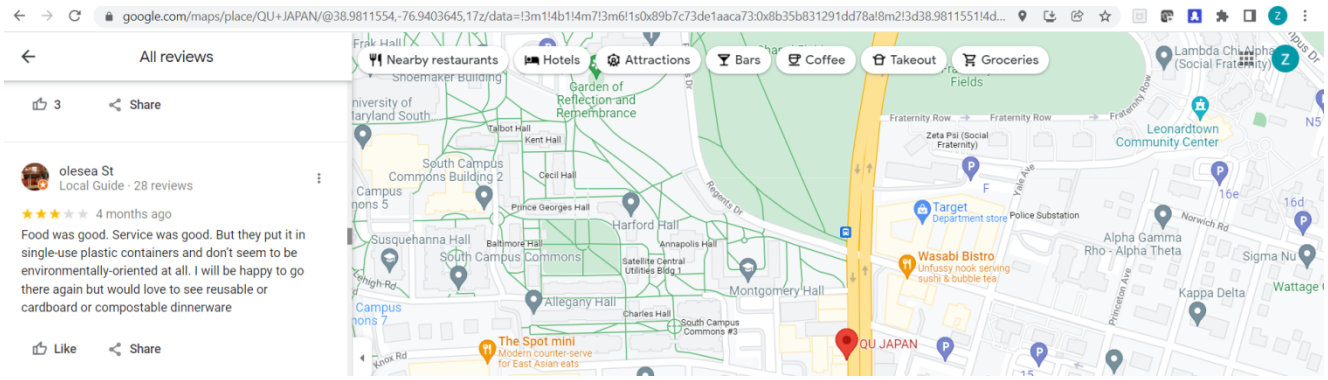
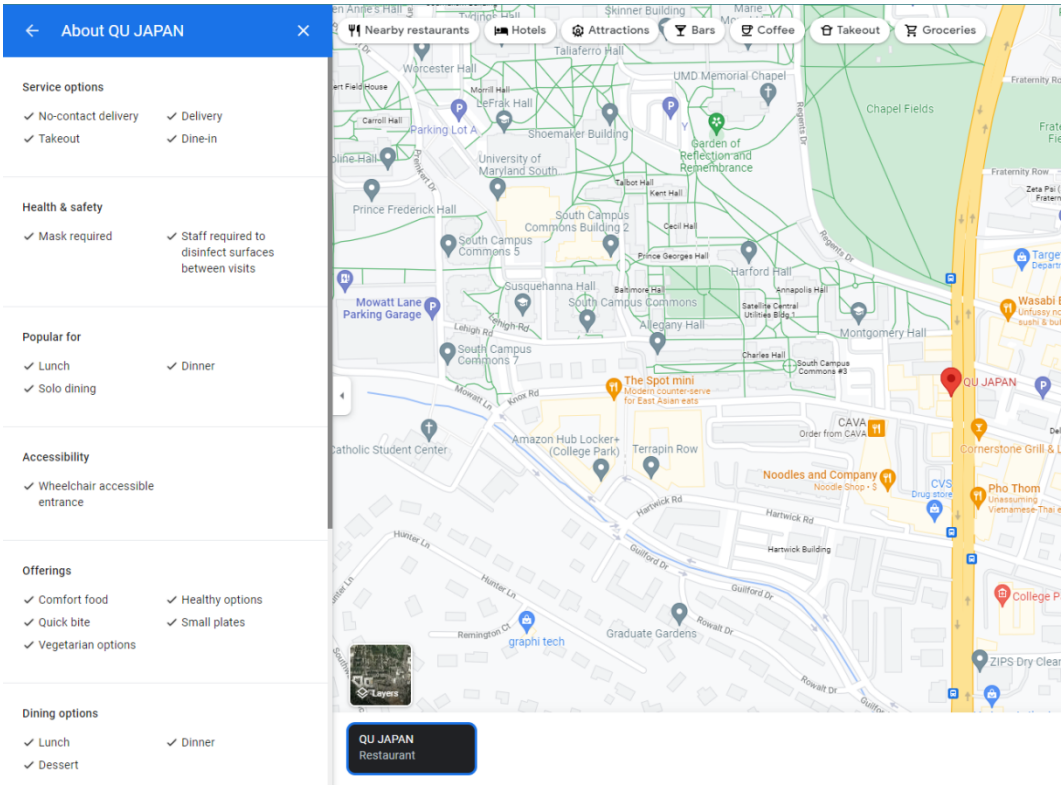
Readme Document

Data sources with Screenshots of sample Reviews:

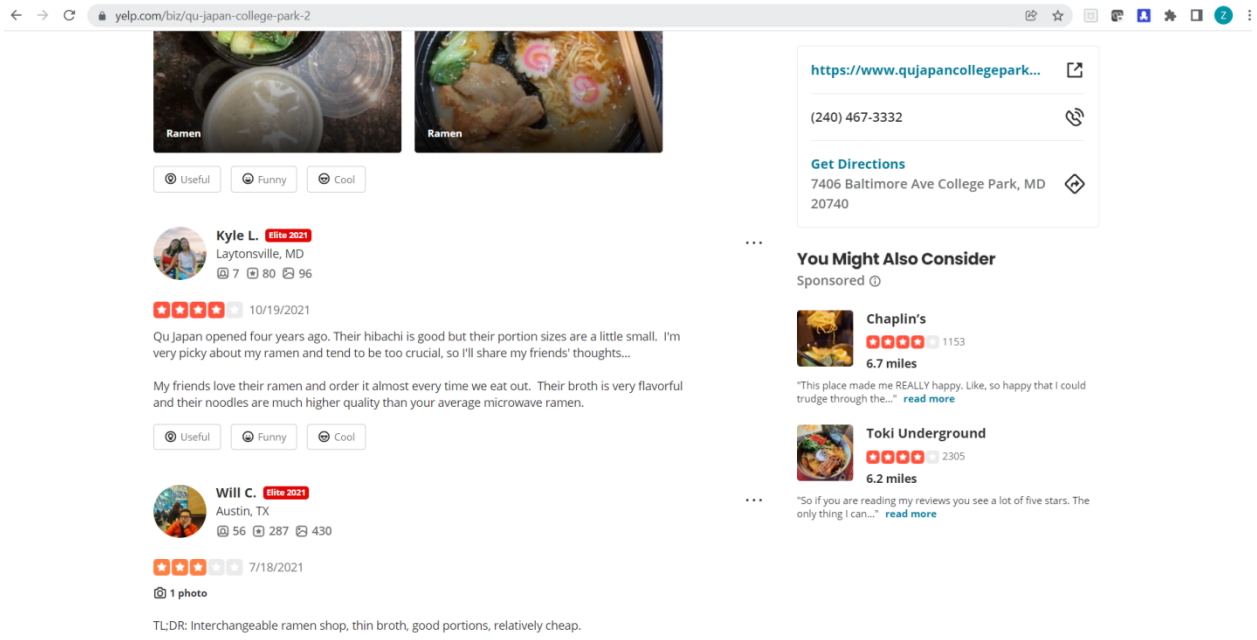
Our group used three platforms to gather restaurant information and reviews and insert them as data within our database. These three platforms are Google Maps, Yelp, and Trip Advisor. We have attached screenshots showing examples of reviews for the restaurant ‘Qu Japan’ we have taken from each data source.

Google Maps https://www.google.com/maps :





Yelp https://www.yelp.com/ :

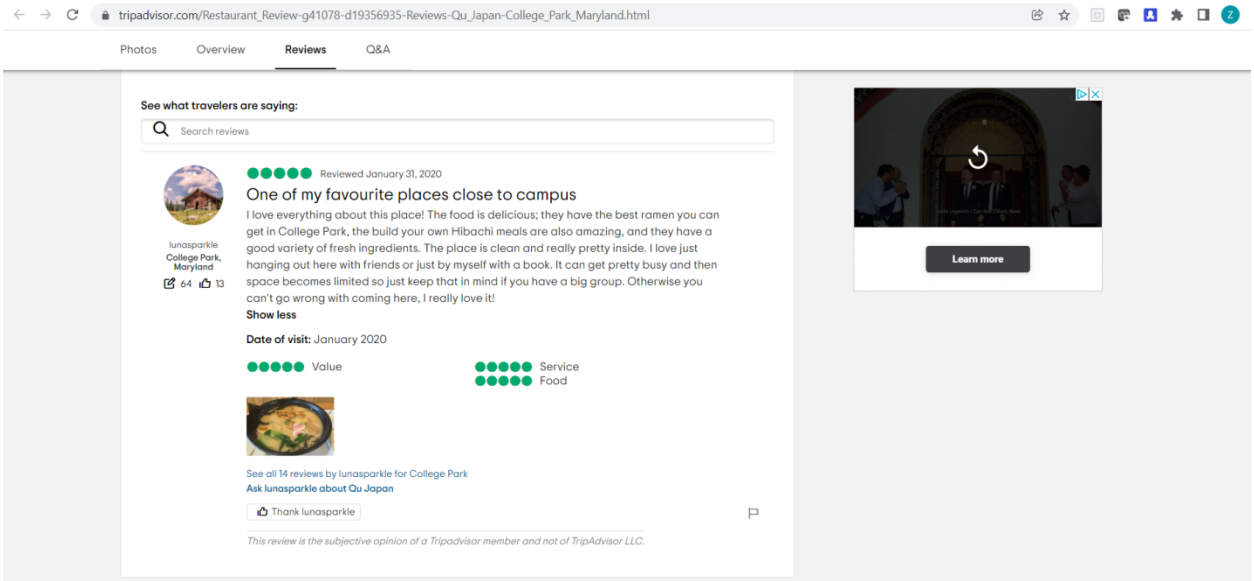


Amenities and More

- Offers Delivery
- Offers Takeout
- No Reservations
- Masks required
- Staff wears masks
- Soy-Free Options
- Vegan Options
- Many Vegetarian Options
- Accepts Credit Cards
- Casual
- Moderate Noise
- Good for Groups
- Good For Kids
- Good for Dinner
- Street Parking
- Free Wi-Fi
- Covered outdoor seating
- No Outdoor Seating
- Offers Catering
- No Waiter Service
- No Happy Hour
- No Alcohol
- Dogs Not Allowed
- Not Wheelchair Accessible
- No TV

Show Less

Trip Advisor https://www.tripadvisor.com/ :



Steps to test the Project:

1. Run the create statements in the 'Project_0501_12_DDL.sql' file, to create the Customer, Restaurant, Platform , Amenity, Cuisine and Review tables .
2. Run the INSERT statements in the above file once the tables are created in the same order to insert the data.
3. Run the 'Project_0501_12_DML.sql' file to execute the queries in order, to get the output for each use case.

Application:

Use Case #1: To find top five restaurants by average review rating to help customers pick the best place to eat

```
GO
CREATE VIEW
Restaurants AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId
GROUP BY rs.rstName
GO

SELECT TOP 5 * FROM Restaurants
ORDER BY 'Average Review Rating' DESC
```

	Restaurant Name	Average Review Rating
1	LaTao Hotpot	5.000000
2	Sarku Japan	5.000000
3	Seoul Spice	5.000000
4	Saburo Ramen Bar	4.500000
5	The Spot Mini	4.500000

For customers like students and families near College Park, who would want to make a quick decision by finding some of the best places to dine, we filter the restaurants by finding the top five restaurants by average reviews.

We do this using the Restaurant and Review table, by using restaurant ID to map the restaurant with its reviews using the 'WHERE' function. We then find an average of all the Review ratings for a particular restaurant and select these two data in the columns restaurant name and average Rating. We create a view for the above selection and then the 'ORDER BY' function for the average rating column to select the top five restaurants.

Result: From the output we suggest our customers to pick from one of the top rated restaurants-'LaTao Hotpot', 'Sarku Japan', 'Seoul Spice' depending on the cuisine they prefer.To choose the cuisine, Customers can filter the above suggested restaurants using query from UseCase #3.

Use Case #2: For the same restaurant, what are the average review ratings across different platforms?

View 1:
To show all the restaurants with review ratings across different platforms

```
GO
CREATE VIEW
ratings_across_multiple_platforms_view_1 AS

SELECT rs.rstName AS 'Restauarant Name', p.pltName AS 'Platform Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM  [EasyReview.Restaurant] rs, [EasyReview.Review] rv , [EasyReview.Platform] p
WHERE (p.pltId = rv.pltId) AND rs.rstId = rv.rstId
GROUP BY rs.rstName, p.pltName
GO

SELECT * FROM ratings_across_multiple_platforms_view_1
ORDER BY 'Restauarant Name', 'Average Review Rating' DESC
```

	Restauarant Name	Platform Name	Average Review Rating
1	Hanami	Tripsdvisor	4.000000
2	Kangnam BBQ	Google Map	4.000000
3	LaTao Hotpot	Google Map	5.000000
4	Nuvegan Cafe	Tripsdvisor	4.000000
5	Potbelly Sandwich Shop	Tripsdvisor	4.000000
6	Qu Japan	Tripsdvisor	5.000000
7	Qu Japan	Google Map	4.000000
8	Qu Japan	Yelp	3.666666
9	Saburo Ramen Bar	Google Map	4.500000
10	Sarku Japan	Google Map	5.000000
11	Sarku Japan	Tripsdvisor	5.000000
12	Sarku Japan	Yelp	5.000000
13	Seoul Spice	Yelp	5.000000
14	The Spot Mini	Google Map	5.000000
15	The Spot Mini	Yelp	4.000000
16	Wasabi Bistro	Yelp	3.000000
17	Wasabi Bistro	Google Map	1.000000

View 2:

To show only the restaurants with at least one review rating across different platforms.

It makes sense to only show those restaurants that have reviews across multiple platforms for customers to compare the ratings.

```
GO
CREATE VIEW
ratings_across_multiple_platforms_view_2 AS
SELECT rs.rstName AS 'Restauarant Name', p.pltName AS 'Platform Name', AVG(rv.rvwRating) as 'Average Review Rating'
FROM  [EasyReview.Restaurant] rs, [EasyReview.Review] rv , [EasyReview.Platform] p
WHERE (p.pltId = rv.pltId) AND rs.rstId = rv.rstId
GROUP BY  p.pltName, rs.rstName
HAVING rs.rstName IN (
            SELECT V.rstName
            FROM
            (
                SELECT rs.rstName , p.pltName, AVG(rv.rvwRating) as 'Average Review Rating'
                FROM  [EasyReview.Restaurant] rs, [EasyReview.Review] rv , [EasyReview.Platform] p
                WHERE (p.pltId = rv.pltId) AND rs.rstId = rv.rstId
                GROUP BY  p.pltName, rs.rstName
            ) V
            GROUP BY (V.rstName)
            HAVING  COUNT(V.rstName) >1
        )
GO
```

```
SELECT * FROM ratings_across_multiple_platforms_view_2
ORDER BY 'Restauarant Name', 'Average Review Rating' DESC
```

	Restauarant Name	Platform Name	Average Review Rating
1	Qu Japan	Tripsdvisor	5.000000
2	Qu Japan	Google Map	4.000000
3	Qu Japan	Yelp	3.666666
4	Sarku Japan	Google Map	5.000000
5	Sarku Japan	Tripsdvisor	5.000000
6	Sarku Japan	Yelp	5.000000
7	The Spot Mini	Google Map	5.000000
8	The Spot Mini	Yelp	4.000000
9	Wasabi Bistro	Yelp	3.000000
10	Wasabi Bistro	Google Map	1.000000

A prospective customer is cautious about the quality of food that he/she wishes to spend on.
To avoid customers making decisions on skewed review ratings by one of the platforms, we have built a query that displays average review ratings across multiple platforms for a restaurant.

We use the Restaurant, Review, and Platform table, using the 'WHERE' function to link them together. We then group by restaurant name and platform name to show the different average rating across different platforms for the same restaurant. In the second view, a subquery is created. Then by using the 'HAVING' clause, we pick only those restaurants that have reviews for more than platforms. We used a subquery to get the names of these restaurants.

Based on the output, we can see that in the case of 'Sarku Japan' all the three platforms have the same ratings. A customer can be confident to make a decision based on this rating. While in the case of 'Qu Japan' and 'Wasabi Bistro', the reviews looked to be skewed in one of the platforms. The customer might have to do more research in this case to make their decision. By this we prevent customers making decisions based on biased ratings in a particular platform.

Also, in case of ambiguity in ratings across multiple platforms, if the customer has preference for a particular platform, then the customer can make his decisions based on the review rating on that particular platform.

Use Case #3: To show all the restaurants for "Japanese" cuisine ordered by average review ratings

```
GO
CREATE VIEW
Japanese_Restaurant AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.RestaurantCuisine] c, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId AND c.rstCuisine = 'Japanese' AND c.rstId = rs.rstId
GROUP BY rs.rstName
GO

SELECT * FROM Japanese_Restaurant
ORDER BY 'Average Review Rating' DESC
```

	Restaurant Name	Average Review Rating
1	Sarku Japan	5.000000
2	The Spot Mini	4.500000
3	Saburo Ramen Bar	4.500000
4	Hanami	4.000000
5	Qu Japan	4.000000
6	Wasabi Bistro	2.000000

This view shows all the restaurants for a cuisine type based on review ratings. Take Japanese restaurants for example. We use Restaurant, Cuisine, and Review tables, use the “WHERE” function to link different tables, and set the cuisine type as “Japanese”. For better application, we use the “ORDER BY” function to show the result on average review rating. Based on the result, users can pick a high-rating restaurant (‘Sarku Japan’) or try a new restaurant.

The following is the application of Korean restaurants making use of the above query.

```
GO
CREATE VIEW
Korean_Restaurant AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.RestaurantCuisine] c, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId AND c.rstCuisine = 'Korean' AND c.rstId = rs.rstId
GROUP BY rs.rstName
GO

SELECT * FROM Korean_Restaurant
ORDER BY 'Average Review Rating' DESC
```

	Restaurant Name	Average Review Rating
1	Seoul Spice	5.000000
2	Kangnam BBQ	4.000000
3	Nuvegan Cafe	4.000000

We can edit the query for whatever cuisine the customer is interested in. For example, if the customer wanted to eat “Korean” instead of “Japanese” cuisine, we can edit the query as we have shown, selecting restaurants only with “Korean” cuisine. Based on our database, we would recommend customers who wanted to eat “Korean” cuisine to visit Seoul Spice.

Use Case #4: What are the restaurants that have reservations, parking, and are good for groups ordered by average review ratings?

```
GO
CREATE VIEW Restaurants_with_amenities AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.Amenity] a, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId AND a.reservations = 1 AND a.parking = 1 AND a.goodForGroups = 1 AND a.rstId = rs.rstId
GROUP BY  rs.rstName
GO


SELECT * FROM Restaurants_with_amenities
ORDER BY 'Average Review Rating' DESC
```

	Restaurant Name	Average Review Rating
1	LaTao Hotpot	5.000000
2	Hanami	4.000000
3	Kangnam BBQ	4.000000

This use case looks for the restaurants that are good for groups, takes reservations, and has parking. As our target audience is college students, many times big groups of college students want to go to a restaurant and eat with each other, so finding the restaurants most accommodating for large groups of people would be helpful to them.

To do this, we use the Restaurant, Review, and Amenity table. By using the ‘WHERE’ function, we link the three tables on ‘rstId’ and only select restaurants that have the amenity of reservations, parking, and are good for groups. We then find an average of the review ratings for these restaurants, and display the restaurant name and the average review rating. After creating this view, we use the ‘ORDER BY’ function to order the results and show the highest average rated restaurant at the top of our query result.

From the result of our query, we can advise large groups of people to visit the restaurant LaTao Hotspot as it had the highest average rating for restaurants with the amenity of reservations, parking, good for groups based on the reviews in our database.

The following is an application of the vegan option amenity making use of the above query

```
GO
CREATE VIEW Vegan_Restaurants AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.Amenity] a, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId AND a.veganOptions = 1 AND a.rstId = rs.rstId
GROUP BY  rs.rstName
GO


SELECT * FROM Vegan_Restaurants
ORDER BY 'Average Review Rating' DESC
```

	Restaurant Name	Average Review Rating
1	LaTao Hotpot	5.000000
2	Sarku Japan	5.000000
3	Seoul Spice	5.000000
4	The Spot Mini	4.500000
5	Nuvegan Cafe	4.000000
6	Potbelly Sandwich Shop	4.000000
7	Qu Japan	4.000000
8	Hanami	4.000000
9	Kangnam BBQ	4.000000
10	Wasabi Bistro	2.000000

We can edit the query to help customers who want to find restaurants with different amenities. For example, if a customer wanted to see all the restaurants with vegan options, the query can be edited so that only restaurants with the amenity vegan options are shown. Based on our database, we would recommend customers who want vegan options to visit LaTao Hotpot, Sarku Japan, or Seoul Spice.

Future Work:

This project can be enhanced further to achieve more, we have identified a few features by which the database can be made robust.

1. Gathering attributes such as restaurant hours, customer age, average delivery time, etc. will allow the database to give more accurate and relevant recommendations to residents within College Park.
2. Database can be enhanced to identify fake reviews using machine learning.
3. Review comments can be processed and analyzed using natural language processing to add more filters for customers to choose from. This will enable customers to search for restaurants with reviews about sea-facing ambience, night life, spicy food provided these words are frequently used in the comments about the restaurant.
4. To have more descriptive statistics data from platforms, we can have more analyses. For example,

we can use customer age to analyze the preference of cuisine type among different age levels.

5. Gathering more review data from different platforms. If our database contained more review information, the recommendations would be more accurate and provide more value to customers.

References:

https://www.w3schools.com/sql/sql_top.asp

https://www.easytechjunkie.com/what-are-associative-entities.htm