Sin City Wasn’t Made for You

https://github.com/sy3da/sincitywasntmadeforyou.git

Elijah Cantu, Tasnim Tabassum, Syeda Reza

SI-206 | 04/21/23

The goals for our project

Our project focuses on exploring the state of Nevada beyond the usual tourist hotspot of Las Vegas. While Las Vegas, aka the Sin City, is often associated with its glitz and glamor, we want to help travelers find other hidden gems in the state that are better suited to their interests.
At first, we planned to use 2 APIs and 1 website. To retrieve populations in cities, we wanted to use the Travel Lemming website. For Hotels, we would use Trip Advisor API to gather information about the cost, rating, and location of hotels in Nevada. Lastly, we planned on using YelpAPI to gather information about restaurants in Nevada cities.

The goals that we achieved

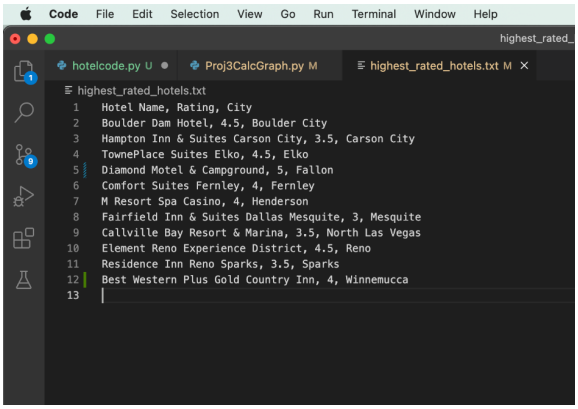
We used the Nevada Demographic website to gather information about cities in Nevada other than LA, and look at their populations. Then, we used YelpAPI to gather information about hotels in Nevada and their ratings, cost, and location. Lastly, we used the Tomtom API to gather information around a 10 km radius of each city. We were able to showcase the different locations in Nevada and make it easier for travelers to pick their destination.

Problems

- Our first problem was finding APIs and websites to use. As we researched more, we changed a lot about our project including two of our API/Websites and switching one of them with another one.
- Because of limited locations in Nevada, our API call wouldn't bring 25 rows of data all the time. Sometimes it brought significantly fewer data. Only a couple of entries would populate when we called our file. This was out of our control and it was extremely time-consuming.
- We had to learn how to combine all databases
- We had some trouble with the SELECT statements. However, we worked together to get ideas from each other to fix this.

Calculations:

```python
#calculation
def highestratedhotel(cur,conn):
    info = cur.execute("select hotel_name, city, rating, cost from YelpData order by city")
    cs = info.fetchall()
    highestRated = {}

    # Loop through the list of hotels and update the dictionary if we find a higher rated hotel for a given city
    for hotel in cs:
        name, city, rating, cost = hotel
        if city not in highestRated or rating > highestRated[city][1]:
            highestRated[city] = (name, rating, city, cost)

    # Convert the dictionary to a list of tuples and return it
    highestratedhotelLst = list(highestRated.values())

    with open('highest_rated_hotels.txt', 'w') as f:
        f.write("Hotel Name, Rating, City\n")
        for hotel in highestratedhotelLst:
            f.write("{}, {}, {}\n".format(hotel[0], hotel[1], hotel[2]))
    return highestratedhotelLst

```

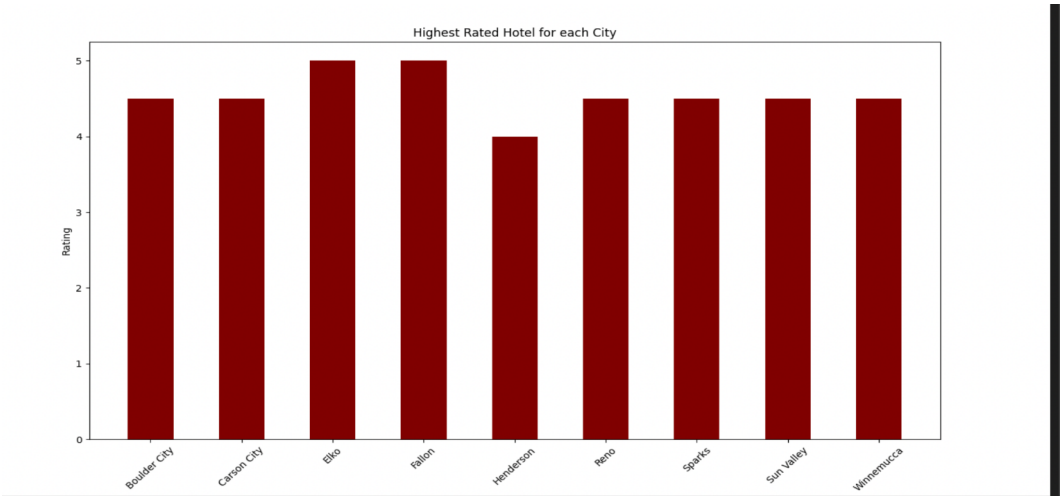


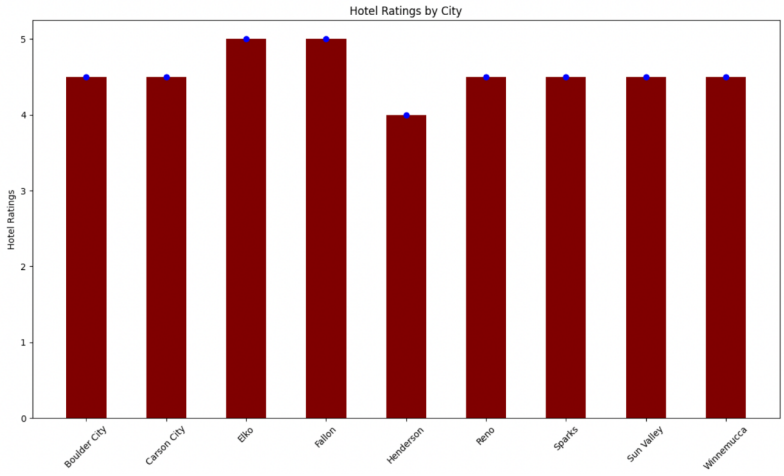

```python


def tourist_attractions_per_city():
  conn = sqlite3.connect('not_sin_city.db')
  cur = conn.cursor()
  cur.execute("SELECT cities.city, COUNT(*) as num_pois FROM pois INNER JOIN cities ON pois.city_id=cities.city_id GROUP BY citie
  data = cur.fetchall()
  cities = [row[0] for row in data]
  pois = [row[1] for row in data]
  avg_pois = sum(pois) / len(pois)

  fig, ax = plt.subplots(figsize=(15,10))
  ax.bar(cities, pois)
  ax.axhline(y=avg_pois, color='r', linestyle='--', label='Average')
  ax.set_xlabel('City', fontsize=16)
  ax.set_ylabel('Number of POIs', fontsize=16)
  ax.set_title('Tourist Attractions per City', fontsize=16)
  ax.legend()

  plt.subplots_adjust(bottom=0.2)
  # Rotate x-axis labels by 45 degrees
  plt.xticks(rotation=45, fontsize=16)


  with open('calculations.txt', 'a') as f:
    f.write('\nCity\tNumber of Tourist Attractions\n')
    for i in range(len(cities)):
        f.write(f'{cities[i]}\t{pois[i]}\n')
    f.write(f'Average\t{avg_pois}\n\n')
  fig.savefig('tourist_attractions_per_city.png')


```

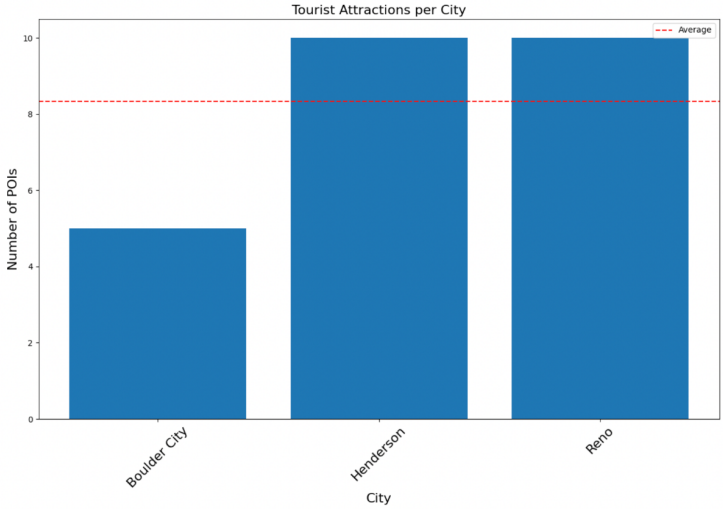
```python
def average_distance_between_tourist_locations_per_city():
    # Connect to the database
    conn = sqlite3.connect('not_sin_city.db')

    # Execute the join query
    query = '''
        SELECT p.poi_id, p.name, c.city, pos.lat, pos.lon
        FROM pois p
        JOIN cities c ON p.city_id = c.city_id
        JOIN positions pos ON p.poi_id = pos.poi_id
    '''
    result = conn.execute(query).fetchall()

    # Group the result by city
    city_data = {}
    for poi_id, name, city, lat, lon in result:
        if city not in city_data:
            city_data[city] = []
        city_data[city].append((lat, lon))

    # Compute the average distances for each city
    city_distances = {}
    for city, coords in city_data.items():
      if len(coords) > 1:
        city_avg_distance = np.mean([geodesic(coords[i], coords[j]).km for i in range(len(coords)) for j in range(i+1, len(coords))])
        if city_avg_distance > 0:
            city_distances[city] = city_avg_distance

    with open('calculations.txt', 'a') as f:
        f.write('\nCity\tAverage Distance between Tourist Locations\n')
        for city, distance in city_distances.items():
            f.write(f'{city}: {distance:.2f} km\n')
        f.write('\n')

    # Plot the results
    plt.subplots(figsize=(15,10))
    plt.bar(city_distances.keys(), city_distances.values())
    plt.xticks(rotation=45, fontsize=16)
    plt.xlabel('City', fontsize=16)
    plt.ylabel('Average distance (km)', fontsize=16)
    plt.title('Average distance between tourist attractions per city', fontsize=16)
    plt.subplots_adjust(bottom=0.2)
    plt.savefig('average_distance_between_tourist_attractions_per_city.png')
```

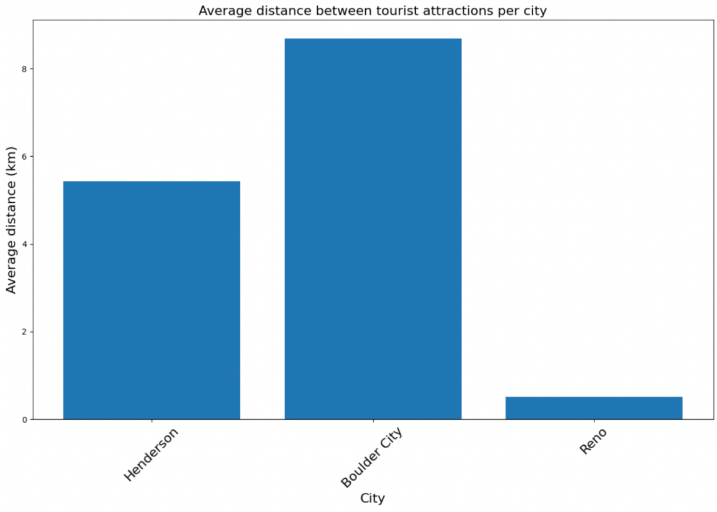
The Visualizations

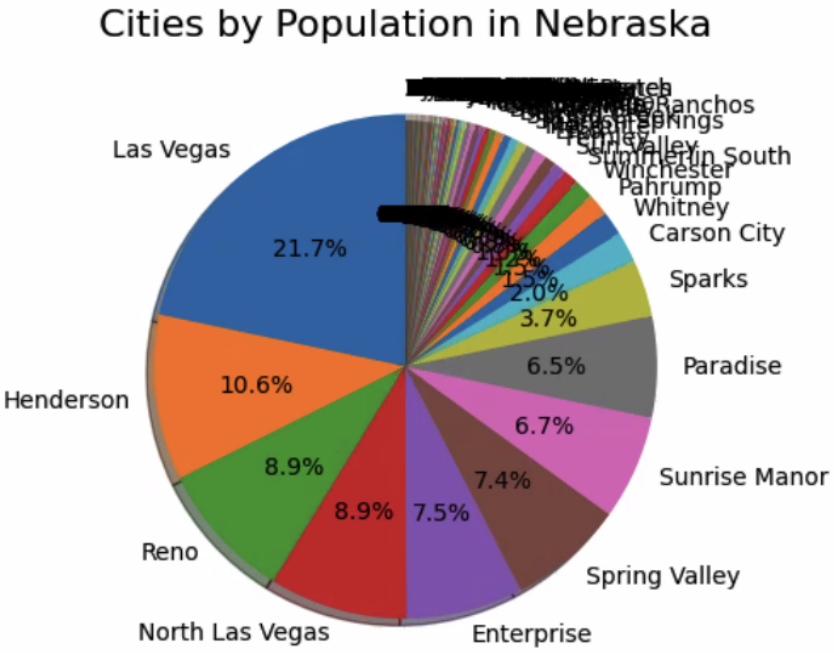
Here are two visualizations that show the cities with the highest rated hotels. One is in the format of a bar graph and the other is in the format of a scatterplot




Hotel Ratings by City
Hotel Ratings
0
1
2
3
4
5
Boulder City
Carson City
Elko
Fallon
Henderson
Reno
Sparks
Sun Valley
Winnemucca


Tourist Attractions per City
Average
Number of POIs
0
2
4
6
8
10
Boulder City
Henderson
Reno
City


Average distance between tourist attractions per city
Average distance (km)
0
2
4
6
8
Henderson
Boulder City
Reno
City


Cities by Population in Nebraska
Las Vegas
21.7%
Henderson
10.6%
Reno
8.9%
North Las Vegas
8.9%
Enterprise
7.5%
Spring Valley
7.4%
Sunrise Manor
6.7%
Paradise
6.5%
Sparks
3.7%
Carson City
2.0%
Whitney
Pahrump
Winchester
Ranchos

Instructions for running your code

1. Run hotelcode.py
2. Run the tomtom.py
3. Run the websiteCode.py
4. Run the visualization.py (run it a few times if all cities don't show up)
5. Run the main.py file

Code Documentation

HotelCode.py

1. getYelpApiData()

   The code is using the Yelp API to search for hotels in several cities and create a list of tuples with the hotel name, city, rating, and price (if available). The list is returned at the end of the function.

   Expected output: [("Hotel 1", "Henderson", 4.0, "$$$"), ("Hotel 2", "Henderson", 3.5, "$$"), ...]

2. open_database(db_name)

   This code defines a function called open_database that opens a connection to an SQLite database specified by the provided db_name. It returns two objects: a cursor object (cur) and a connection object (conn). These objects can be used to execute SQL commands on the database.

   Expected output: There is no expected output from this function as it only returns two objects that can be used to interact with an SQLite database.

3. make_yelp_table(data, cur, conn)

   This code defines a function called make_yelp_table which takes three arguments: data, cur, and conn. It creates a table in the SQLite database if it doesn't already exist, and then iterates through the data list and attempts to insert each record into the table. If a record already exists (based on a unique combination of hotel name and city), it is skipped. Finally, it commits the changes to the database.

   Expected output: None.

4. main()

   The main function calls getYelpAPIData to get hotel data for cities in Nevada, then appends it to finalList. The function then calls open_database to connect to an SQLite database and make_yelp_table to create a new table or update an existing one with the hotel data.

Proj3CalcGraph.py

1. open_database(db_name)

   This code defines a function called open_database that opens a connection to an SQLite database specified by the provided db_name. It returns two objects: a cursor object (cur) and a connection object (conn). These objects can be used to execute SQL commands on the database.

   Expected output: There is no expected output from this function as it only returns two objects that can be used to interact with an SQLite database.

2. highestratedhotel(cur, conn)

   The function takes two parameters, a cursor and a connection to an SQLite database. It retrieves data from the YelpData table sorted by city, identifies the highest-rated hotel for each city, stores them in a dictionary, writes them to a file named highest_rated_hotels.txt, and returns a list of tuples containing the highest-rated hotels.

   Expected outcome: The expected outcome is a list of tuples containing the name, rating, city, and cost of the highest rated hotel in each city, as well as a file "highest_rated_hotels.txt" which contains the same information in a formatted table.

3. bargraphHotel(hotelList)

This function takes a list of tuples containing hotel names, ratings, and cities, and creates a bar graph showing the highest rated hotel for each city. It loops through the original list, extracts the relevant data, creates a bar plot using Matplotlib, labels the x- and y-axes, rotates the x-axis labels for better visibility, and displays the graph using plt.show().

Expected outcome: The expected outcome is a bar graph showing the highest rated hotel for each city, where the x-axis represents the city and the y-axis represents the rating.

4. scatterplot(hotelList)

   This function takes in a list of hotels and their ratings by city and creates a scatter plot showing the relationship between hotel ratings and cities. It first extracts the hotel names, ratings, and cities from the input list. Then, it creates a scatter plot with cities on the x-axis and hotel ratings on the y-axis. The x-axis labels are rotated 90 degrees for better visibility.

   Expected output: A scatter plot showing the distribution of hotel ratings by city.

5. main()

   The code establishes a database connection to YelpData.db, then calculates and outputs the highest rated hotel per city to a text file. It also generates a bar graph and a scatter plot based on the highest rated hotel data.

webSiteCode.py

6. open_database(db_name)

   Like the previous file, this function serves to open the database if needed.

7. make_pop_table(cur, conn)

   The function takes two parameters, a cursor and a connection to an SQLite database. It goes on to utilize BeautifulSoup to scrape the website 'https://www.nevada-demographics.com/cities_by_population' to get Nevada Cities and their populations. This information is needed so we can get an understanding of the largest populated non-vegas destination for those who like a crowd, or the lowest crowd for those

who would like some peace in their vacation in Nevada. It then loads each row into the not_sin_city.db database in the populations table for later use.

Resources

Date	Issue Description	Location of Resource	Result (Did it solve the issue?)
04/06/23	Finding an API that can show hotel ratings. The first one didn't work	https://docs.developer.yelp.com/reference/v3_business_search https://docs.developer.yelp.com/docs/fusion-intro	Yes
4/07/23	Learning about parameters	https://docs.developer.yelp.com/reference/v3_business_search	yes
4/10/23	Wanted to check if we have duplicate data	https://stackoverflow.com/questions/42334197/add-only-unique-values-to-a-list-in-	yes

		python	
4/18/23	Learning how to do a bar plot	https://www.geeksforgeeks.org/bar-plot-in-matplotlib/	yes
4/21/23	Try and except for UNIQUE method for duplicates	https://stackoverflow.com/questions/35516130/how-to-catch-unique-constraint-failed-auth-user-username-error	yes
4/21/23	Avoiding adding duplicate city name into table	https://www.tutorialspoint.com/mysql/mysql-handling-duplicates.htm	yes
4/21/23	Pie chart in matplotlib?	https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html	yes