در ابتدا کتابخانه هایی که برای این پروژه به آنها نیاز داریم را import میکنیم:

خواندن کتابخانه ها

```
# TO-DO
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[12] Python

در ادامه با استفاده از دستور زیر، مجموعه داده شهرهای جهان را که در اختیار مان قرار داده شده است میخوانیم:

دریافت مجموعه داده

```
# load data
file_path = 'worldcities.xlsx'
df = pd.read_excel(file_path)
df
```

[2] Python

	ville	ville_ascii	lat	lng	pays	iso2	iso3	admin_nom	capital	population	id
0	A Coruña	A Coruna	43.3667	-8.3833	Spain	ES	ESP	Galicia	minor	245468.0	1.724417e+09
1	A Yun Pa	A Yun Pa	13.3939	108.4408	Vietnam	VN	VNM	Gia Lai	minor	53720.0	1.704946e+09
2	Aabenraa	Aabenraa	55.0444	9.4181	Denmark	DK	DNK	Syddanmark	minor	16401.0	1.208000e+09
3	Aachen	Aachen	50.7756	6.0836	Germany	DE	DEU	North Rhine-Westphalia	minor	249070.0	1.276806e+09
4	Aadorf	Aadorf	47.4939	8.8975	Switzerland	CH	CHE	Thurgau	NaN	9036.0	1.756023e+09
...	...	...	...	...	...	...	...	...	...	...	...
44662	Żychlin	Zychlin	52.2453	19.6236	Poland	PL	POL	Łódzkie	NaN	9021.0	1.616509e+09
44663	Żyrardów	Zyrardow	52.0500	20.4333	Poland	PL	POL	Mazowieckie	minor	39374.0	1.616146e+09
44664	Zyryanka	Zyryanka	65.7360	150.8900	Russia	RU	RUS	Sakha (Yakutiya)	NaN	3627.0	1.643202e+09
44665	Zyryanovsk	Zyryanovsk	49.7453	84.2548	Kazakhstan	KZ	KAZ	NaN	minor	49658.0	1.398361e+09
44666	Żywiec	Zywiec	49.6892	19.2058	Poland	PL	POL	Śląskie	minor	30334.0	1.616870e+09

44667 rows × 11 columns

همانطور که مشاهده میشود، داده های ما ۴۴۶۶۷ سطر دارند و ۱۱ ستون که هر ستون بیانگر یک feature است و هر سطر یک sample.

در قسمت بعدی باید برخی ستون های این دیتافریم را حذف کنیم و نام برخی دیگر را تغییر دهیم. برای حالت اول کافی است لیستی از ستون های مدنظر را به عنوان ورودی به متود drop بدهیم و برای تغییر نام از یک دیکشنری استفاده میکنیم که مقادیر key همان اسم های اولیه و value ها همان اسم های جدید هستند و این دیکشنری را به عنوان ورودی به متود rename میدهیم.

حذف و تغییر نام ستون ها

ستون‌های `ville_ascii`، `capital`، `id` و `admin_nom` را از دیتافریم حذف نمایید. سپس نام ستون‌های `pays` و `ville` که به زبان فرانسوی نامگذاری شده‌اند را به ترتیب به `country` و `city` تغییر دهید.

```
# drop and rename
columns_to_remove = ['admin_nom', 'id', 'capital', 'ville_ascii']
columns_to_rename = {'ville': 'city', 'pays': 'country'}
df = df.drop(columns=columns_to_remove)
df = df.rename(columns= columns_to_rename)
df
```

[3] Python

	city	lat	lng	country	iso2	iso3	population
0	A Coruña	43.3667	-8.3833	Spain	ES	ESP	245468.0
1	A Yun Pa	13.3939	108.4408	Vietnam	VN	VNM	53720.0
2	Aabenraa	55.0444	9.4181	Denmark	DK	DNK	16401.0
3	Aachen	50.7756	6.0836	Germany	DE	DEU	249070.0
4	Aadorf	47.4939	8.8975	Switzerland	CH	CHE	9036.0
...	...	...	...	...	...	...	...
44662	Żychlin	52.2453	19.6236	Poland	PL	POL	9021.0
44663	Żyrardów	52.0500	20.4333	Poland	PL	POL	39374.0
44664	Zyryanka	65.7360	150.8900	Russia	RU	RUS	3627.0
44665	Zyryanovsk	49.7453	84.2548	Kazakhstan	KZ	KAZ	49658.0
44666	Żywiec	49.6892	19.2058	Poland	PL	POL	30334.0

44667 rows × 7 columns

در قسمت بعدی ابتدا باید شهرهایی که جمعیت کمتر از ۱ میلیون نفر دارند را از دیتافریم حذف کنیم.
برای این کار از دستور زیر استفاده میکنیم:

فیلتر کردن داده ها

شهرهای با جمعیت اکیدا کمتر از ۱ میلیون نفر را از دیتافریم حذف کرده و سپس نوع داده‌های ستون `population` را از `float` به `int32` تبدیل نمایید.

```
# filter data
df = df[df['population'] >= 1000000]
df
```

[7] ✓ 0.0s Python

	city	lat	lng	country	iso2	iso3	population
19	Aba	5.1167	7.3667	Nigeria	NG	NGA	1530000.0
83	Abidjan	5.3167	-4.0333	Côte d'Ivoire	CI	CIV	4980000.0
121	Abu Dhabi	24.4667	54.3667	United Arab Emirates	AE	ARE	1483000.0
131	Abuja	9.0667	7.4833	Nigeria	NG	NGA	3770000.0
169	Accra	5.5500	-0.2000	Ghana	GH	GHA	2388000.0
...	...	...	...	...	...	...	...
44496	Zhumadian	32.9773	114.0253	China	CN	CHN	7231234.0
44506	Zhuzhou	NaN	113.1469	China	CN	CHN	4020800.0
44511	Zibo	36.7831	118.0497	China	CN	CHN	2631647.0
44517	Zigong	29.3498	104.7645	China	CN	CHN	2678899.0
44629	Zunyi	27.7050	106.9336	China	CN	CHN	6606675.0

776 rows × 7 columns

دستور df['population'] >= 100000 به ما یک Series برمیگرداند که سطرهای آن متناظر با سطرهای df میباشد و مقادیر هر سطر True یا False است که مبنای این ارزش گذاری، همان عبارت شرطی ای است که بالا نوشتیم. وقتی از این Series به شکل بالا استفاده میکنیم، عملا سطرهایی از df را نگه میداریم که جمعیت آنها بیشتر یا مساوی 1 میلیون نفر باشد.

برای تغییر تایپ مقادیر ستون جمعیت نیز از دستور زیر استفاده میکنیم:

```
# change type population
df['population'] = df['population'].astype('int32')
df
```

[11] ✓ 0.0s Python

```
/tmp/ipykernel_8433/2018149386.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['population'] = df['population'].astype('int32')
```

	city	lat	lng	country	iso2	iso3	population
19	Aba	5.1167	7.3667	Nigeria	NG	NGA	1530000
83	Abidjan	5.3167	-4.0333	Côte d'Ivoire	CI	CIV	4980000
121	Abu Dhabi	24.4667	54.3667	United Arab Emirates	AE	ARE	1483000
131	Abuja	9.0667	7.4833	Nigeria	NG	NGA	3770000
169	Accra	5.5500	-0.2000	Ghana	GH	GHA	2388000
...	...	...	...	...	...	...	...
44496	Zhumadian	32.9773	114.0253	China	CN	CHN	7231234
44506	Zhuzhou	NaN	113.1469	China	CN	CHN	4020800
44511	Zibo	36.7831	118.0497	China	CN	CHN	2631647
44517	Zigong	29.3498	104.7645	China	CN	CHN	2678899
44629	Zunyi	27.7050	106.9336	China	CN	CHN	6606675

776 rows × 7 columns

در بخش بعدی ردیف های تکراری را حذف میکنیم و سپس داده هایی که حداقل ۶ مقدار آن ها سالم و گم نشده است را نگه میداریم، این مثل آن است که بگوییم داده هایی را نگه میداریم که حداکثر ۱ missing value دارند.

کار با داده های تکراری و گم شده

نخست ردیف‌های تکراری دیتافریم و سپس ردیف‌هایی که بیش از یک مورد (۲ مورد و بیش‌تر) از اطلاعات آن‌ها گم‌شده است را از دیتافریم حذف نمایید.

```python
# remove duplicated and missed values
df = df.drop_duplicates()
df = df.dropna(thresh=6)
df
```

[6] Python

	city	lat	lng	country	iso2	iso3	population
19	Aba	5.1167	7.3667	Nigeria	NG	NGA	1530000
83	Abidjan	5.3167	-4.0333	Côte d'Ivoire	CI	CIV	4980000
121	Abu Dhabi	24.4667	54.3667	United Arab Emirates	AE	ARE	1483000
131	Abuja	9.0667	7.4833	Nigeria	NG	NGA	3770000
169	Accra	5.5500	-0.2000	Ghana	GH	GHA	2388000
...	...	...	...	...	...	...	...
44496	Zhumadian	32.9773	114.0253	China	CN	CHN	7231234
44506	Zhuzhou	NaN	113.1469	China	CN	CHN	4020800
44511	Zibo	36.7831	118.0497	China	CN	CHN	2631647
44517	Zigong	29.3498	104.7645	China	CN	CHN	2678899
44629	Zunyi	27.7050	106.9336	China	CN	CHN	6606675

767 rows × 7 columns

سپس مقادیر گم شده برای ستون های lat و lng را با استفاده از میانگین این مقادیر برای همان کشور پر میکنیم. این کار در چند مرحله انجام شده است و از ترکیب آن ها نتیجه حاصل شده است:

پر کردن داده های گم‌شده

برای مقادیر گم شده در ستون های `lat` و `lng` ، میانگین همان ستون را در **همان کشور** پر کنید

```python
# fill the missing values by their country
df['lat'] = df.groupby('country')['lat'].transform(lambda x: x.fillna(x.mean()))
df['lng'] = df.groupby('country')['lng'].transform(lambda x: x.fillna(x.mean()))
```

[13] ✓ 0.0s Python

ابتدا بر حسب مقادیر ستون country، دیتافریم را گروه بندی میکنیم. سپس مقادیر ستون مربوط به 'lat' و 'lng' را سلکت میکنیم و پس از آن از متود transform استفاده میکنیم. کار این متد آن است که روی تک تک گروه های ایجاد یک تابع را اجرا میکند، در شکل بالا ما یک lamda function را به آن پاس دادیم و کاری که این تابع انجام میدهد دقیقا همان چیزی است که صورت سوال مطرح کرده است.

در مرحله بعدی فاصله بین همه شهرها از تهران را محاسبه میکنیم و یک دیتافریم یک ستونه برای آن میسازیم. ابتدا با استفاده از کتابخانه نامپای، مقادیر lat و lng را که بر حسب درجه هستند به رادیان تبدیل میکنیم. سپس یک تابع تعریف میکنیم و دقیقا فرمول نوشته شده در داک پروژه را با استفاده از نامپای و توابع آن پیاده سازی میکنیم:

```python
# distance function
tehran_data = df[df['city'] == 'Tehran']
tehran_lat = np.radians(tehran_data['lat'].values[0])
tehran_lng = np.radians(tehran_data['lng'].values[0])

def haversine_distance_from_tehran():
    results = pd.DataFrame(columns=['distance_from_tehran'])
    a = (np.sin((np.radians(df['lat']) - tehran_lat) / 2) ** 2 +
         np.cos(np.radians(df['lat'])) * np.cos(tehran_lat) * np.sin((np.radians(df['lng']) - tehran_lng) / 2) ** 2)
    results['distance_from_tehran'] = 2 * 6371 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    return results


new_column = haversine_distance_from_tehran()
new_column
```

Python

	distance_from_tehran
19	5634.737243
83	6575.533770
121	1280.150009
131	5331.673578
169	6230.918394
...	...
44496	5659.546660
44506	5616.330650
44511	5851.598592
44517	4993.215896
44629	5260.416825

767 rows × 1 columns

سپس این ستون جدید را به دیتافریم قبلی اضافه میکنیم:

```python
# add a new column in DataFrame
df['distance_from_tehran'] = new_column
df
```

Python

	city	lat	lng	country	iso2	iso3	population	distance_from_tehran
19	Aba	5.116700	7.3667	Nigeria	NG	NGA	1530000	5634.737243
83	Abidjan	5.316700	-4.0333	Côte d'Ivoire	CI	CIV	4980000	6575.533770
121	Abu Dhabi	24.466700	54.3667	United Arab Emirates	AE	ARE	1483000	1280.150009
131	Abuja	9.066700	7.4833	Nigeria	NG	NGA	3770000	5331.673578
169	Accra	5.550000	-0.2000	Ghana	GH	GHA	2388000	6230.918394
...	...	...	...	...	...	...	...	...
44496	Zhumadian	32.977300	114.0253	China	CN	CHN	7231234	5659.546660
44506	Zhuzhou	32.174206	113.1469	China	CN	CHN	4020800	5616.330650
44511	Zibo	36.783100	118.0497	China	CN	CHN	2631647	5851.598592
44517	Zigong	29.349800	104.7645	China	CN	CHN	2678899	4993.215896
44629	Zunyi	27.705000	106.9336	China	CN	CHN	6606675	5260.416825

767 rows × 8 columns

در ادامه شهرها را ابتدا بر اساس حروف الفبا به صورت صعودی مرتب میکنیم و سپس بر اساس مقدار lat، به صورت نزولی مرتب میکنیم و این کار با استفاده از متود sort_values به آسانی انجام میگیرد:

مرتب سازی

شهرها را بر اساس حروف الفبا به صورت صعودی مرتب نمایید و سپس بر اساس مقدار ستون lat به صورت نزولی مرتب کرده

```python
# Sorting
df = df.sort_values(by=['city', 'lat'], ascending=[True, False])
df
```

Python

	city	lat	lng	country	iso2	iso3	population	distance_from_tehran
19	Aba	5.1167	7.3667	Nigeria	NG	NGA	1530000	5634.737243
83	Abidjan	5.3167	-4.0333	Côte d'Ivoire	CI	CIV	4980000	6575.533770
121	Abu Dhabi	24.4667	54.3667	United Arab Emirates	AE	ARE	1483000	1280.150009
131	Abuja	9.0667	7.4833	Nigeria	NG	NGA	3770000	5331.673578
169	Accra	5.5500	-0.2000	Ghana	GH	GHA	2388000	6230.918394
...	...	...	...	...	...	...	...	...
41227	Ürümqi	43.8225	87.6125	China	CN	CHN	4335017	3197.472953
385	Āgra	27.1800	78.0200	India	IN	IND	1585704	2687.466060
17397	İzmir	38.4200	27.1400	Turkey	TR	TUR	4320519	2166.791444
28638	Ōsaka	34.6939	135.5022	Japan	JP	JPN	15126000	7381.949949
34814	Şanlıurfa	37.1583	38.7917	Turkey	TR	TUR	1985753	1138.007632

767 rows × 8 columns

برای ذخیره سازی نیز همانند پایین عمل میکنیم و نام فایل را به عنوان آرگومان به متود to_csv میدهیم و چون نمیخواهیم index ها ذخیره شوند، index=False را در این متود استفاده میکنیم:

ذخیره سازی

دیتافریم مرتب شده را بدون نمایه‌هایش (index) در فایلی به فرمت روبه‌رو ذخیره کنید StudentNumber.csv

شماره دانشجویی خودتان را جایگزین StudentNumber کنید

```
# Save CSV file
df.to_csv('9931012.csv', index=False)
```

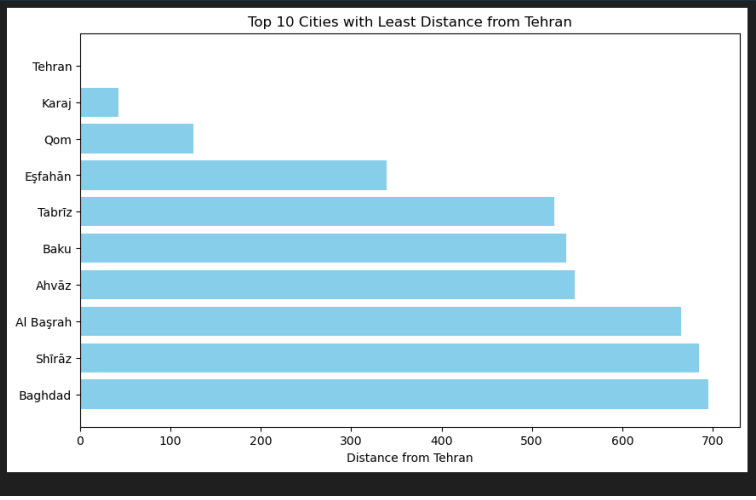
[11] Python

در قسمت بعدی ابتدا دیتافریم را بر حسب distance_from_tehran سورت میکنیم و ۱۰ داده ی بالایی آن را که نشان دهنده ۱۰شهر نزدیک به تهران هستند را انتخاب میکنیم.

سپس با استفاده از matplotlib.pyplot یک figure میسازیم و مقادیر distance_from_tehran را به عنوان ارتفاع و نام خود شهرها را به عنوان کتگوری به متود barh میدهیم. بقیه موارد نیز برای مشخص کردن لیبل مربوط به محور x و عنوان خود نمودار است. از دستور savefig نیز به منظور ذخیره این تصویر استفاده میکنیم:

```
# 10 Nearest Cities to Tehran
near_cities = df.sort_values(by='distance_from_tehran').head(10)
plt.figure(figsize=(10, 6))
plt.barh(near_cities['city'], near_cities['distance_from_tehran'], color='skyblue')
plt.xlabel('Distance from Tehran')
plt.title('Top 10 Cities with Least Distance from Tehran')
plt.gca().invert_yaxis()
plt.savefig('plot_1.png', bbox_inches='tight')
plt.show()
```

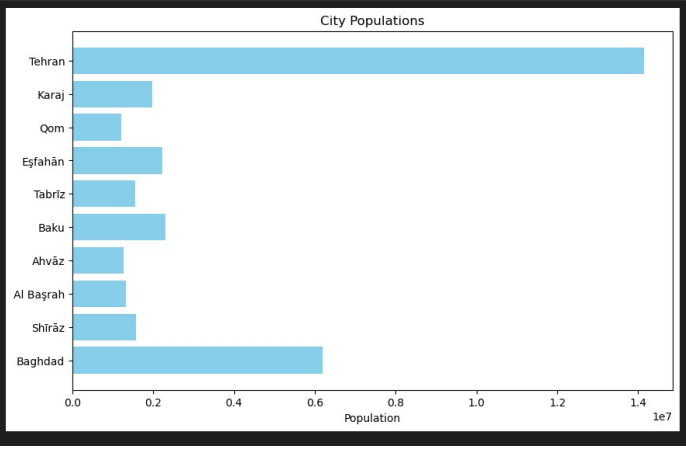
[14] ✓ 0.1s Python



قسمت بعدی نیز صرفا جمعیت همین شهرها را نمایش میدهیم و تفاوتش با قسمت قبلی این است که ارتفاع نمودار توسط جمعیت مشخص میشود:

```python
# Population of the 10 Nearest Cities to Tehran
plt.figure(figsize=(10, 6))
plt.barh(near_cities['city'], near_cities['population'], color='skyblue')
plt.xlabel('Population')
plt.title('City Populations')
plt.gca().invert_yaxis()
plt.savefig('plot_2.png', bbox_inches='tight')
plt.show()
```

[19] Python



در بخش آخر نیز scatter plot مربوط به طول و عرض جغرافیایی همه ی شهرها را نمایش میدهیم که نتیجه آن جالب است زیرا مجموعه نقاط، شبیه نقشه جهان میشوند:

```python
# City Latitudes and Longitudes
plt.figure(figsize=(10, 6))
plt.scatter(df['lng'], df['lat'], color='blue', marker='o')
plt.xlabel('Longitude (lng)')
plt.ylabel('Latitude (lat)')
plt.title('Scatter Plot of Latitude vs Longitude')
plt.savefig('plot_3.png',bbox_inches='tight')
plt.grid(True)
```

[22] Python