Project Report: List and Description

Course Project: Campus Eats

Team Members: Akhila Sirikonda, Sanket Gaikwad, Hanisha Shaik, Rishitha Reddy Gaddam

Project Description: Food delivery services has become one of the concerning sectors due to the Corona virus. Instead of hiring external people to deliver the food, local businesses are looking for simple ways to deliver the food to consumers. Even when things that were affected by Corona virus are returning to normal, many experts believe that food delivery will have become a part of our daily routine. Students prefer food delivery services while present in campus premises. University personnel dislike the constant influx of guests who may or may not be affiliated with the university. Although companies like UberEats and GrubHub are delighted to deliver on campus, many colleges are debating whether they should take control of the delivery and ensure that only students and approved university workers deliver food on campus for safety and health reasons.

Table Information for newly added tables:

1. **Table name:** ratings

   Purpose: To maintain the ratings for the respective orders.

   Table content:

```
SELECT * from ratings;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	order_id	overall_rating
1	63	1
2	39	2
3	34	1
4	52	1
5	72	2
6	84	1
7	89	5
8	84	5
9	18	2
10	24	2
11	12	4
12	26	4
13	81	1
14	20	1
15	57	3
16	18	5

2. **Table name:** driver_rating

   Purpose: To maintain the delivery driver's rating

   Table content:

```
34 •  SELECT * from driver_rating;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	order_id	driver_id	deliverydriver_rating
1	44	4	1
2	64	8	4
3	83	7	4
4	95	5	5
5	34	2	3
6	64	6	4
7	60	1	1
8	12	3	5
9	13	5	2
10	67	3	4
11	60	3	5
12	27	4	3
13	52	3	5
14	79	7	4
15	85	1	1
16	85	2	5

3. **Table name:** food_rating

Purpose: To maintain the overall food ratings for the respective orders

Table content:

```
35 •  SELECT * from food_rating;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

foodrating_id	order_id	restaurant_id	overallfood_rating
1	93	26	2
2	2	50	3
3	77	36	1
4	7	38	1
5	31	79	2
6	93	17	2
7	21	37	4
8	48	90	4
9	28	1	3
10	28	56	1
11	48	100	3
12	8	40	4
13	1	54	1
14	61	73	5
15	84	8	1
16	28	96	5

View:

1. **FoodOrderAssociatedPerson:**

Purpose: For an individual person, get the order details, which include location name, restaurant name, delivery charges and overall rating.

Query screenshot:

```
drop view FoodOrderAssociatedPerson;

create view FoodOrderAssociatedPerson as
select o.order_id, p.person_name, l.location_name, re.restaurant_name, o.delivery_charge, ra.overall_rating
from person as p join campus_eats_fall2020.order as o on p.person_id = o.person_id
join location as l on l.location_id = o.location_id
join ratings as ra on ra.order_id = o.order_id
join restaurant as re on re.restaurant_id = o.restaurant_id;

select * from FoodOrderAssociatedPerson;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	person_name	location_name	restaurant_name	delivery_charge	overall_rating
85	Javier Dooley	Apt. 010	Fisher, Yundt and Wiegand	8.84	2
85	Javier Dooley	Apt. 010	Fisher, Yundt and Wiegand	8.84	5
85	Javier Dooley	Apt. 010	Fisher, Yundt and Wiegand	8.84	4
50	Dr. Winston Gottlieb DVM	Apt. 030	Jenkins-Greenholt	9.92	4
2	Ms. Antonette Keeling	Apt. 080	Kerluke-Herman	9.43	5
2	Ms. Antonette Keeling	Apt. 080	Kerluke-Herman	9.43	2
41	Alexandrine Donnelly	Apt. 130	Corkery, Kilback and Skiles	7.06	1
18	Eddie Osinski	Apt. 158	Hauck, Rodriguez and Cremin	5.69	2
18	Eddie Osinski	Apt. 158	Hauck, Rodriguez and Cremin	5.69	5
18	Eddie Osinski	Apt. 158	Hauck, Rodriguez and Cremin	5.69	5
12	Isadore Auer IV	Apt. 170	Abbott-Schmitt	7.4	4
22	Sigmund Torp	Apt. 190	Eichmann-Casper	1.39	4
22	Sigmund Torp	Apt. 190	Eichmann-Casper	1.39	3
87	Prof. Kamryn Armstrong	Apt. 198	Veum PLC	4.32	2
66	Mr. Shane Lebsack	Apt. 224	Stanton-Marvin	1.88	1
15	Mr. Dayne Grimes DVM	Apt. 286	Connelly, Wolf and Murazik	4.77	2

FoodOrderAssociatedPerson 2

2. **PeopleOrderRatings:**

Purpose: Get the detailed ratings given by a person

Query screenshot:

```
drop view PeopleOrderRatings;

create view PeopleOrderRatings as
select p.person_name, p.cell, o.total_price, ifnull(cr.delivery_rating, 'NIL') as delivery_rating, ifnull(fr.food_rating, 'NIL') as food_rating,ifnull(cr.overall_
from person as p join campus_eats_fall2020.order as o on p.person_id = o.person_id
left join (select r.order_id, round(avg(r.overall_rating),2) as overall_rating , round(avg(dr.deliverydriver_rating),2) as delivery_rating
 from ratings as r join driver_rating as dr on r.order_id = dr.order_id group by r.order_id)
 as cr on cr.order_id = o.order_id left join
 (select ra.order_id, round(avg(f.overallfood_rating),2) as food_rating
 from ratings as ra join food_rating as f on ra.order_id = f.order_id
 group by ra.order_id) as fr
 on fr.order_id = cr.order_id
 order by overall_rating;

 select * from PeopleOrderRatings;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_name	cell	total_price	delivery_rating	food_rating	overall_rating
Baron Feil	9863696126	3.21	1.00	1.00	1.00
Samson Hansen	9139417942	15.51	2.75	5.00	1.00
Alexandrine Donnelly	9403417764	8.85	2.00	1.00	1.00
Mr. Terrell Becker	9519262117	16.72	4.00	2.00	1.00
Mrs. Freida Murphy	9837045438	7.94	4.50	1.00	1.67
Mr. Gabe Bauch PhD	9416224204	8.6	2.00	NIL	2.00
Randi Witting IV	9418786674	12.49	5.00	NIL	2.00
Candace Koss	9308906690	4.73	2.00	1.00	2.00
Juvenal McLaughlin	9246052737	3.27	3.00	NIL	2.00
Elnora Weissnat	9348642594	13.39	5.00	NIL	2.00
Lilliana Emard	9945887810	14.92	4.00	1.50	2.00
Mr. Baron Windler I	9242394545	7.39	4.67	4.00	2.00
Mrs. Kaitlyn Jacobs Sr.	9259287865	11.61	2.00	2.50	2.33

3. **AverageDriversDeliveryRating:**

Purpose: To maintain the average rating for derivers

Qyery Screenshot:

```sql
drop view AverageDriversDeliveryRating;
create view AverageDriversDeliveryRating as
select d.driver_id, d.license_number, sp.person_name, odr.avg_rating from driver as d left join
(select s.student_id, p.person_name from person as p
join student as s on p.person_id = s.person_id) as sp
on d.student_id = sp.student_id left join
(select o.driver_id, o.order_id, round(avg(dr.deliverydriver_rating),2) as avg_rating
from campus_eats_fall2020.order as o join
driver_rating as dr on dr.driver_id = o.driver_id
group by o.driver_id) as odr on d.driver_id = odr.driver_id
order by d.driver_id ;

select * from AverageDriversDeliveryRating;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

driver_id	license_number	person_name	avg_rating
1	fi3980	Leopoldo Welch	2.55
2	js4003	Mr. Sigrid Morissette	3.56
3	xt8429	Leann O'Kon Sr.	3.83
4	yv6392	Stone Kshlerin	2.92
5	cf3679	Ms. Sincere McDermott	3.50
6	rx3942	Fabiola Gusikowski V	2.89
7	mf2647	Caitlyn Runolfsdottir	3.82
8	ia9372	Royce Ledner II	3.25

Stored Procedure:

1. **Get driver's rating:**

Purpose: To get each driver's rating

```
-- To get a list of drivers' details who has the given rating
DELIMITER //
CREATE PROCEDURE `get_drivers_with_rating`(in rating int(1))
BEGIN
SELECT person_name as student_name, rating, cell as student_contact, graduation_year, major, s.type as college_type
FROM campus_eats_fall2020.driver AS d
INNER JOIN person AS p1
    INNER JOIN student AS s
    ON s.student_id = d.student_id
    AND s.person_id = p1.person_id
    AND d.rating = rating
END
DELIMITER ;

CALL get_drivers_with_rating(3);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_name	rating	student_contact	graduation_year	major	college_type
Mr. Sigrid Morissette	3	9699469427	2015	Philosophy	Undergraduate
Leann O'Kon Sr.	3	9521975342	1985	Cyber Security	Graduate
Stone Kshlerin	3	9851384624	1975	Philosophy	Graduate
Fabiola Gusikowski V	3	9121545851	1997	Data Science	Undergraduate
Caitlyn Runolfsdottir	3	9157717821	1996	Environmental	Undergraduate
Royce Ledner II	3	9511542747	1973	Data Science	Graduate

2. **Get order details:**

Purpose: Get order details which includes delivery person name, person who ordered the order, location name and location address.

```
DELIMITER //
-- Get order details like driver name, name of the person who ordered, delivery location and address for the given id
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_order_details`(in order_id varchar(100))
BEGIN
SELECT order_id, p1.person_name as driver, p.person_name as ordered_by, location_name, location_address
FROM campus_eats_fall2020.order AS o
INNER JOIN
   person AS p1
   ON o.driver_id = p1.person_id
   AND o.order_id = order_id
   INNER JOIN
   person AS p
   ON o.person_id = p.person_id
   AND o.order_id = order_id
   INNER JOIN
   location AS l
   ON o.location_id = l.location_id
   AND o.order_id = order_id
END
DELIMITER ;
CALL get_order_details(101);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	driver	ordered_by	location_name	location_address
101	Noel Emard	Keith Turner	Suite 057	28742 Cole Forest Suite 48...

Functions

1. **Get the years served for a specific driver id:**

Purpose: Get the years of service for the delivery person with driver_id

```
DELIMITER //
CREATE DEFINER=`root`@`localhost` FUNCTION `get_years_of_service`(id INTEGER) RETURNS int
    DETERMINISTIC
BEGIN
    DECLARE years_since_hired INT;
    SELECT TIMESTAMPDIFF(YEAR, d.date_hired, CURDATE()) INTO years_since_hired from driver as d WHERE d.driver_id=id;
    RETURN years_since_hired;
END
DELIMITER ;

SELECT get_years_of_service(2);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	get_years_of_service(2)
▶	4

Indexes:

1. **Index faculty_index for faculty table on faculty_id column:**

```
96 •  CREATE UNIQUE INDEX faculty_index ON faculty (faculty_id);
97 •  SHOW INDEX FROM campus_eats_fall2020.faculty;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
faculty	0	PRIMARY	1	faculty_id	A	26	NULL	NULL		BTREE			YES	NULL
faculty	0	faculty_index	1	faculty_id	A	26	NULL	NULL		BTREE			YES	NULL
faculty	1	fk_F_person_id	1	person_id	A	26	NULL	NULL		BTREE			YES	NULL