Answer key - Assignment 01

Problem - 1

1. Import Python packages that may be useful for examining a given dataset, and set plot style as "`searborn-darkgrid`".
2. Import `tips.csv` dataset and print the size and the first five observations of the dataset
3. Use the parameter **total_bill** to compute the following summary statistics
    - Mean
    - Median
    - Standard Deviation
    - Range
    - The Interquartile Range?
    - Skewness
4. Also find how many outlier observations are there?

[$6 + 3 + (3 \times 6) + 3 = 30$ marks]

In [27]:

```
## Solution 1.1:
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use("seaborn-darkgrid")
```

In [28]:

```
## Solution 1.2:
df = pd.read_csv("tips.csv") ## importing the .csv file
print(df.shape)
df.head(5)
```

```
(244, 7)
```

Out[28]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [29]:

```
## Solution 1.3:
df.total_bill.mean()
```

Out[29]:

```
19.785942622950824
```

In [30]:

```
df.total_bill.median()
```

Out[30]:

```
17.795
```

In [31]:

```
df.total_bill.std()
```

Out[31]:

```
8.902411954856856
```

In [32]:

```
df.total_bill.max() - df.total_bill.min()
```

Out[32]:

```
47.74
```

In [33]:

```
Q1, Q3 = df.total_bill.quantile([0.25,0.75]).values;
IQR = Q3 - Q1
print(IQR)
```

```
10.779999999999998
```

In [34]:

```
df.total_bill.skew()
```

Out[34]:

```
1.1332130376158205
```

In [35]:

```
## Solution 1.4:
(df.total_bill < Q1 - 1.5 * IQR).sum() + (df.total_bill > Q3 + 1.5 * IQR).sum()
```

Out[35]:

```
9
```

Problem - 2

1. Describe **total_bill** based on the summary statistics
   (Hint: Here we are interested in the shape of the distribution. You may consider the following options:
   - Examine histogram and check size/length of tails: (longer left tail $\implies$ left-skewed, longer right tail $\implies$ right-skewed)
   - Use skewness (negative=left-skewed, positive=right-skewed)
   - Compare the relative position of median and mean (mean<median $\implies$ left-skewed, mean>median $\implies$ right-skewed))
2. Plot the distribution of the parameter **total_bill** and justify your answer in part 1.

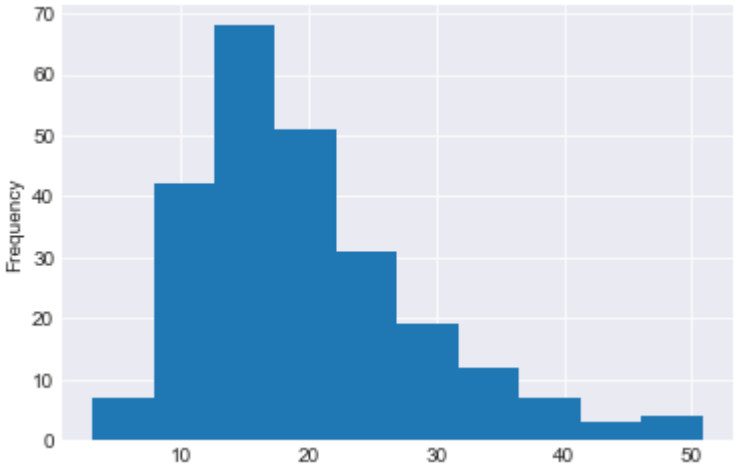
[$10 + 10 = 20$ marks]

Solution 2.1
The mean of the variable "total_bill" is 19.785942622950824 and the median is 17.795. As the median < mean, we can conclude that the distribution of the variable "total_bill" is right-skewed.
OR
The skewness measure for the variable "total_bill" is 1.1332130376158205. Hence, the ditsribution is right-skewed.
OR
Use the histogram.

In [36]:

```
## Solution 2.2:

df.total_bill.plot(kind="hist");
```



From the histogram, it is evident that the distribution is skewed to the right, which justifies the answer in part 1.

Problem - 3 : Data filtering

1. What is the average number of bills per day?
2. How many days had more that 60 bills?
3. What is the average percentage tip for male servers (compute by using operations: `query`, `indexer` and `groupby`)?
   Hint:
   • Method `query` uses a string to describe the condition used in the filter.

- df.query("sex=='Male'") performs the filtering of the required rows.
- df.query("sex=='Male'")['percentage_tip'] performs the filtering of rows and then selects the required column.
- df.query("sex=='Male'")['percentage_tip'].mean() performs … and then calculates the required mean.

• Method `loc` uses python expression to describe the condition used in the filter.

- loc is an indexer so even though it is a function it uses [] and not ()

• Method `groupby` divides the dataset into groups

- df.groupby("sex") group data based on specified column(s)
- df.groupby("sex")['percentage_tip'] select column percentage_tip in all groups
- df.groupby("sex")['percentage_tip'].mean() apply required aggregate function

4. Which group size gives the lowest percentage tip on average?
5. What is the max percentage tip for female servers over the weekend (Sat or Sun)? Solve by using operations `query` and `loc`.
6. Picking a bill at random, what is the probability that the server is female? (Hint: Filter using `query`/`loc` and compute probability)
7. Picking a bill at random, what is the probability that the server is female given that the day was Sunday
8. Which meal time has the highest mean tips?
9. True or False: "**Despite female servers having higher average percentage tips, it is male servers who earn the highest percentage tips**"

   This sentence has two claims:

   - "female servers have higher average percentage tips"
   - "male servers earn the highest percentage tips"

   you must verify both.
10. Which is the busiest day (in terms of the number of meals served)?

[$5 \times 10 = 50$ marks]

In [37]:

```
## Solution 3.1:
df.day.value_counts().mean()
```

Out[37]:

```
61.0
```

On an average, there were 61 bills per day.

In [38]:

```
## Solution 3.2:
# Step 1 - count number of bills on each day
df.day.value_counts()
```

Out[38]:

```
Sat     87
Sun     76
Thur    62
Fri     19
Name: day, dtype: int64
```

In [39]:

```
# Step 2 - convert count to False/True based on condition >60
df.day.value_counts()>60
```

Out[39]:

```
Sat      True
Sun      True
Thur     True
Fri     False
Name: day, dtype: bool
```

In [40]:

```
# Step 3 - count days that match condition by summing (remember False->0␣
#,→and True->1 when conveerting to int)
(df.day.value_counts()>60).sum()
```

Out[40]:

```
3
```

3 days had more than 60 bills.

In [41]:

```
## Solution 3.3:
# Step 1 - create new variable to store the percentage tip
## NOTE that this variable is used for all subsequent questions/calculations
df["percentage_tip"] = df.tip / df.total_bill * 100
```

In [42]:

```
# Soultion using function/method query
df.query("sex=='Male'")['percentage_tip'].mean()
```

Out[42]:

```
15.765054700429744
```

In [43]:

```
# Solution using intexer loc
df.loc[df.sex=='Male']['percentage_tip'].mean()
```

Out[43]:

```
15.765054700429744
```

In [44]:

```
# Solution using groupby
df.groupby("sex")['percentage_tip'].mean()
```

Out[44]:

```
sex
Female    16.649074
Male      15.765055
Name: percentage_tip, dtype: float64
```

In [45]:

```
# and a variation to output just the required answer
df.groupby("sex")['percentage_tip'].mean()['Male']
```

Out[45]:

```
15.765054700429744
```

On an average, the male servers receive 15.76% of the total bill as tip

In [46]:

```
## Solution 3.4:
df.groupby("size")["percentage_tip"].mean().sort_values().index[0]
```

Out[46]:

```
5
```

On an average, group size of 5 gives the lowest percentage tip

In [47]:

```
## Solution 3.5:
df.query("sex=='Female' and (day=='Sat' or day=='Sun') ")['percentage_tip'].max()
```

Out[47]:

```
41.66666666666667
```

In [48]:

```
# Alternative solution

df.loc[(df.sex=='Female') & ((df.day=='Sat') | (df.day=='Sun')),"percentage_tip"].max()
```

Out[48]:

```
41.66666666666667
```

The maximum percentage tip for female servers over the weekend is 41.67%

In [49]:

```
## Solution 3.6:
df.query("sex=='Female'").shape[0] / df.shape[0] # shape[0] represent number of rows
```

Out[49]:

```
0.35655737704918034
```

In [50]:

```
# Alternative solution - filter using query/loc and compute probability
# property size returns the number of values in a variable (series/coolumn)
df.query("sex=='Female'").sex.size / df.sex.size
```

Out[50]:

```
0.35655737704918034
```

Picking a bill at random, the probability that the server is female is
$\frac{\text{Number of orders served by female servers}}{\text{Total number of orders}} = 0.35656$

In [51]:

```
## Solution 3.7:
# filter to rows based on the given condition (the day was Sunday)
df_tmp = df.query("day=='Sun'")
# compute probability = number of rows matching required condition (female) /number of rows
# df_tmp.shape[0] returns number of rows
(df_tmp.sex=='Female').sum() / df_tmp.shape[0]
```

Out[51]:

```
0.23684210526315788
```

In [52]:

```
# filter to rows based on the given condition (the day was Sunday)
df_tmp = df.query("day=='Sun'")
# compute probability using value_counts
df_tmp.sex.value_counts(normalize=True)['Female']
```

Out[52]:

```
0.23684210526315788
```

In [53]:

```
df.query("day=='Sun'").sex.value_counts(normalize=True)['Female']
```

Out[53]:

```
0.23684210526315788
```

Picking a bill at random, the probability that the server is female given that the day was Sunday = $\frac{\text{Number of orders served by female servers on Sunday}}{\text{Total number of orders on Sunday}} = 0.236842$

In [54]:

```
## Solution 3.8:
df.groupby('time').tip.mean().index[0]
```

Out[54]:

```
'Dinner'
```

In [55]:

```
## Solution 3.9:
# claim 1: "female servers have higher average percentage tips"
df.groupby('sex').percentage_tip.mean()
```

Out[55]:

```
sex
Female    16.649074
Male      15.765055
Name: percentage_tip, dtype: float64
```

In [56]:

```
# store result in a boolean variable for later use
claim_1 = df.groupby('sex').percentage_tip.mean().index[0] == 'Female'
claim_1
```

Out[56]:

```
True
```

In [57]:

```
# "male servers earn the largest percentage tips"
df.groupby('sex').percentage_tip.max()
```

Out[57]:

```
sex
Female    41.666667
Male      71.034483
Name: percentage_tip, dtype: float64
```

In [58]:

```
# Note the reversing of the sort order - other approaches avialble also
claim_2 = df.groupby('sex').percentage_tip.max().sort_values(ascending=False).index[0] == '
claim_2
```

Out[58]:

```
True
```

In [59]:

```
# Test overall statement
claim_1 and claim_2
```

Out[59]:

```
True
```

Thus the statement: **“Despite female servers having higher average percentage tips, it is male servers who earn the highest percentage tips”** is verified to be **true**.

In [60]:

```
## Solution 3.10:
df.groupby('day')['size'].sum().sort_values(ascending=False).index[0]
```

Out[60]:

```
'Sat'
```

On Saturday, the total number of customers served is the highest. Hence, it is the busiest day.