Introductions To Computer Sciences

12.hafta Grup-12

—

Prepared by

Part-1

Metin Mirza Vatansever

Barış Köse

Part-2
Eren Demir
Emircan Koç
Münir Göktan Soykurt

Part-3
Ahmet Berkan Yurt
Ahmet Erdem Cesur

Q1

- **Import the dataset (strategy_csv). Summarize the data, look at the first few elements and structure of the dataset by using suitable python functions.**
- **Write a python code that finds the average user rating and user rating count for the game PUBG MOBILE.**
- **Write a python code that finds the names of the strategy games with the average user rating ≥ 4.5 and with the user rating count ≥ 300000.**
- **Write a python code that creates a new column in the dataset called FREE by using the Price variable. The FREE variable should have the value True if the game is free, and False if the game is not free.**
- **Select a suitable graphical method and visualize with Python to check whether free games have higher average user rating than paid games. Make a comment on the graph.**
- **Find the average user rating for free an ed not free games separately.**

Info about which libraries that we used:

Pandas: Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib: Is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

NumPy: NumPy is a mathematical library for the Python programming language that allows us to perform scientific calculations.

- **Import the dataset (strategy_csv). Summarize the data, look at the first few elements and structure of the dataset by using suitable python functions.**

Firstly, we import our libraries and our dataset.

Input:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
strategy_games = pd.read_csv("strategy_games.csv")
print(strategy_games)
```

Output:

```
              ID                      Name  ...  Original.Release.Date     Genre
0       284921427                    Sudoku  ...                   2008    Puzzle
1       284926400                   Reversi  ...                   2008     Board
2       284946595                   Morocco  ...                   2008     Board
3       285755462             Sudoku (Free)  ...                   2008    Puzzle
4       285831220              Senet Deluxe  ...                   2008     Board
...           ...                       ...  ...                    ...       ...
17002  1474626442  Stack Puzzle : Rise Tower  ...                   2019    Casual
17003  1474919257                 EachOther  ...                   2019    Family
17004  1474962324        Rabbit Vs Tortoise  ...                   2019  Strategy
17005  1474963671                   FaTaLL  ...                   2019    Action
17006  1475076711   The Three Kingdoms :Bomb  ...                   2019    Puzzle

[17007 rows x 9 columns]
```

And with the .shape method we saw number of rows and columns.

Input:

```
print(strategy_games.shape)
```

Output:

```
In [3]: print(strategy_games.shape)
(17007, 9)
```

With .dtypes method we saw the data types.

Input:

```
print(strategy_games.dtypes)
```

Output:

```
In [4]: print(strategy_games.dtypes)
ID                          int64
Name                       object
Average.User.Rating       float64
User.Rating.Count         float64
Price                     float64
Developer                  object
Age.Rating                 object
Original.Release.Date       int64
Genre                      object
dtype: object
```

With .head() method we saw the first 5 rows and columns.

Input:

```
strategy_games.head()
```

Output:

```
In [6]: strategy_games.head()
Out[6]:
          ID           Name  ...  Original.Release.Date   Genre
0  284921427         Sudoku  ...                   2008  Puzzle
1  284926400        Reversi  ...                   2008   Board
2  284946595        Morocco  ...                   2008   Board
3  285755462  Sudoku (Free)  ...                   2008  Puzzle
4  285831220   Senet Deluxe  ...                   2008   Board

[5 rows x 9 columns]
```

With .tail() method we saw the last 5 rows and columns.

Input:

```
strategy_games.tail()
```

Output:

```
In [7]: strategy_games.tail()
Out[7]:
             ID                       Name  ...  Original.Release.Date     Genre
17002  1474626442  Stack Puzzle : Rise Tower  ...                   2019    Casual
17003  1474919257                  EachOther  ...                   2019    Family
17004  1474962324          Rabbit Vs Tortoise  ...                   2019  Strategy
17005  1474963671                      FaTaLL  ...                   2019    Action
17006  1475076711   The Three Kingdoms :Bomb  ...                   2019    Puzzle

[5 rows x 9 columns]
```

With the .info() method we got general info about data. With this method we saw non-null number of input and type of datas.

Input:

```
strategy_games.info()
```

Output:

```
In [8]: strategy_games.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17007 entries, 0 to 17006
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     17007 non-null  int64
 1   Name                   17007 non-null  object
 2   Average.User.Rating    7561 non-null   float64
 3   User.Rating.Count      7561 non-null   float64
 4   Price                  16983 non-null  float64
 5   Developer              17007 non-null  object
 6   Age.Rating             17007 non-null  object
 7   Original.Release.Date  17007 non-null  int64
 8   Genre                  16650 non-null  object
dtypes: float64(3), int64(2), object(4)
memory usage: 1.2+ MB
```

- **Write a python code that finds the average user rating and user rating count for the game PUBG MOBILE.**

First of all we created a new variable named index_pubg and we matched this variable with PUBG MOBILE. Then with .loc[,] function we found the average user rating and user rating count for the PUBG MOBILE.

Input:

```
index_pubg=strategy_games["Name"]=="PUBG MOBILE"
strategy_games[index_pubg]
print(strategy_games.loc[index_pubg, "Average.User.Rating"])
print(strategy_games.loc[index_pubg, "User.Rating.Count"])
```

Output:

```
In [9]: index_pubg=strategy_games["Name"]=="PUBG MOBILE"
   ...: strategy_games[index_pubg]
   ...: print(strategy_games.loc[index_pubg, "Average.User.Rating"])
   ...: print(strategy_games.loc[index_pubg, "User.Rating.Count"])
13414    4.5
Name: Average.User.Rating, dtype: float64
13414    711409.0
Name: User.Rating.Count, dtype: float64
```

- **Write a python code that finds the names of the strategy games with the average user rating ≥ 4.5 and with the user rating count ≥ 300000.**

  Input:

```
strategy_games[(strategy_games['Average.User.Rating']>=4.5) & (strategy_games['User.Rating.Count']>=300000)]
```

  Output:

```
Out[57]:
              ID                            Name  ...       Genre  FREE
1378    529479190                 Clash of Clans  ...      Action  True
1921    597986893      Plants vs. Zombies\u2122 2  ...   Adventure  True
2410    672150402                      Boom Beach  ...      Action  True
7187   1053012308                    Clash Royale  ...      Action  True
12473  1270598321  Cash, Inc. Fame & Fortune Game  ...  Simulation  True
13414  1330123889                     PUBG MOBILE  ...      Action  True

[6 rows x 10 columns]
```

- **Write a python code that creates a new column in the dataset called FREE by using the Price variable. The FREE variable should have the value True if the game is free, and False if the game is not free.**

  Input:

```
print(strategy_games.loc[:,["Price"]])
strategy_games["FREE"]=strategy_games["Price"]<=0
print(strategy_games)
```

  Output:

```
In [12]: print(strategy_games.loc[:,["Price"]])
       Price
0       2.99
1       1.99
2       0.00
3       0.00
4       2.99
...      ...
17002   0.00
17003   0.00
17004   0.00
17005   0.00
17006   0.00
```

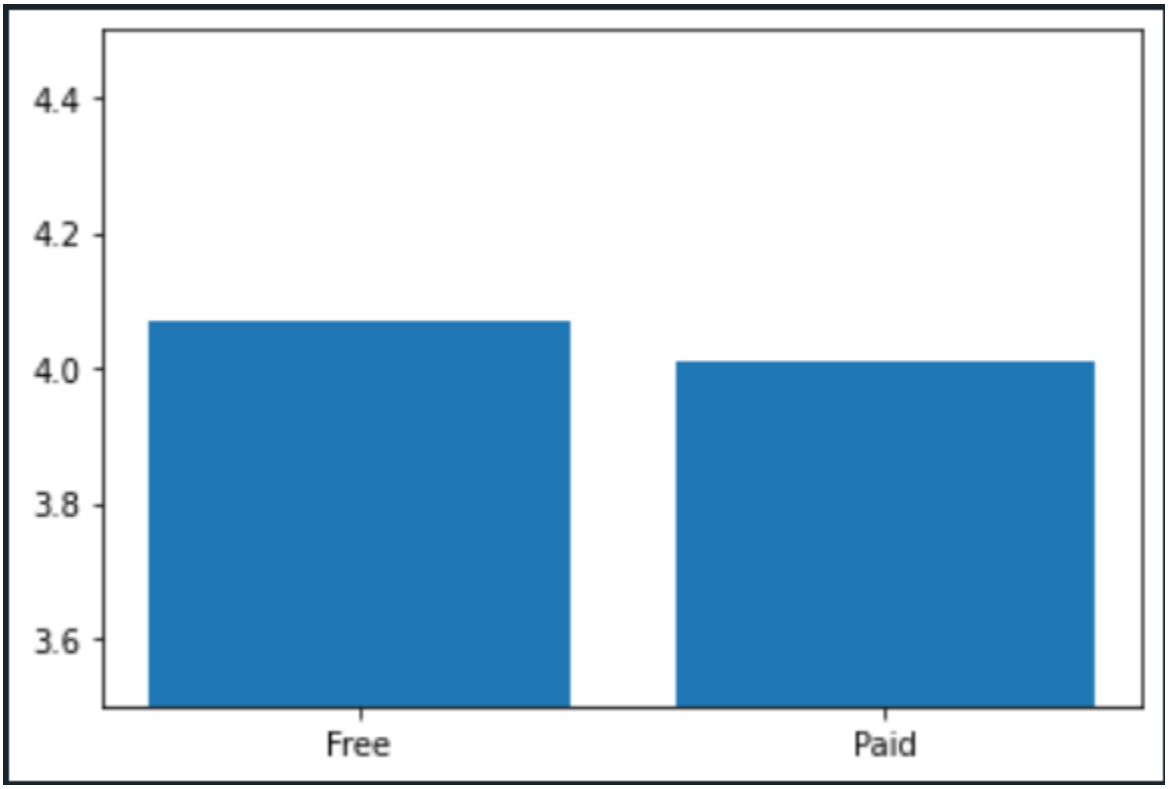
```
In [13]: strategy_games["FREE"]=strategy_games["Price"]<=0

In [14]: print(strategy_games)
              ID                         Name  ...     Genre   FREE
0       284921427                       Sudoku  ...    Puzzle  False
1       284926400                      Reversi  ...     Board  False
2       284946595                      Morocco  ...     Board   True
3       285755462                Sudoku (Free)  ...    Puzzle   True
4       285831220                 Senet Deluxe  ...     Board  False
...           ...                          ...  ...       ...    ...
17002  1474626442  Stack Puzzle : Rise Tower  ...    Casual   True
17003  1474919257                    EachOther  ...    Family   True
17004  1474962324           Rabbit Vs Tortoise  ...  Strategy   True
17005  1474963671                       FaTaLL  ...    Action   True
17006  1475076711   The Three Kingdoms :Bomb  ...    Puzzle   True

[17007 rows x 10 columns]
```

- **Select a suitable graphical method and visualize with Python to check whether free games have higher average user rating than paid games. Make a comment on the graph.**

  Input:

```
free = strategy_games.Price == 0
strategy_games['FREE'] = free.values
free_mean = strategy_games.groupby(['FREE'])['Average.User.Rating'].mean()
freeis=['Free','Paid']
means=[free_mean[1],free_mean[0]]
plt.ylim(3.5,4.5)
plt.bar(freeis,means)
plt.show()
```

  Output:



- **Find the average user rating for free and not free games separately.**

  Input:

```
index_free=strategy_games["FREE"]
print(strategy_games.loc[index_free, "Average.User.Rating"])
```

  Output:

```
In [19]: index_free=strategy_games["FREE"]
    ...: print(strategy_games.loc[index_free, "Average.User.Rating"])
2        3.0
3        3.5
5        3.0
6        2.5
8        2.5
        ...
17002    NaN
17003    NaN
17004    NaN
17005    NaN
17006    NaN
Name: Average.User.Rating, Length: 14212, dtype: float64
```

References:

https://medium.com/@isogretmen2018/pandas-k%C3%BCt%C3%BCphanesinde-loc-ve-iloc-kullan%C4%B1m%C4%B1-322b68e6c9da

https://medium.com/bili%C5%9Fim-hareketi/veri-bilimi-i%CC%87%C3%A7in-temel-python-k%C3%BCt%C3%BCphaneleri-1-numpy-750429a0d8e5#:~:text=NumPy%20(Numerical%20Python)%20bilimsel%20hesaplamalar%C4%B1,a%C3%A7%C4%B1s%C4%B1ndan%20python%20listelerinden%20daha%20kullan%C4%B1%C5%9Fl%C4%B1d%C4%B1r.

https://medium.com/bili%C5%9Fim-hareketi/veri-bilimi-i%CC%87%C3%A7in-temel-python-k%C3%BCt%C3%BCphaneleri-2-pandas-dcc12ae01b7d

https://medium.com/datarunner/matplotlibkutuphanesi-1-99087692102b#:~:text=Matplotlib%20Nedir%3F,boyutlu%20%C3%A7izimlerde%20ba%C5%9Fka%20k%C3%BCt%C3%BCphanelerden%20yararlan%C4%B1l%C4%B1r.

https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673

Q2

- **Find a suitable dataset from the internet to analyze.**
  - **we chose dataset about number of franchised fast food restaurants**
- **Put forward at least two hypotheses from the data.**
- **Filter the data according to your hypothesis.**
- **Use exploratory data analysis and graphical methods to check your hypothesis.**

Info about what we used:

matplotlib.pyplot: Is a collection of functions that make matplotlib work like MATLAB. Each pylot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

os.chdir: Give us a chance to change the current working directory to specified path.It takes only a single argument as a new directory path.

getcwd() : The method of the os module in python returns a string that contains the absolute path of the current working directory. The returned string doesn't include the trailing slash character.

pd.read_csv: It helps us to read the data we need, by partition. It only takes adding an argument and it will give you a list of what you want and also finds the column name.

.info(): It gives us the data's type , how many columns does it consist of,what is the type of variables and gives us an idea about if is there any missing data(NaN).

.head(): It helps us when we work in big datasets by showing first datas. The head() function is used to get the first n rows. It is useful for quickly testing if your object has the right type of data in it.

.tail(): This function returns last n rows from the object based on position. It is useful for quickly verifying data, for example, after sorting or appending rows.

.describe(): The .describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

.groupby: .groupby is a python package that analyze and process a data. This package makes it easy to read and create files in various formats.

.mean(): Categorically averages the data.

.loc: It helps us to locate and obtain data values from a dataset with ease.

startswith(): It allows us to select a column in a dataset and retrieve only data that we ask for that category.

sns.barplot(): Shows point estimates and confidence intervals as rectangular bars.

plt.xticks(): It determines how many degrees angle the texts on the x-axis of the graph we have created will be made.

plt.show(): Sh ows the charts and graphs that we created.

pd.concat(): It allows us to combine the two dataframes we have.

Q2-
First of all, we took the datasets that we use from Kaggle.
(A dataset related to fast food restaurants in the United States.)
Then we import our libraries.

```
In [14]: import numpy as np
         import pandas as pd
         import seaborn as sns
         import os
         import matplotlib.pyplot as plt
```

Then, with the os.chdir() command, we reached the directory where the dataset we wanted was located, and with the os.getcwd() command, we checked and verified which directory we were in for control purposes.

```
In [26]: os.chdir("C:\\Users\\oguzt\\Downloads\\archive (1)")
         os.getcwd()
Out[26]: 'C:\\Users\\oguzt\\Downloads\\archive (1)'
```

With pd.read_csv, we show the dataset and read it.

Input:

```
In [31]: data=pd.read_csv("top_50_fast_food_US.csv")
         data
```

Output:

Out[31]:

	company	category	sales_in_millions_2019	sales_per_unit_thousands_2019	franchised_units_2019	company_owned_units_2019	total_units_2019	un	total_units_2019	unit_change_from_2018
0	mcdonalds	burger	40413	2912	13154	692	13846		13846	-66
1	starbucks	snack	21550	1454	6768	8273	15041		15041	216
2	chick_fil_a	chicken	11000	4517	2500	0	2500		2500	130
3	taco_bell	global	11000	1502	6622	467	7089		7089	181
4	burger_king	burger	10300	1399	7294	52	7346		7346	16
5	subway	sandwich	10000	410	23802	0	23802		23802	-998
6	wendys	burger	9865	1666	5495	357	5852		5852	30
7	dunkin	snack	9220	968	9630	0	9630		9630	42
8	dominos	pizza	7100	1178	5815	342	6157		6157	253
9	panera_bread	sandwich	5925	2751	1202	1023	2225		2225	132
10	chipotle	global	5520	2195	0	2580	2580		2580	130
11	pizza_hut	pizza	5380	714	7393	23	7416		7416	-40
12	kfc	chicken	4820	1196	4009	56	4065		4065	-9
13	sonic_drive_in	burger	4687	1320	3329	197	3526		3526	-74
14	arbys	sandwich	3885	1183	2170	1189	3359		3359	30
15	little_caesars	pizza	3850	899	3652	561	4213		4213	-49
16	panda_express	global	3800	1765	138	2046	2184		2184	80
17	dairy_queen	snack	3760	713	4379	2	4381		4381	-26
18	popeyes_chicken	chicken	3750	1541	2458	41	2499		2499	131
19	jack_in_the_box	burger	3505	1565	2106	137	2243		2243	6
20	papa_johns	pizza	2655	837	2544	598	3142		3142	-57
21	whataburger	burger	2566	3080	127	703	830		830	5
22	jimmy_johns	sandwich	2105	759	2735	52	2787		2787	-13
23	hardees	burger	2070	1126	1713	117	1830		1830	-16
24	zaxbys	chicken	1840	2030	755	149	904		904	6
25	culvers	burger	1730	2435	726	6	733		733	46
26	five_guys	burger	1662	1359	872	496	1368		1368	10
27	raising_canes	chicken	1466	3208	86	371	457		457	57
28	wingstop	chicken	1400	1250	1200	31	1231		1231	107
29	carls_jr	burger	1390	1252	1052	48	1100		1100	-21
30	jersey_mikes	sandwich	1340	824	1595	72	1667		1667	173
31	bojangles	chicken	1290	1717	434	312	746		746	-10
32	in_n_out_burger	burger	1000	2882	0	354	354		354	14
33	steak_n_shake	burger	932	1673	183	365	577		577	-19
34	el_pollo_loco	chicken	894	1865	277	203	480		480	4
35	qdoba	global	850	1250	380	350	730		730	-21
36	checkers_rallys	burger	862	1087	634	256	890		890	4
37	firehouse_subs	sandwich	855	729	1115	38	1153		1153	24
38	del_taco	global	850	1554	296	300	596		596	16
39	tim_hortons	sandwich	840	1165	715	0	715		715	-12
40	moes	global	785	1095	719	3	722		722	3
41	papa_murphys	pizza	748	551	1301	67	1368		1368	-99
42	mcalisters_deli	sandwich	719	1629	438	31	469		469	18
43	jasons_deli	sandwich	705	2478	110	180	290		290	11
44	churchs_chicken	chicken	700	688	885	165	1050		1050	-35
45	shake_shack	burger	630	4214	22	163	185		185	59
46	marcos_pizza	pizza	628	726	875	40	915		915	32
47	baskin_robbins	snack	626	247	2524	0	2524		2524	-26
48	tropical_smoothie	snack	577	769	833	1	834			
49	auntie_annes	snack	563	562	1200	11	1211			

With data.info() we got information about the directory.

```
In [33]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 8 columns):
company                          50 non-null object
category                         50 non-null object
sales_in_millions_2019           50 non-null int64
sales_per_unit_thousands_2019    50 non-null int64
franchised_units_2019            50 non-null int64
company_owned_units_2019         50 non-null int64
total_units_2019                 50 non-null int64
unit_change_from_2018            50 non-null int64
dtypes: int64(6), object(2)
memory usage: 3.2+ KB
```

With data.head() we saw the first 5 columns.

```
In [34]: data.head()
```

Out[34]:

	company	category	sales_in_millions_2019	sales_per_unit_thousands_2019	franchised_units_2019	company_owned_units_2019	total_units_2019	unit_chang
0	mcdonalds	burger	40413	2912	13154	692	13846	
1	starbucks	snack	21550	1454	6768	8273	15041	
2	chick_fil_a	chicken	11000	4517	2500	0	2500	
3	taco_bell	global	11000	1502	6622	467	7089	
4	burger_king	burger	10300	1399	7294	52	7346	

```
In [35]: data.tail()
```

With data.tail() we saw the last 5 columns

```
In [35]: data.tail()
```

Out[35]:

	company	category	sales_in_millions_2019	sales_per_unit_thousands_2019	franchised_units_2019	company_owned_units_2019	total_units_2019	unit_
45	shake_shack	burger	630	4214	22	163	185	
46	marcos_pizza	pizza	628	726	875	40	915	
47	baskin_robbins	snack	626	247	2524	0	2524	
48	tropical_smoothie	snack	577	769	833	1	834	
49	auntie_annes	snack	563	562	1200	11	1211	

With data.describe() we got these informations.

```
In [36]: data.describe()
```

Out[36]:

	sales_in_millions_2019	sales_per_unit_thousands_2019	franchised_units_2019	company_owned_units_2019	total_units_2019	unit_change_from_2018
count	50.00000	50.000000	50.000000	50.000000	50.000000	50.000000
mean	4292.16000	1537.780000	2765.240000	470.400000	3236.240000	10.020000
std	6598.80288	921.338363	4123.294787	1229.366055	4390.605154	163.001501
min	563.00000	247.000000	0.000000	0.000000	185.000000	-996.000000
25%	851.25000	852.500000	487.000000	32.750000	767.000000	-19.750000
50%	1785.00000	1286.000000	1200.000000	156.000000	1517.500000	8.000000
75%	4786.75000	1753.000000	3180.500000	363.000000	3484.250000	54.250000
max	40413.00000	4517.000000	23802.000000	8273.000000	23802.000000	253.000000

HYPOTHESIS[1]= Subway is the most franchising brand in the sandwich category.
Code:

```
In [39]: data.groupby("category")["franchised_units_2019"].mean().sort_values(ascending=False)
```

```
Out[39]: category
         snack       4222.333333
         sandwich    3764.666667
         pizza       3596.666667
         burger      2621.928571
         chicken     1400.444444
         global      1359.166667
         Name: franchised_units_2019, dtype: float64
```

And;

```
In [42]: a=data.loc[data["category"].str.startswith("sandwich"),["company","franchised_units_2019"]]
         a
```

Out[42]:

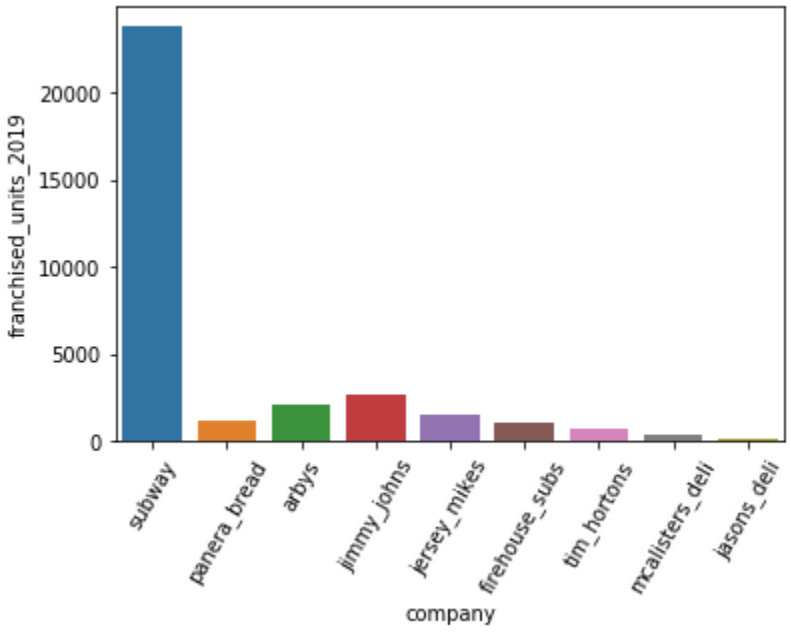
	company	franchised_units_2019
5	subway	23802
9	panera_bread	1202
14	arbys	2170
22	jimmy_johns	2735
30	jersey_mikes	1595
37	firehouse_subs	1115
39	tim_hortons	715
42	mcalisters_deli	438
43	jasons_deli	110

We reached all the restaurants selling sandwiches and as can be seen from the table, Subway is the restaurant that gives the most franchises. So our hypothesis is correct.

Let's check our hypothesis on the graph.

```
In [43]: sns.barplot(x="company",y="franchised_units_2019",data=a)
         plt.xticks(rotation=60)
         plt.show()
```

And the result



Hypothesis is correct.

HYPOTHESIS[2]=Burger King is the top selling brand in the burger category in the USA.

Code:

```
In [45]: data.groupby("category")["sales_in_millions_2019"].mean().sort_values(ascending=False)

Out[45]: category
         snack       6049.333333
         burger      5829.428571
         global      3800.833333
         pizza       3393.500000
         chicken     3017.777778
         sandwich    2930.444444
         Name: sales_in_millions_2019, dtype: float64
```

And;

```
In [46]: b=data.loc[data["category"].str.startswith("burger"),["company","sales_in_millions_2019"]]
         b
```

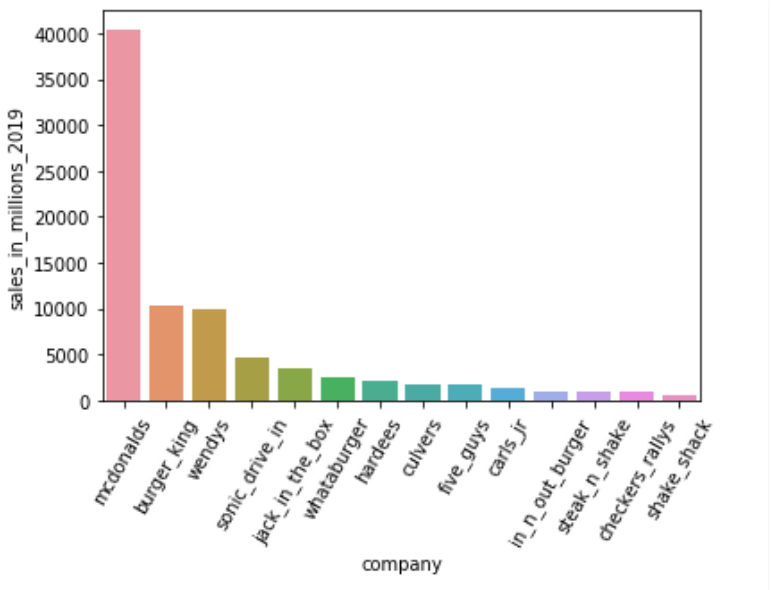
Out[46]:

	company	sales_in_millions_2019
0	mcdonalds	40413
4	burger_king	10300
6	wendys	9865
13	sonic_drive_in	4687
19	jack_in_the_box	3505
21	whataburger	2566
23	hardees	2070
25	culvers	1730
26	five_guys	1662
29	carls_jr	1390
32	in_n_out_burger	1000
33	steak_n_shake	932
36	checkers_rallys	862
45	shake_shack	630

We categorically separated all fast food sales, and then we reached the data of restaurants selling hamburgers.

```
In [47]: sns.barplot(x="company",y="sales_in_millions_2019",data=b)
         plt.xticks(rotation=60)
         plt.show()
```

We graphed it to understand better with this code.



As we can see from the graph and the table, our hypothesis turned out to be wrong, the best seller is McDonald's.

HYPOTHESIS[3]= In the USA, world cuisine flavors are trending and dealer opening rates are high.
Code:

```
In [75]: data.groupby("category")["unit_change_from_2018"].mean().sort_values(ascending=False)

Out[75]: category
         global      64.833333
         snack       50.333333
         chicken     42.333333
         pizza       11.666667
         burger      -0.428571
         sandwich   -70.555556
         Name: unit_change_from_2018, dtype: float64
```

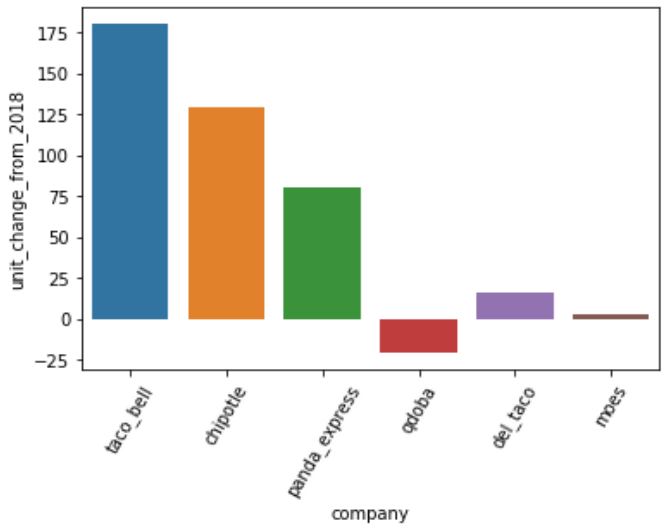
And;

```
In [66]: c=data.loc[data["category"].str.startswith("global"),["company","unit_change_from_2018"]]
         c
```

Out[66]:

	company	unit_change_from_2018
3	taco_bell	181
10	chipotle	130
16	panda_express	80
35	qdoba	-21
38	del_taco	16
40	moes	3

With categorically, we have accessed how many shops each product has opened and closed, and then, in detail, the data of the shops of each world cuisine.



As can be seen in the graph and table, the interest in world cuisines is increasing in the USA.

HYPOTHESIS[4]=In the USA, hamburger is preferred more than pizza.
Code:

```
In [98]: e=data.loc[data["category"].str.startswith("pizza"),["company","sales_in_millions_2019"]]
         e
```

Out[98]:

	company	sales_in_millions_2019
8	dominos	7100
11	pizza_hut	5380
15	little_caesars	3850
20	papa_johns	2655
41	papa_murphys	748
46	marcos_pizza	628

And;

```
In [93]: d=data.loc[data["category"].str.startswith("burger"),["company","sales_in_millions_2019"]]
         d
```

Out[93]:

	company	sales_in_millions_2019
0	mcdonalds	40413
4	burger_king	10300
6	wendys	9865
13	sonic_drive_in	4687
19	jack_in_the_box	3505
21	whataburger	2566
23	hardees	2070
25	culvers	1730
26	five_guys	1662
29	carls_jr	1390
32	in_n_out_burger	1000
33	steak_n_shake	932
36	checkers_rallys	862
45	shake_shack	630

We reached hamburger and pizza sales.

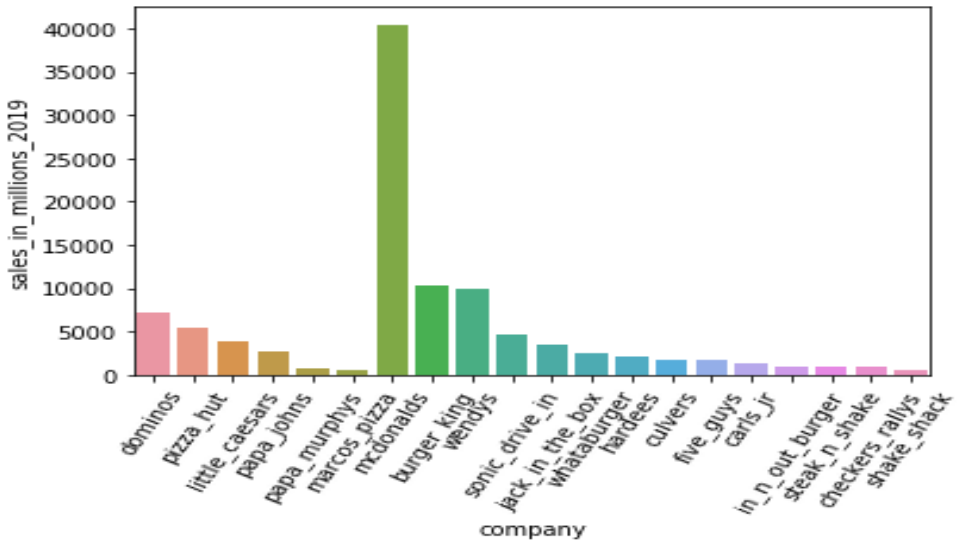
And we combined the two data with pd.concat.

```
In [101]: g=pd.concat([e,d])
          g
```

Out[101]:

	company	sales_in_millions_2019
8	dominos	7100
11	pizza_hut	5380
15	little_caesars	3850
20	papa_johns	2655
41	papa_murphys	748
46	marcos_pizza	628
0	mcdonalds	40413
4	burger_king	10300
6	wendys	9865
13	sonic_drive_in	4687
19	jack_in_the_box	3505
21	whataburger	2566
23	hardees	2070
25	culvers	1730
26	five_guys	1662
29	carls_jr	1390
32	in_n_out_burger	1000
33	steak_n_shake	932
36	checkers_rallys	862
45	shake_shack	630

and when we put it in graph;



As we can see from the graph and the chart, people prefer hamburgers more than pizza, so our hypothesis is true.

References:

https://python-istihza.yazbel.com/standart_moduller/os.html

https://www.veribilimiokulu.com/python-pandas-ile-temel-islemler/
https://medium.com/datarunner/python-ile-temel-veri-analizi-c51b29158f30
https://www.askpython.com/python-modules/pandas/python-loc-function#:~:text=The%20loc()%20function%20helps,value%20passed%20to%20the%20function.
https://seaborn.pydata.org/generated/seaborn.barplot.html
https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.xticks.html

Q3

What Is a Chi-Square Statistic?

A chi-square statistic is a test that measures how a model compares to actual observed data. The data used in calculating a chi-square statistic must be random, raw, mutually exclusive, drawn from independent variables, and drawn from a large enough sample. For example, the results of tossing a fair coin meet these criteria.

Chi-square tests are often used in hypothesis testing. The chi-square statistic compares the size of any discrepancies between the expected result and the actual result, given the size of the sample and the number of variables in the relationship.

For these tests, degrees of freedom are utilized to determine if a certain null hypothesis can be rejected based on the total number of variables and samples within the experiment. As with any statistic, the larger the sample size, the more reliable the result.

Example 1:

The libraries we used in question 3.

```
from scipy.stats import chi2_contingency
from scipy.stats import chi2
import pandas as pd
```

Data: Children born in two different regions are classified according to their weight

```
          düşük  normal  yüksek
1.Bölge      25      32      36
2.Bölge      21      36      22
```

Hypothesis: Is there a relationship between the place where babies are born and their weight.

Input:

```
veri = pd.DataFrame({"1.Bölge": [25,32,36],"2.Bölge": [21,36,22]}, index=["düşük","normal","yüksek"])
veri = veri.T
print(veri)
X2, p, serbestlik_der, beklenen = \
        chi2_contingency(veri)

alfa = 0.05

X2_tablo = chi2.ppf((1-alfa), serbestlik_der)

print(f"SerbestlikD.:{serbestlik_der}")
print(f"beklenen:\n{beklenen}")

if X2 > X2_tablo:
    print("ilişki vardır")
elif X2 < X2_tablo:
    print("ilişki yoktur")
```

Output:

```
In [26]: runfile('C:/Users/metin/Desktop/prj/Q3example.py', wdir='C:/Users/metin/Desktop/prj')
          düşük  normal  yüksek
1.Bölge      25      32      36
2.Bölge      21      36      22
SerbestlikD.:2
beklenen:
[[24.87209302 36.76744186 31.36046512]
 [21.12790698 31.23255814 26.63953488]]
ilişki yoktur
```

Result: No correlation was found between the place where babies are born and their weight.

Example 2:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
national = pd.DataFrame(["German"]*100.000.000 +["Norway"]*6.000.000 + \
                        ["Irish"]*6.000.000 + ["Sweden"]*12.000.000 + ["Other"]*6.752.000.000)


minnesota = pd.DataFrame(["German"]*1.900.000 + ["Norway"]*850.000 + \
                         ["Irish"]*600.000 +["Sweden"]*500.000 + ["Other"]*1.150.000)

national_table = pd.crosstab(index=national[0], columns="count")
minnesota_table = pd.crosstab(index=minnesota[0], columns="count")

print( "National")
print(national_table)
print(" ")
print( "Minnesota")
print(minnesota_table)
```

Result:

When we look at the results of the Chi-square analysis, we realize that the density of different populations in Minnesota does not have the same results and densities when we look at it on earth.

References:

https://www.investopedia.com/terms/c/chi-square-statistic.asp
https://github.com/oguzhankir/SHORTS/tree/main/ki_kare