دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

گزارش تمرین عملی اول

داده کاوی

اروند درویش

۹۸۳۱۱۳۷

```
[1]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

ابتدا در اینجا کتابخانه های مورد نیازمان را import می کنیم.

```
[2]  df = pd.read_excel('/content/worldcities.xlsx') # df is our dataFrame
```

سپس فایل اکسل مسئله موردنظرمان را با کتابخانه پانداز باز می کنیم. توجه شود آدرسی که به عنوان ورودی به تابع خواندن فایل اکسل داده شده است، آدرس فایل در گوگل کولب است و در صورت نیاز باید تغییر داده شود در سیستم های دیگر.

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

دراینجا نیز ابتدا در خط اول ستون های گفته شده را با استفاده از تابع drop حذف میکنیم و در خط دوم دو ستون ville و pays را نام هایشان را عوض میکنیم تا به خوانایی دیتا فریم ما کمک ود زیرا کلمات قبلی فرانسوی بودند.

```
[4]  df = df[df['population']>1000000]
```

```
[5]  df['population']=df['population'].astype('int32')
```

در این قسمت سطر هایی یا در حقیقت شهر هایی که جمعیتشان کمتر از ۱ میلیون نفر بوده است حذف شده است به صورتی که ما سطر هایی از دیتافریم را انتخاب کردیم که مقدار مربوط به ستون population آن بیشتر از ۱ میلیون باشد و آن را با df که همان دیتا فریم اصلی مان بود جایگزین کردیم و در انتها نیز تایپ مقادیر این ستون را به نوع int32 درآوردیم.

```
[6]  df = df.drop_duplicates()
     df = df.dropna(thresh=df.shape[1] - 1)
```

تابع dropna با پارامتر thresh می‌تواند تعداد کمینه‌ی مقادیر غیرNaN را برای هر ردیف مشخص کند. در اینجا، با توجه به تعداد ستون‌های دیتافریم (df.shape[1]) ، ما مقدار آن را یکی کاهش داده‌ایم تا حداقل تعداد مقادیر معتبر برای هر ردیف را مشخص کنیم. این باعث می‌شود ردیف‌هایی که دارای بیش از یک مقدار NaN هستند حذف شوند.

```
df['lat'].fillna(df['lat'].mean(), inplace=True)
df['lng'].fillna(df['lng'].mean(), inplace=True)
```

سپس در اینجا با استفاده از تابع fillna مقادیری از ستون lat (یا lng) که NaN بود را با میانگین کل آن ستون یا ویژگی جایگزین (پر) کردیم.

```
import math

lat_teh = math.radians(df[df['city'] == 'Tehran']['lat'])
lng_teh = math.radians(df[df['city'] == 'Tehran']['lng'])
r = 6371
def tehran_distance(lat_dest, lng_dest):

    # Haversine formula
    dlat = lat_dest - lat_teh
    dlng = lng_dest - lng_teh
    a = math.sin(dlat/2)**2 + math.cos(lat_teh) * math.cos(lat_dest) * math.sin(dlng/2)**2
    d = 2 * r * math.atan2(math.sqrt(a), math.sqrt(1-a))

    return d
```

```
[9]  # add a new column in DataFrame

     for i in df.index:
       lat_dest = math.radians(df.at[i, 'lat'])
       lng_dest = math.radians(df.at[i, 'lng'])
       df.at[i, 'teh_dist'] = tehran_distance(lat_dest, lng_dest)
```

در اینجا همانگونه که در توضیحات صورت سوال گفته شد، فرمول هاورسین را مو به مو پیاده سازی کردیم و با زدن یک حلقه روی سطر های دیتافریم فاصله هر شهر را با تهران توسط تابع بدست آوردیم و در ستون جدید ذخیره کردیم به صورت مستقیم، تنها نکته پیاده سازی استفاده از تابع تبدیل رادیان، کسینوس و... بود که از کتابخانه math استفاده کردیم و همچنین با استفاده از امکاناتی که کتابخانه پانداز در خواندن دیتا فریم به ما داده بود توانستیم به مقادیر lat و lng شهر ها دسترسی پیدا کنیم و با استفاده از ایندکس، ویژگی های هر سطر را داشته باشیم. تعریف ستون جدید نیز در دیتا فریم ما به صورت مستقیم (direct) انجام شده است.

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

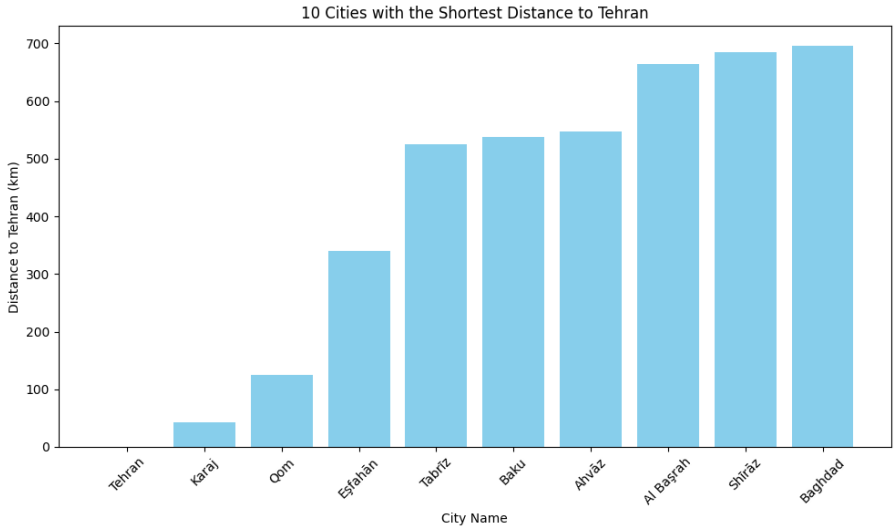
داده ها را به صورت گفته شده، یعنی ابتدا با توجه به ترتیب الفبایی اسم شهر ها و سپس به ترتیب بزرگ به کوچیک lat مرتب کرده ایم.

```python
# Save CSV file
df.to_csv('9831137.csv', index=False)
```

اینجا هم دیتافریم بعد از تمام پیش پردازش های انجام شده را در فایل CSV ذخیره میکنیم. (بدون ایندکس ها)

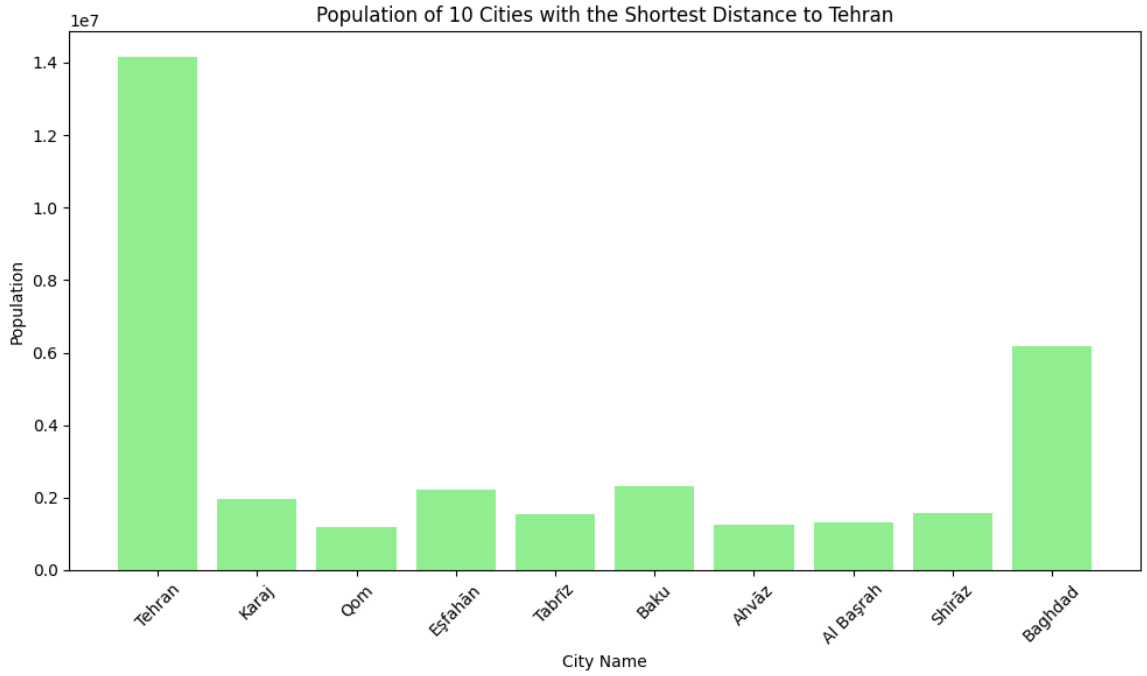
```python
# 10 Nearest Cities to Tehran
top_cities = df.nsmallest(10, 'teh_dist')

# Plotting the bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_cities['city'], top_cities['teh_dist'], color='skyblue')
plt.title('10 Cities with the Shortest Distance to Tehran')
plt.ylabel('Distance to Tehran (km)')
plt.xlabel('City Name')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('plot_1.png')
```

در این قسمت نموداری رسم می کنیم که فاصله ۱۰ شهری که کمترین فاصله را از تهران دارند را در قالب یک bar chart نمایش دهد.



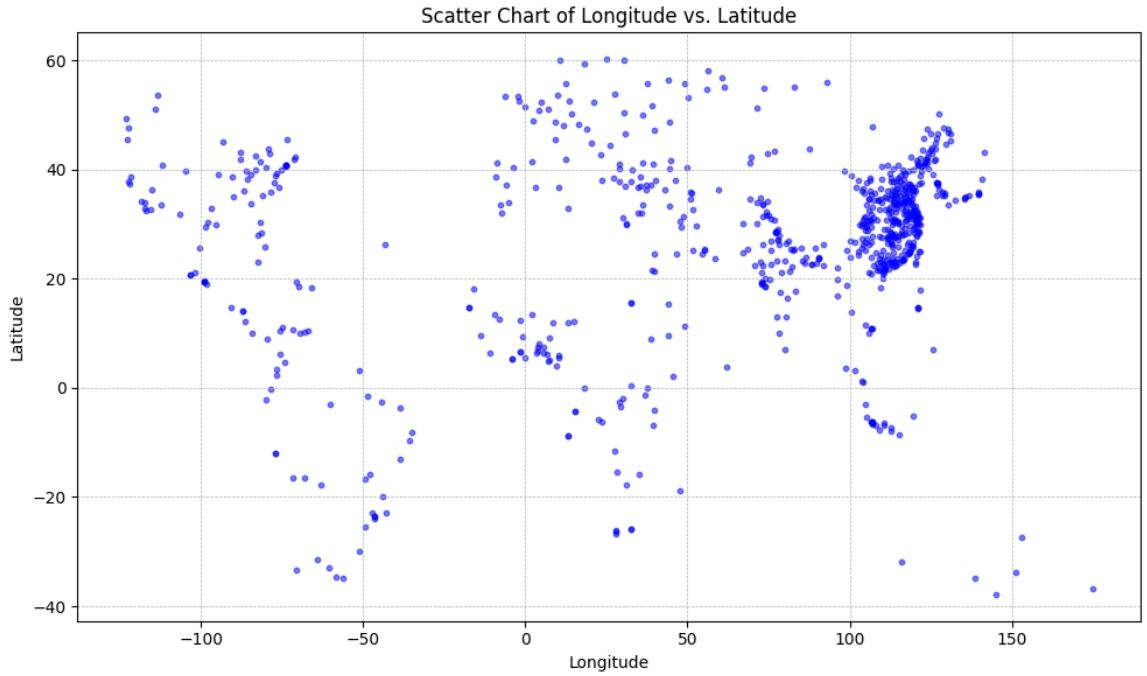
```python
# Population of the 10 Nearest Cities to Tehran

plt.figure(figsize=(10, 6))
plt.bar(top_cities['city'], top_cities['population'], color='lightgreen')
plt.title('Population of 10 Cities with the Shortest Distance to Tehran')
plt.ylabel('Population')
plt.xlabel('City Name')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('plot_2.png')
```

دراینجا اما جمعیت ۱۰ شهری که کمترین فاصله را با تهران دارند را رسم میکنیم.



```
# City Latitudes and Longitudes
plt.figure(figsize=(10, 6))
plt.scatter(df['lng'], df['lat'], c='blue', alpha=0.5, s=10)
plt.title('Scatter Chart of Longitude vs. Latitude')
plt.ylabel('Latitude')
plt.xlabel('Longitude')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.savefig('plot_3.png')
```

و در آخر نمودار Scatter ای رســـم می کنیم که محور افقی آن Longtitude و محور عمودی آن Latitude با شد. در پیاده سازی این قسمت نیز باید بگویم که پارامتر alpha در اسکتر پلات مشخص کننده شفاف بودن نقطه های پلت ما است که در حالت کامل یعنی یک نقطه کامل و پررنگ مقدار آن ۱ است ولی اینجا آن را ۰,۵ قرار دادیم تا تراکم و رو هم افتادن نقاط بهتر مشخص شود، پارامتر دیگر یعنی s نیز نشان دهنده سایز نقاط مشخص شده است. همچنین نمودار را به صورت گرید بندی نشان داده ایم تا بهتر مقادیر مشخص شوند. خروجی نهایی این نمودار با توجه به حجم بالای دیتاست تقریبا نقشه یک بعدی کره زمین را به همراه قاره ها نشان می دهد.



پایان