```

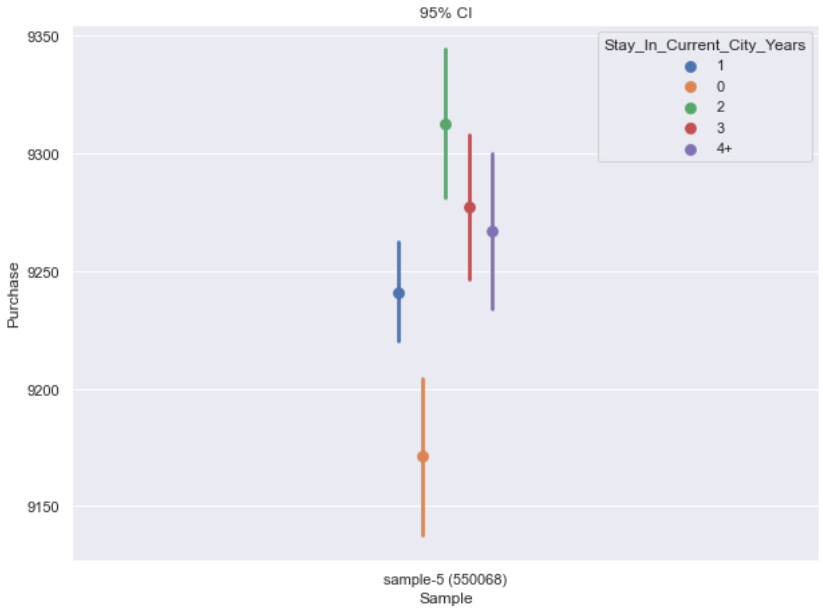
[221]:
```
9255.02429608703
```

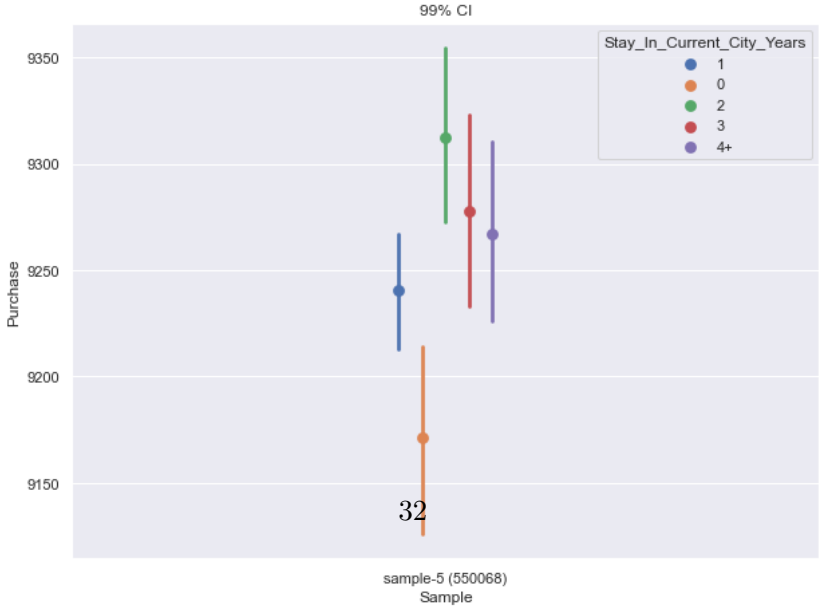
Estimating average spending per transaction by Stay_In_Current_City_Years for samples and calculating CI

Note: for brevity, we only compute CI for the last sample. We also skip computing CI with formula and only show graphs using bootstraping approach.

[222]:
```
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")

fig, ax = plt.subplots(3, 1, figsize=(10, 25))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="Stay_In_Current_City_Years", x='Sample',␣
 ↪ci=90, data=samples[len(samples) - 1], dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="Stay_In_Current_City_Years", x='Sample',␣
 ↪ci=95, data=samples[len(samples) - 1], dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="Stay_In_Current_City_Years", x='Sample',␣
 ↪ci=99, data=samples[len(samples) - 1], dodge=True, ax=ax[2])

plt.show()
```


90% CI
Stay_In_Current_City_Years
1
0
2
3
4+
9325
9300
9275
9250
9225
9200
9175
9150
Purchase
sample-5 (550068)
Sample


95% CI
Stay_In_Current_City_Years
1
0
2
3
4+
9350
9300
9250
9200
9150
Purchase
sample-5 (550068)
Sample


99% CI
Stay_In_Current_City_Years
1
0
2
3
4+
9350
9300
9250
9200
9150
Purchase
sample-5 (550068)
Sample

Observations

1. Average spending per transaction is lowest for costomers who have moved recently into the city (less than 1 year) or has stayed for less than 2 years.
2. Average spending per transaction is highest for customers who have stayed between 2-3 years. For customers with 3-4 and 4+ years of stay, the average spending per transaction are just above average.
3. The estimated mean values and their corresponding **confidence intervals are overlapping** across various stay durations.

Recommendations

1. The consumers who have moved in the city very recently (less than a year) or stayed for between one to two years have the lowest spending per transaction. This is presumably because their stay is either temporary or they are yet to make a decision about settling in the city for longer duration. Such consumers may be less likely spend on expensive and bulky, non-portable items such as furnitures and home decor items. Business can target such customers with products which are portable, less expensive, suitable for shorter use, and easily resalable.

1.12 Average spending per transaction by Occupation

[223]:
```
grps = df.groupby('Occupation')
cnt = grps['User_ID'].count()
total = grps['Purchase'].sum()
print((total / cnt).sort_values())

df['Purchase'].sum()/df['Purchase'].size
```

```
Occupation
9     8633.796137
19    8701.731297
20    8825.010563
2     8941.699319
1     8943.004069
10    8953.449652
0     9115.642681
18    9163.815539
3     9170.015751
11    9199.612377
4     9207.120457
6     9247.516482
13    9290.449081
5     9327.052887
16    9386.075086
7     9417.698645
```

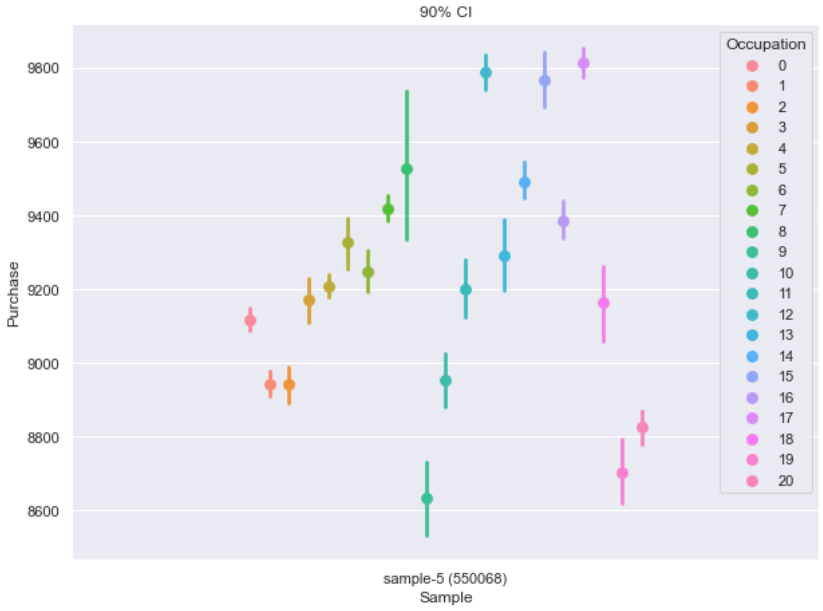
```
14    9491.389945
8     9525.778784
15    9767.482244
12    9786.140351
17    9813.336888
dtype: float64
```

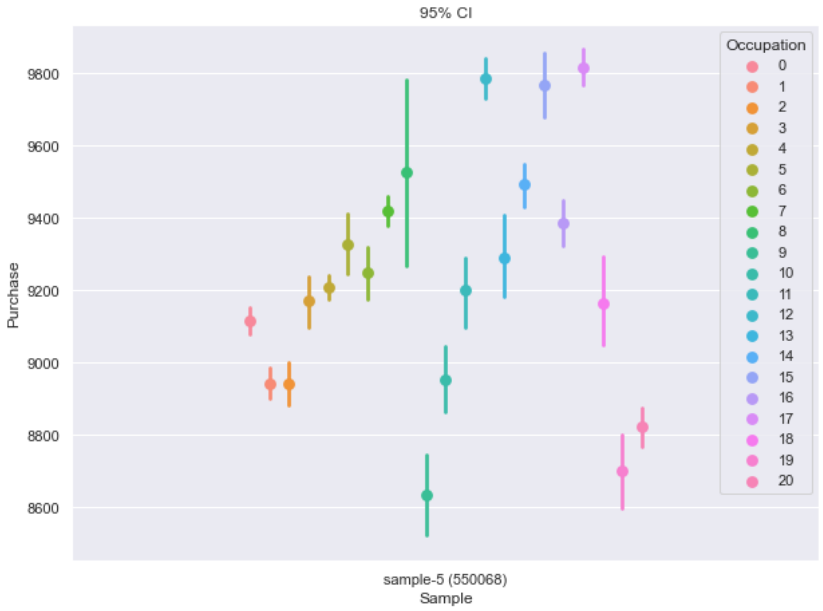
[223]: 9255.02429608703

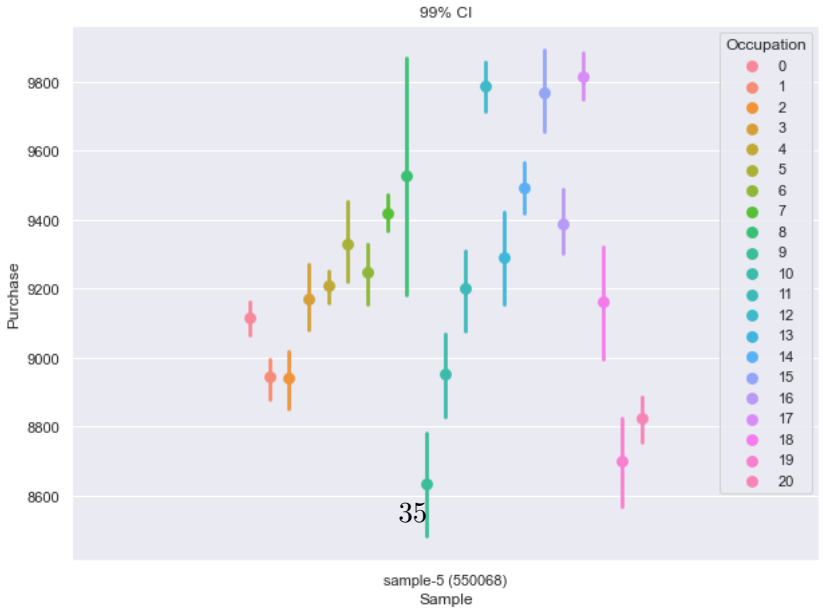
Estimating average spending per transaction by Occupation for samples and calculating CI

Note: for brevity, we only compute CI for the last sample. We also skip computing CI with formula and only show graphs using bootstraping approach.

[224]:
```
# calculating CI using bootstraping approach

import seaborn as sns
sns.set_theme(style="darkgrid")

fig, ax = plt.subplots(3, 1, figsize=(10, 25))

ax[0].title.set_text('90% CI')
sns.pointplot(y="Purchase", hue="Occupation", x='Sample', ci=90,␣
↪data=samples[len(samples) - 1], dodge=True, ax=ax[0])

ax[1].title.set_text('95% CI')
sns.pointplot(y="Purchase", hue="Occupation", x='Sample', ci=95,␣
↪data=samples[len(samples) - 1], dodge=True, ax=ax[1])

ax[2].title.set_text('99% CI')
sns.pointplot(y="Purchase", hue="Occupation", x='Sample', ci=99,␣
↪data=samples[len(samples) - 1], dodge=True, ax=ax[2])

plt.show()
```


90% CI
Occupation
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Purchase
9800
9600
9400
9200
9000
8800
8600
sample-5 (550068)
Sample


95% CI
Occupation
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Purchase
9800
9600
9400
9200
9000
8800
8600
sample-5 (550068)
Sample


99% CI
Occupation
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Purchase
9800
9600
9400
9200
9000
8800
8600
sample-5 (550068)
Sample

Observations

1. Average spending per transaction is highest for Occupation 17 (\$9813), 12 (\$9786) and 15 (\$9767). Above average for occupation 8, 14, 7, 16, 5 and 13.
2. Average spending per transaction is below average for the remaining occupations. It lowest for occupations 9, 19, 20, 2, 1, and 10.
3. The estimated mean values and their corresponding **confidence intervals are overlapping** across various occupations.

Recommendations

1. Customers from occupations 17, 12 and 15 have considerably higher spending per transaction. Business must continue to focus on these customers as they bring maximum revenue. Customers with occupations 8. 14. 7. 16. 5 and 15 also bring above average spending per transaction and should be a focus of business strategy.
2. For costomers in occupations with below average and lowest spending per transaction should be targetted with more affordable products/schemes to widen the customer base and effectively offset lower spending per transaction.