Easy Review

Group 12

Chia-Lin Tsai
Krishik Nataraj Gowda
Nishant Jadhav
Zaid Dar

12/11/2021

Mission statement

To create a database for reviews of restaurants from multiple platforms in the College Park area and to analyze information across different categories to provide recommendations for restaurants in the College Park area

Background

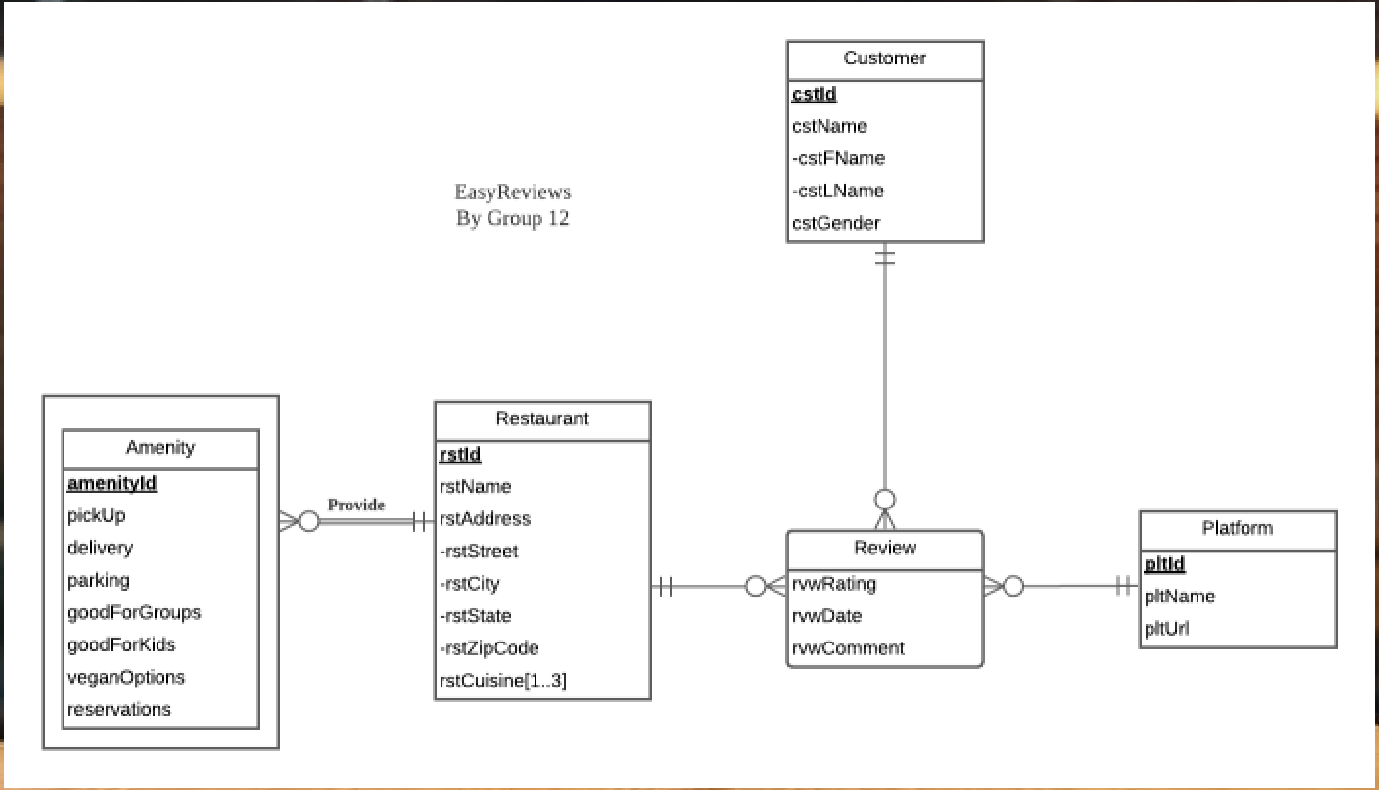
Users: Customers, Restaurant Owners

Data Sources: Yelp, Google Maps, TripAdvisor

Mission Objectives

1. To find the top five restaurants by average review rating to help customers pick a place to eat.
2. For the same restaurant, what are the average review ratings across different platforms?
3. To show the best restaurants for a specific cuisine based on review ratings.
4. What are the restaurants that have reservations, parking, and are good for groups ordered by average review rating?

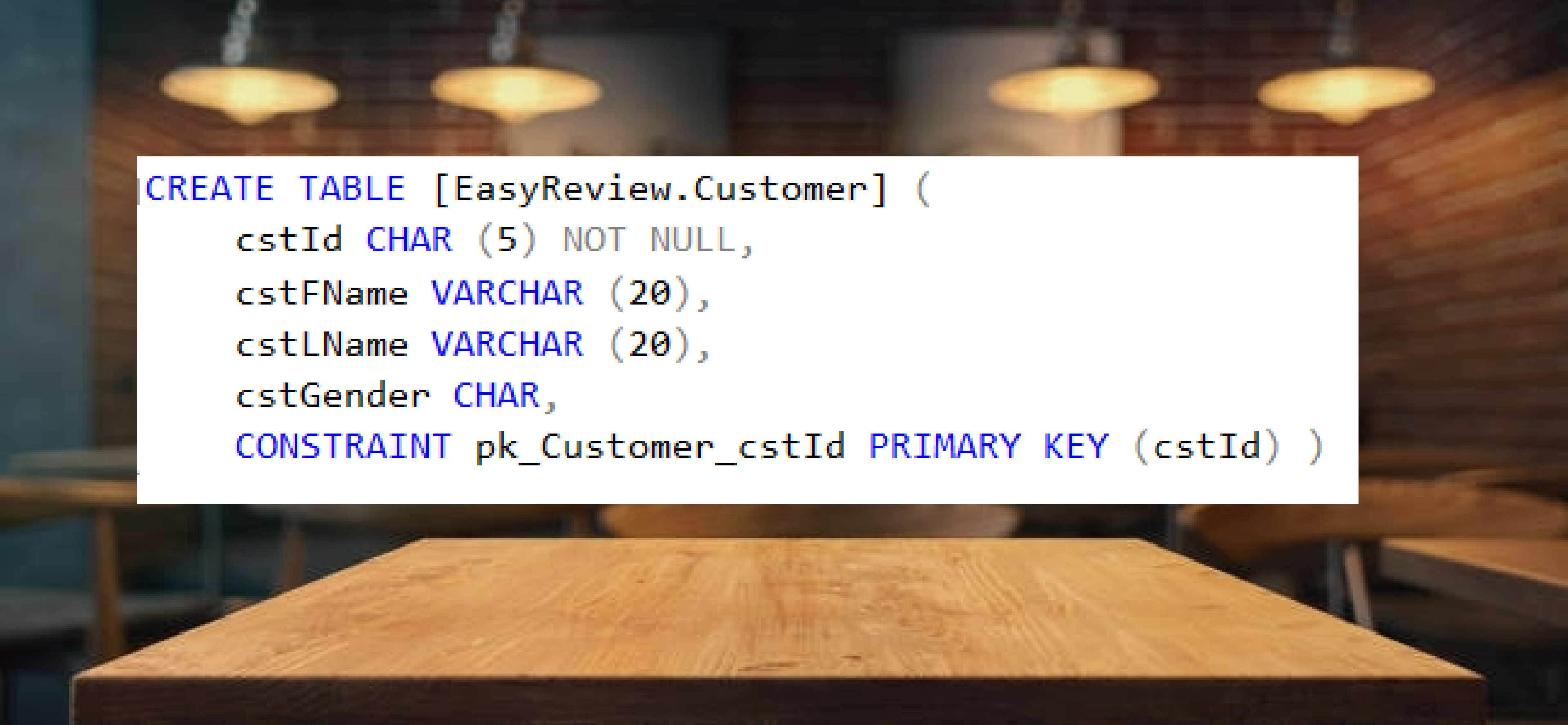
Conceptual Database Design - ER DIAGRAM

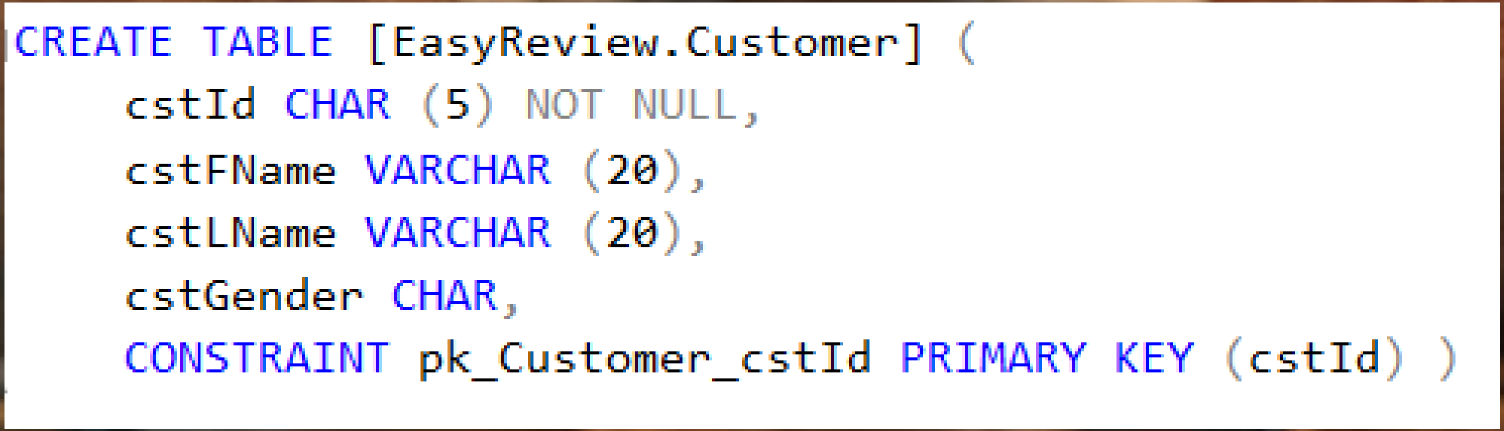


Logical Database Design:

- Customer(**cstId**, cstFName, cstLName, cstGender)
- Restaurant(**rstId**, rstName, rstStreet, rstCity, rstState, rstZipCode)
- RestaurantCuisine(**rstCuisine**, ***rstId***)
- Amenity(**amenityId**, ***rstId***, pickUp, delivery, parking, goodForGroups, goodForKids, veganOptions, reservations)
- Platform(**pltId**, pltName, pltUrl)
- Review(***pltId***, ***cstId***, ***rstId***, rvwRating, rvwDate, rvwComment)

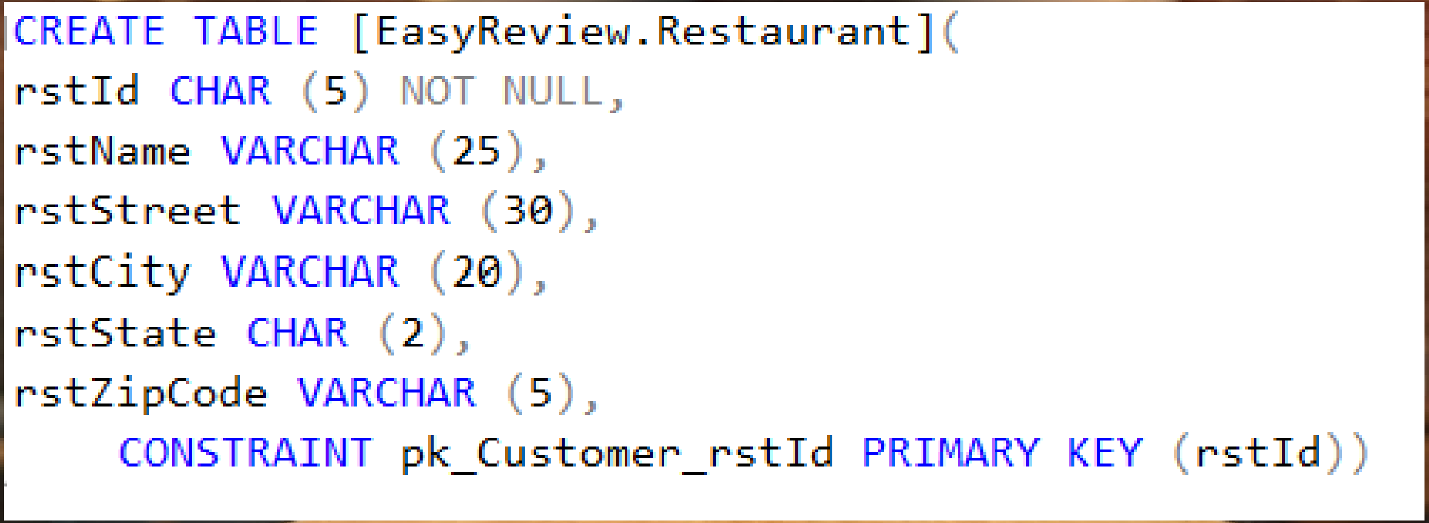
Physical Database Design - Customer Table





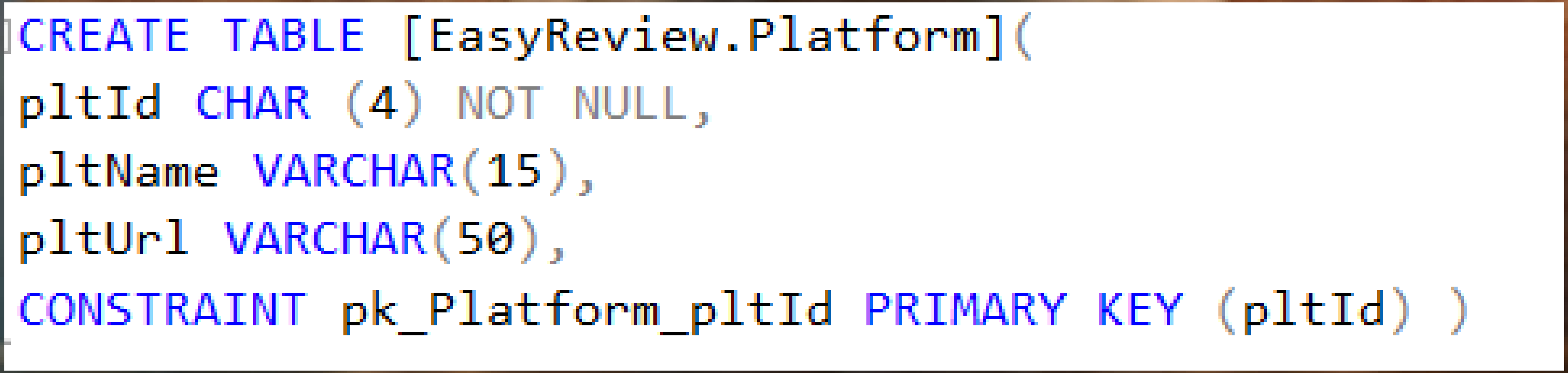
```
CREATE TABLE [EasyReview.Customer] (
    cstId CHAR (5) NOT NULL,
    cstFName VARCHAR (20),
    cstLName VARCHAR (20),
    cstGender CHAR,
    CONSTRAINT pk_Customer_cstId PRIMARY KEY (cstId) )
```

Physical Database Design - Restaurant Table



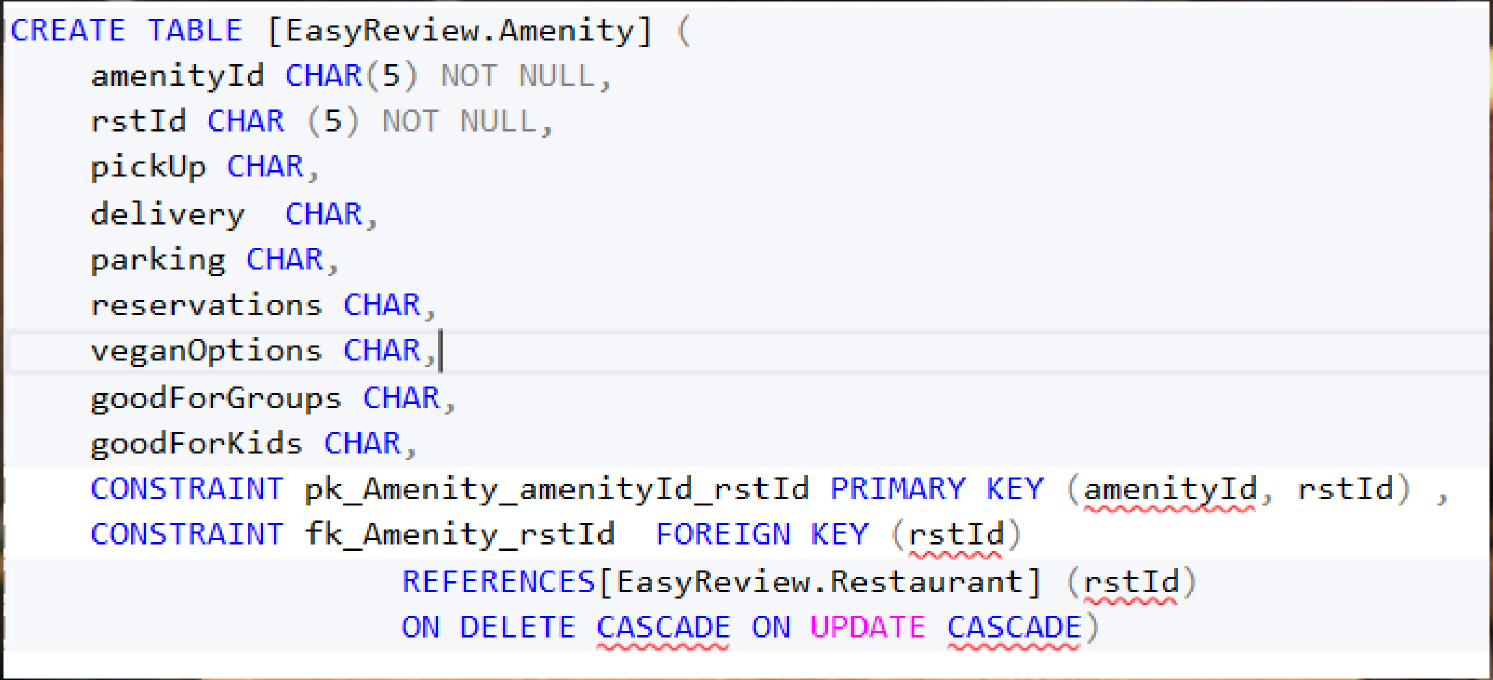
```
CREATE TABLE [EasyReview.Restaurant](
rstId CHAR (5) NOT NULL,
rstName VARCHAR (25),
rstStreet VARCHAR (30),
rstCity VARCHAR (20),
rstState CHAR (2),
rstZipCode VARCHAR (5),
    CONSTRAINT pk_Customer_rstId PRIMARY KEY (rstId))
```

Physical Database Design - Platform Table



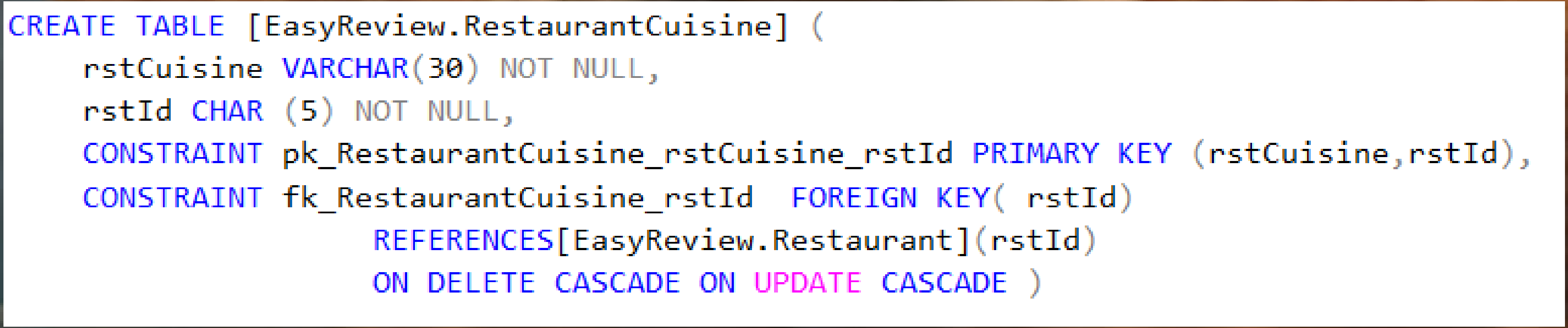
```
CREATE TABLE [EasyReview.Platform](
pltId CHAR (4) NOT NULL,
pltName VARCHAR(15),
pltUrl VARCHAR(50),
CONSTRAINT pk_Platform_pltId PRIMARY KEY (pltId) )
```

Physical Database Design - Amenity Table



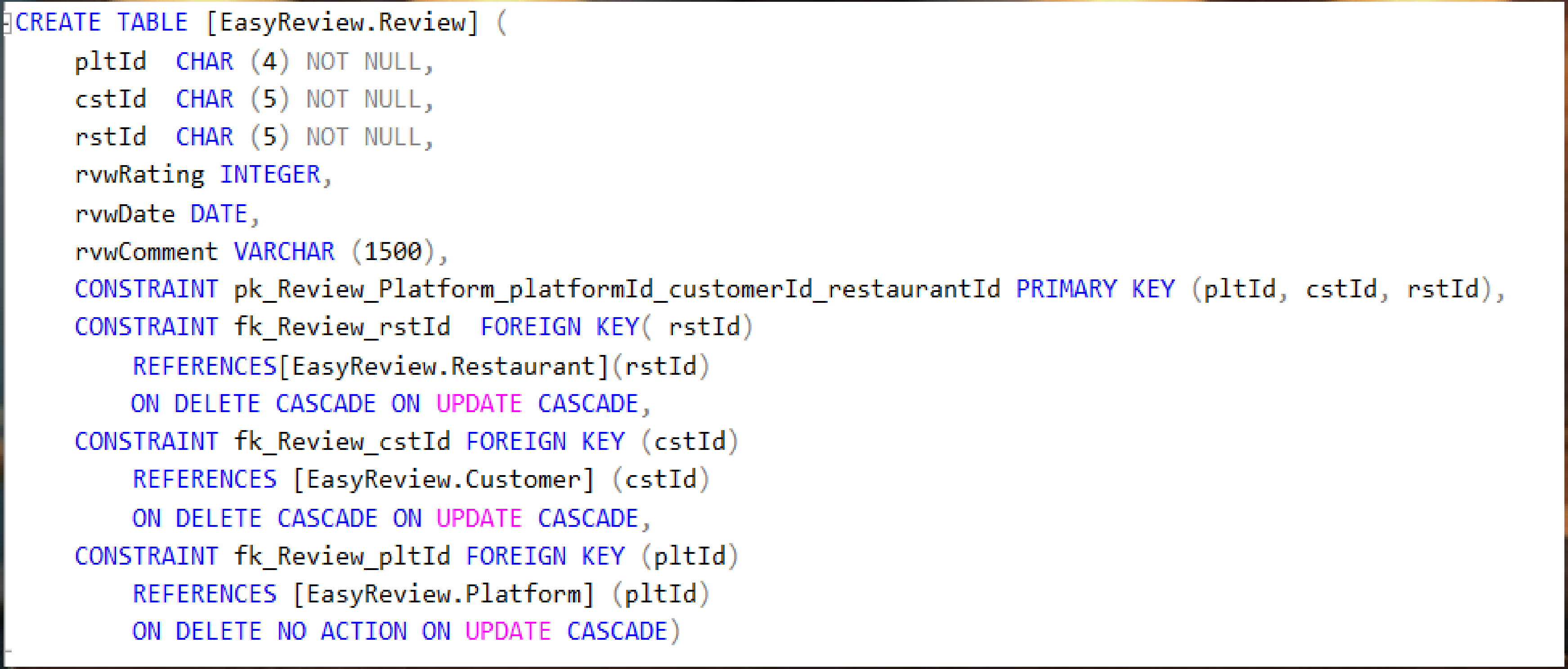
```
CREATE TABLE [EasyReview.Amenity] (
    amenityId CHAR(5) NOT NULL,
    rstId CHAR (5) NOT NULL,
    pickUp CHAR,
    delivery  CHAR,
    parking CHAR,
    reservations CHAR,
    veganOptions CHAR,
    goodForGroups CHAR,
    goodForKids CHAR,
    CONSTRAINT pk_Amenity_amenityId_rstId PRIMARY KEY (amenityId, rstId) ,
    CONSTRAINT fk_Amenity_rstId  FOREIGN KEY (rstId)
                    REFERENCES[EasyReview.Restaurant] (rstId)
                    ON DELETE CASCADE ON UPDATE CASCADE)
```

Physical Database Design - Cuisine Table



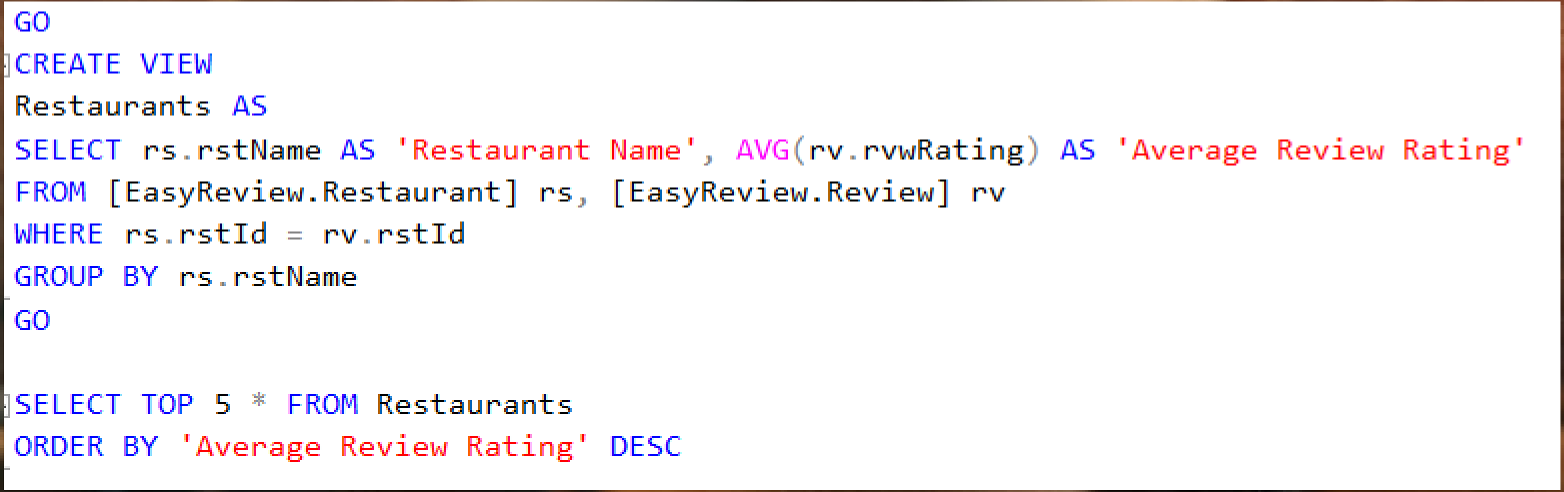
```
CREATE TABLE [EasyReview.RestaurantCuisine] (
    rstCuisine VARCHAR(30) NOT NULL,
    rstId CHAR (5) NOT NULL,
    CONSTRAINT pk_RestaurantCuisine_rstCuisine_rstId PRIMARY KEY (rstCuisine,rstId),
    CONSTRAINT fk_RestaurantCuisine_rstId  FOREIGN KEY( rstId)
                    REFERENCES[EasyReview.Restaurant](rstId)
                    ON DELETE CASCADE ON UPDATE CASCADE )
```

Physical Database Design - Review Table



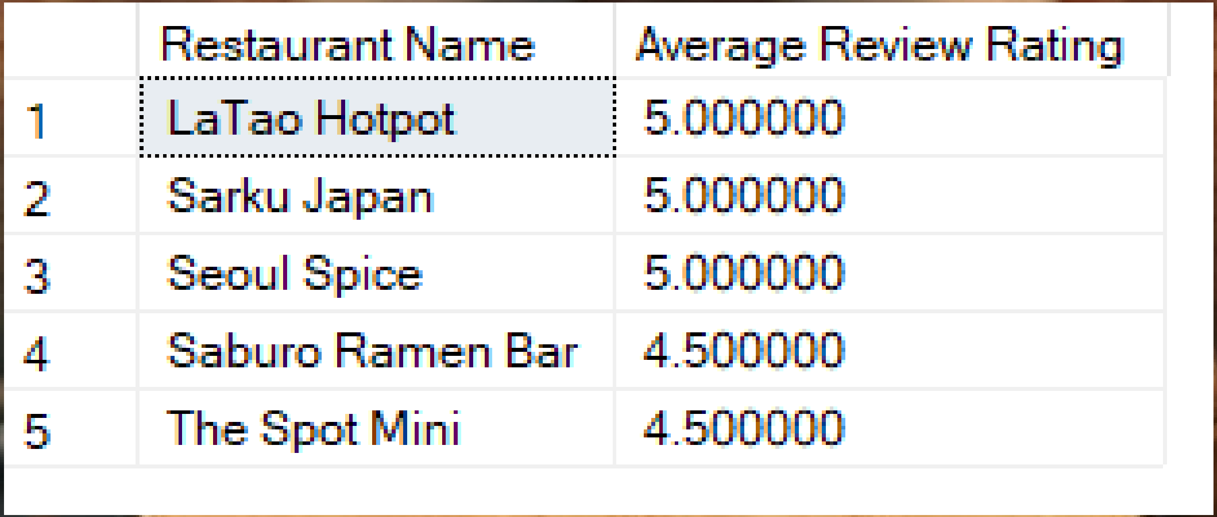
```
CREATE TABLE [EasyReview.Review] (
    pltId  CHAR (4) NOT NULL,
    cstId  CHAR (5) NOT NULL,
    rstId  CHAR (5) NOT NULL,
    rvwRating INTEGER,
    rvwDate DATE,
    rvwComment VARCHAR (1500),
    CONSTRAINT pk_Review_Platform_platformId_customerId_restaurantId PRIMARY KEY (pltId, cstId, rstId),
    CONSTRAINT fk_Review_rstId  FOREIGN KEY( rstId)
        REFERENCES[EasyReview.Restaurant](rstId)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_Review_cstId FOREIGN KEY (cstId)
        REFERENCES [EasyReview.Customer] (cstId)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_Review_pltId FOREIGN KEY (pltId)
        REFERENCES [EasyReview.Platform] (pltId)
        ON DELETE NO ACTION ON UPDATE CASCADE)
```

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



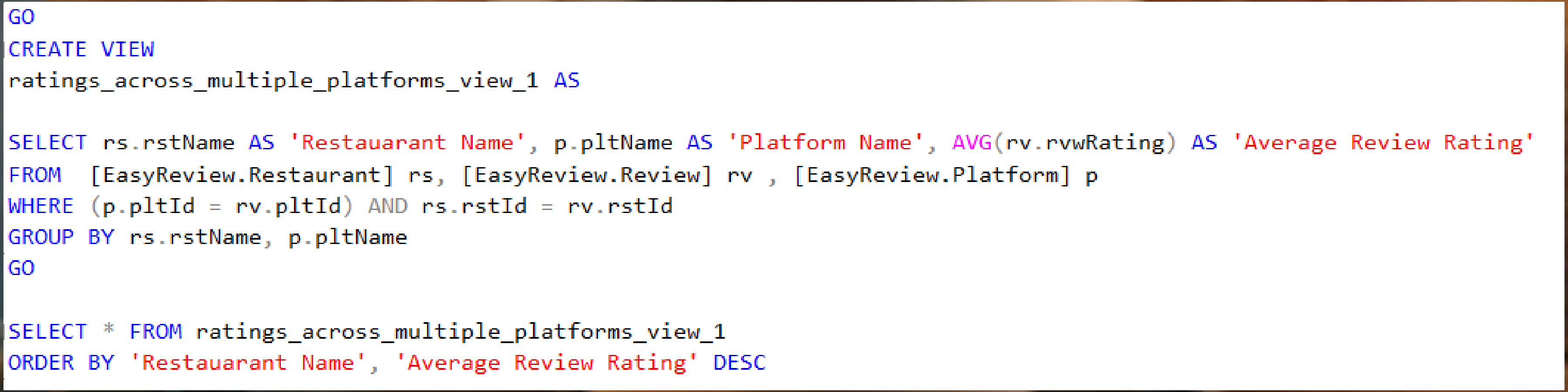
```
GO
CREATE VIEW
Restaurants AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId
GROUP BY rs.rstName
GO

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

Application #1 - To find top five restaurants by average review rating to help customers pick the best place to eat



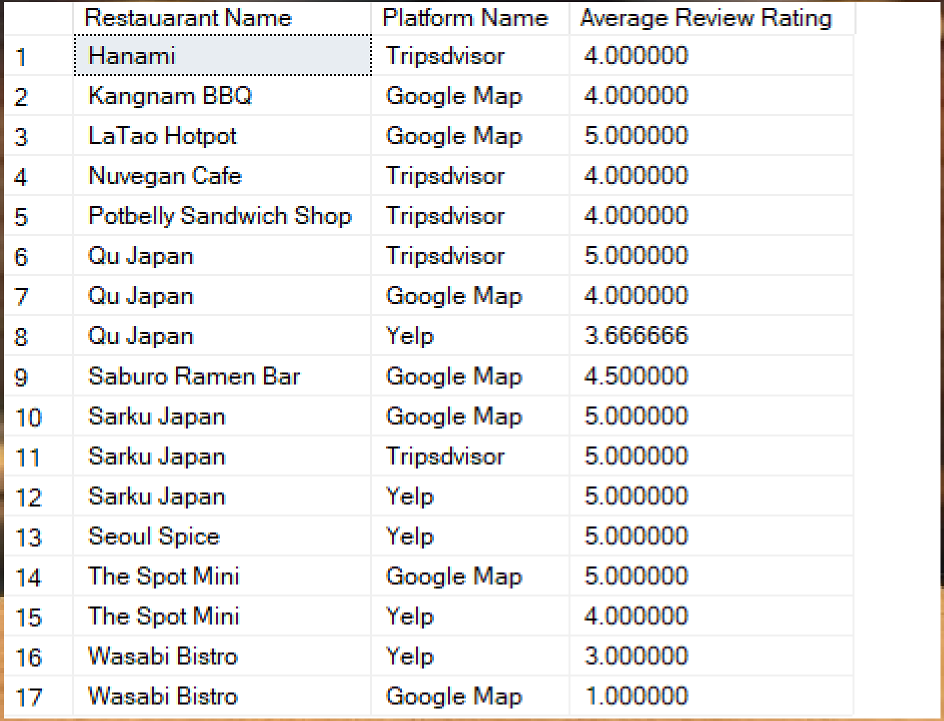
	Restaurant Name	Average Review Rating
1	LaTao Hotpot	5.000000
2	Sarku Japan	5.000000
3	Seoul Spice	5.000000
4	Saburo Ramen Bar	4.500000
5	The Spot Mini	4.500000

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



```sql
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
```

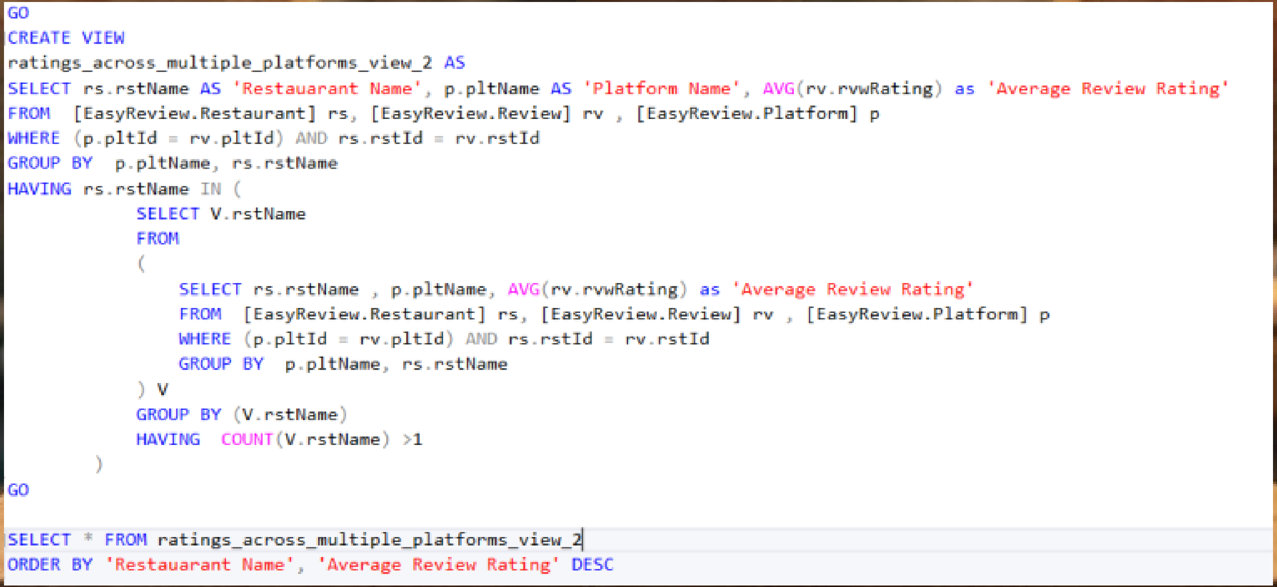
Application #2 - For the same restaurant, what are the average review ratings across different platforms?



	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

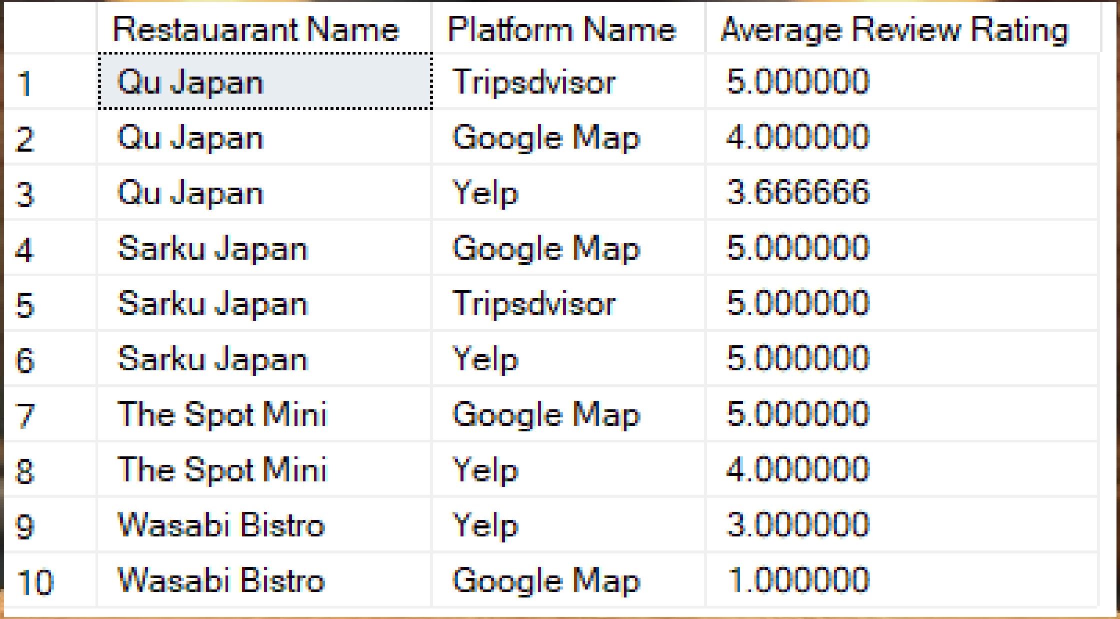
Use Case #2 - For the same restaurant, what are the average review ratings across different platforms?

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



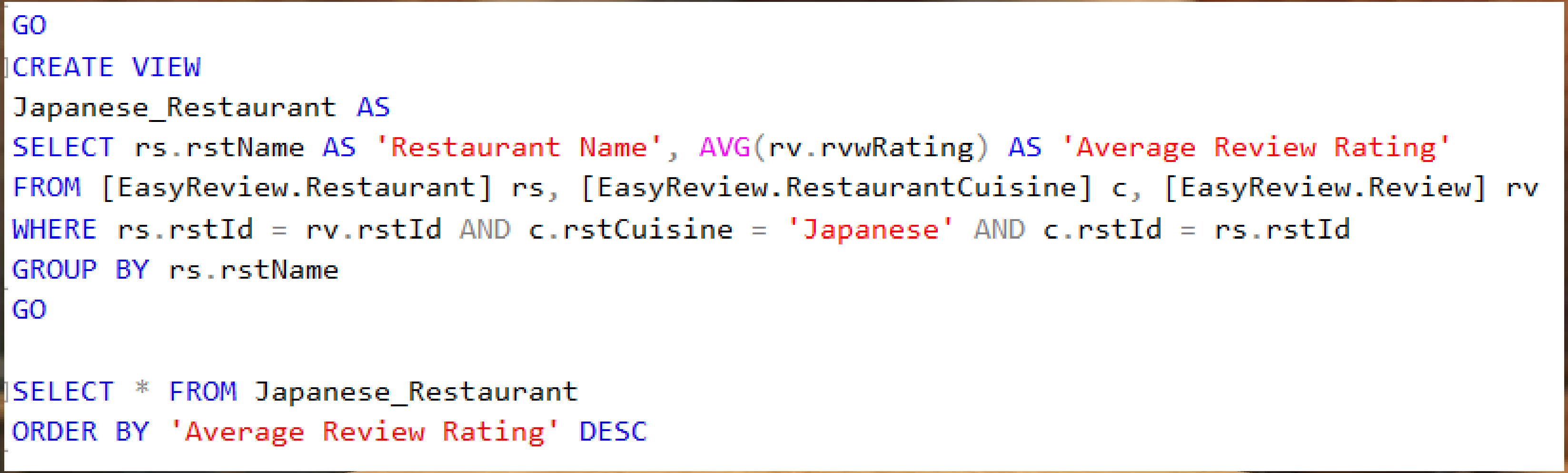
```sql
GO
CREATE VIEW
ratings_across_multiple_platforms_view_2 AS
SELECT rs.rstName AS 'Restauarant Name', p.pltName AS 'Platform Name', AVG(rv.rvwRating) as 'Average Review Rating'
FROM  [EasyReview.Restaurant] rs, [EasyReview.Review] rv , [EasyReview.Platform] p
WHERE (p.pltId = rv.pltId) AND rs.rstId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

Application #2 - For the same restaurant, what are the average review ratings across different platforms?



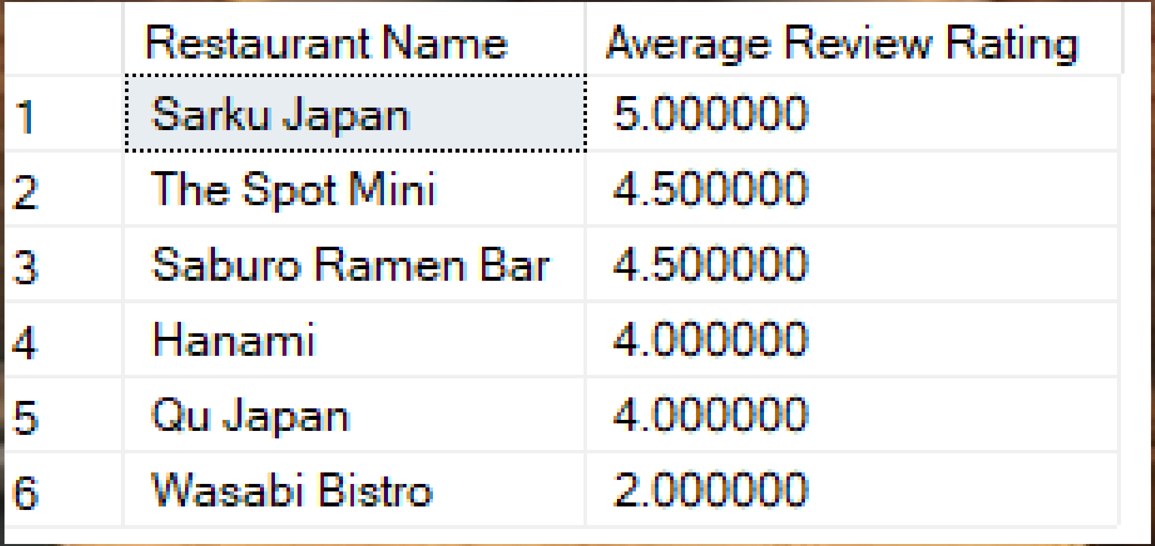
	Restauarant Name	Platform Name	Average Review Rating
1	Qu Japan	Tripsdvisor	5.000000
2	Qu Japan	Google Map	4.000000
3	Qu Japan	Yelp	3.666666
4	Sarku Japan	Google Map	5.000000
5	Sarku Japan	Tripsdvisor	5.000000
6	Sarku Japan	Yelp	5.000000
7	The Spot Mini	Google Map	5.000000
8	The Spot Mini	Yelp	4.000000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

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



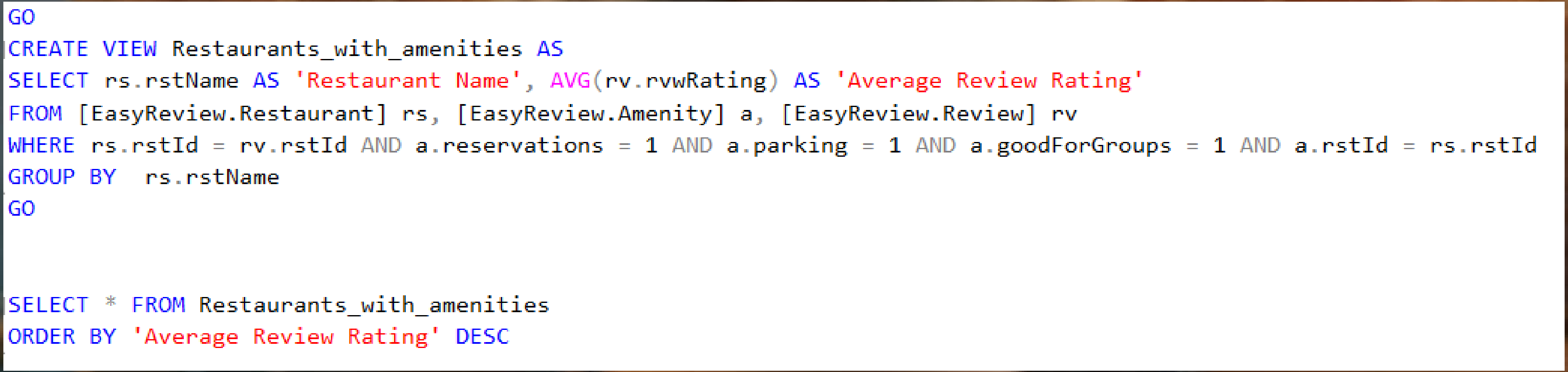
```
GO
CREATE VIEW
Japanese_Restaurant AS
SELECT rs.rstName AS 'Restaurant Name', AVG(rv.rvwRating) AS 'Average Review Rating'
FROM [EasyReview.Restaurant] rs, [EasyReview.RestaurantCuisine] c, [EasyReview.Review] rv
WHERE rs.rstId = rv.rstId AND c.rstCuisine = 'Japanese' AND c.rstId = rs.rstId
GROUP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

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

Application #3 -
To show all the restaurants for "Japanese" cuisine ordered by average review ratings



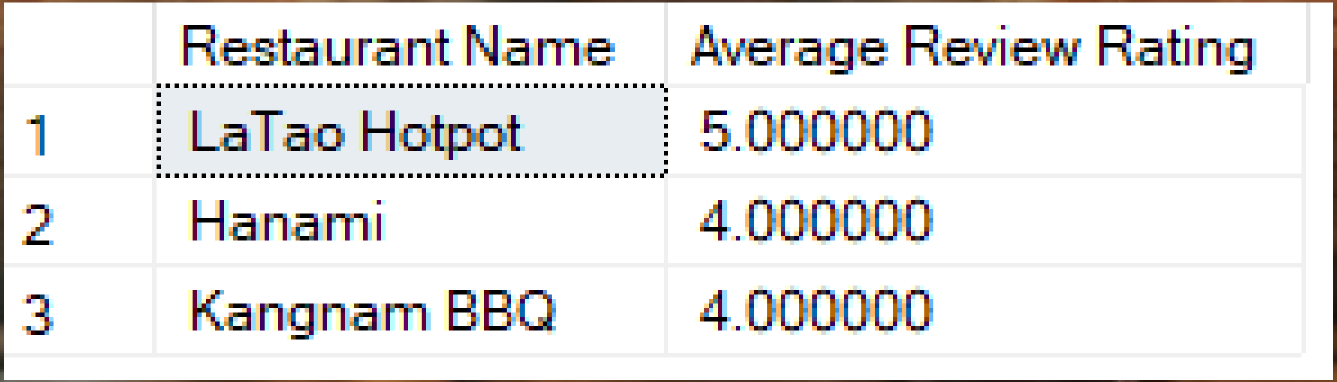
	Restaurant Name	Average Review Rating
1	Sarku Japan	5.000000
2	The Spot Mini	4.500000
3	Saburo Ramen Bar	4.500000
4	Hanami	4.000000
5	Qu Japan	4.000000
6	Wasabi Bistro	2.000000

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
```

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



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

Future work

- 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.
- Database can be enhanced to identify fake reviews using machine learning.
- 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.

Future work

- 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.
- 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.

Thank you