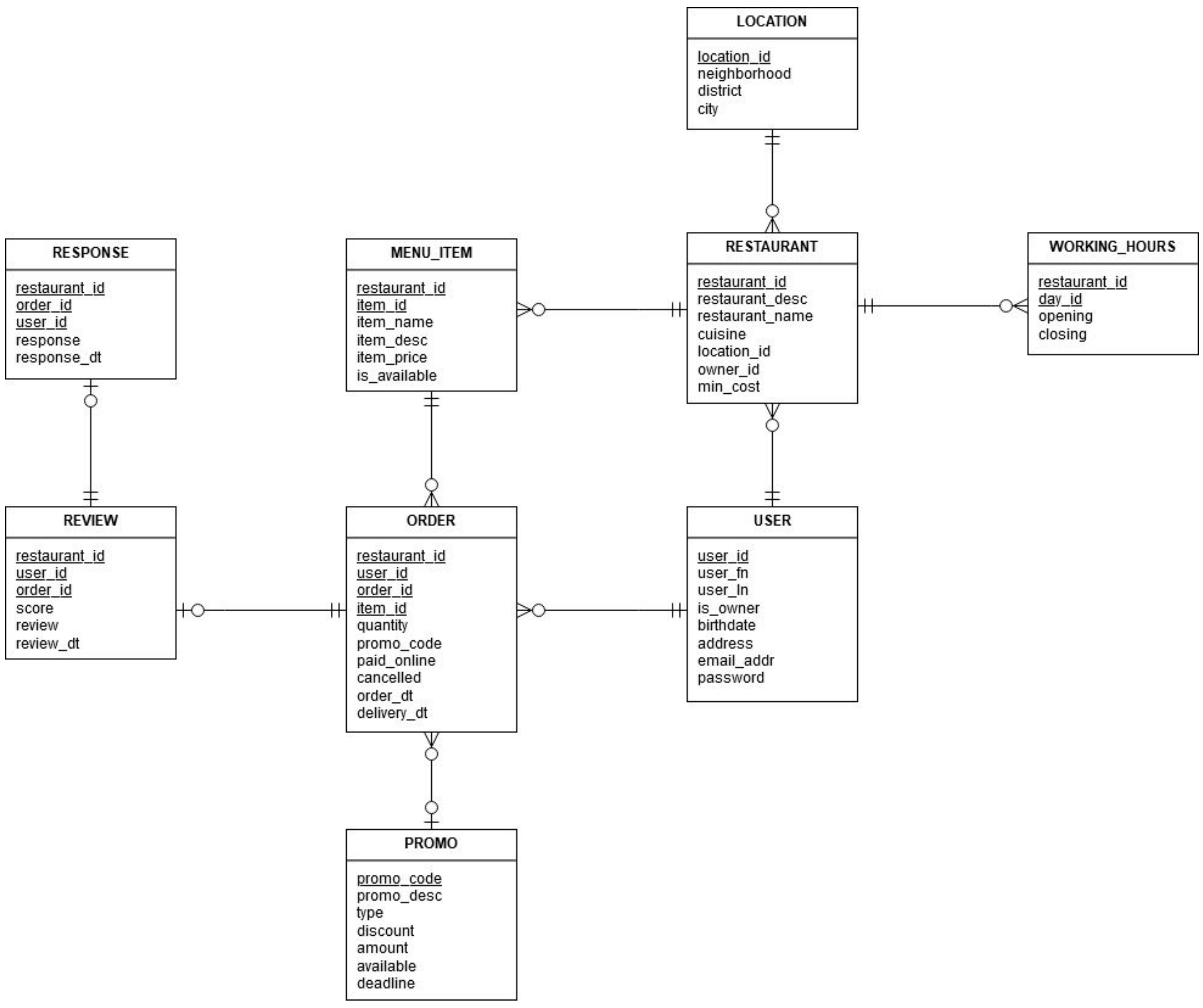
IS 503
Fall 2019
Assignment #2

Deadline: December 1, 2019 23:59

Please remember the previous assignment and check the corresponding relational schema below (which is a sample solution except some missing attributes). **Your solutions must comply with the given schema.** You can find the schema files (that you can import and use for verification) here. The data is obviously mostly fabricated, so do not use it for anything else. Download and extract this folder, open MySQL Workbench, "Data Import/Restore" from the navigation, find the file, and start the import process. If you do not know how to use MySQL Workbench, you can use this guide.



A) Write the following queries in relational algebra.

1. Find reviews that have a score greater than 7. Retrieve their restaurant IDs and review texts. (6 points)

   $\text{RESULT} \leftarrow \pi_{\text{restaurant\_id, review}} (\sigma_{\text{score} > 7} (\text{REVIEW}))$

2. Find the user who has the maximum amount of orders. Retrieve their ID, first name, and last name. (7 points)

   $\text{DISTINCT\_ORDERS} \leftarrow \pi_{\text{restaurant\_id, user\_id, order\_id}} (\text{ORDER})$

   $\text{ORDER\_COUNTS(user\_id, order\_count)} \leftarrow {}_{\text{user\_id}}\mathcal{F}_{\text{COUNT restaurant\_id, order\_id}}$ (DISTINCT_ORDER)

   $\text{MAX\_ORDER} \leftarrow \mathcal{F}_{\text{MAX order\_count}} (\text{ORDER\_COUNTS})$

   $\text{MAX\_ORDER\_USER} \leftarrow \pi_{\text{user\_id}} (\sigma_{\text{order\_count = MAX\_ORDER}} (\text{ORDER\_COUNTS}))$

   $\text{RESULT} \leftarrow \pi_{\text{user\_id, user\_fn, user\_ln}} (\text{MAX\_ORDER\_USER} \bowtie_{\text{MAX\_ORDER\_USER.user\_id = USER.user\_id}}$ $\text{USER})$

3. Find the restaurants that received the highest and the lowest number of reviews. Combine these two restaurants together and retrieve their names and IDs. (10 points)

   $\text{REVIEW\_COUNTS(restaurant\_id, review\_count)} \leftarrow {}_{\text{restaurant\_id}}\mathcal{F}_{\text{COUNT *}} (\text{REVIEW})$

   $\text{REVIEW\_MAX} \leftarrow \mathcal{F}_{\text{MAX review\_count}} (\text{REVIEW\_COUNTS})$

   $\text{REVIEW\_MIN} \leftarrow \mathcal{F}_{\text{MIN review\_count}} (\text{REVIEW\_COUNTS})$

   $\text{MAX\_RESTAURANT} \leftarrow \pi_{\text{restaurant\_id}} (\sigma_{\text{review\_count = REVIEW\_MAX}}\ \text{REVIEW\_COUNTS})$

   $\text{MIN\_RESTAURANT} \leftarrow \pi_{\text{restaurant\_id}} (\sigma_{\text{review\_count = REVIEW\_MIN}}\ \text{REVIEW\_COUNTS})$

   $\text{RESTAURANTS} \leftarrow \text{MAX\_RESTAURANT} \cup \text{MIN\_RESTAURANT}$

   $\text{RESULT} \leftarrow \pi_{\text{restaurant\_id, restaurant\_name}} (\text{RESTAURANTS} \bowtie_{\text{RESTAURANTS.restaurant\_id = RESTAURANT.restaurant\_id}} \text{RESTAURANT})$

4. Find the users who made at least five orders yet reviewed none of them. Retrieve their ID, first name, and last name. (10 points)

NO_REVIEW $\leftarrow \pi_{\text{user\_id}}$ (USER) - $\pi_{\text{user\_id}}$ (REVIEW)

DISTINCT_ORDERS $\leftarrow \pi_{\text{restaurant\_id, user\_id, order\_id}}$ (ORDER)

ORDERS(user_id, order_count) $\leftarrow {}_{\text{user\_id}}\mathcal{F}_{\text{COUNT restaurant\_id, order\_id}}$ (DISTINCT_ORDERS)

FIVE_ORDERS $\leftarrow \pi_{\text{user\_id}}$ ($\sigma_{\text{order\_count > 4}}$ (ORDERS))

PEOPLE $\leftarrow$ (NO_REVIEW $\cap$ FIVE_ORDERS)

RESULT $\leftarrow \pi_{\text{user\_id, user\_fn, user\_ln}}$ (PEOPLE $\bowtie_{\text{PEOPLE.user\_id = USER.user\_id}}$ USER)

5. Find the restaurant type that received the maximum amount of reviews. Retrieve restaurants IDs, number of customers (people who have made an order from that restaurant), and the total number of orders for each restaurant that belongs to that restaurant type you found. (12 points)

REVIEW_COUNTS(cuisine, review_count) $\leftarrow {}_{\text{cuisine}}\mathcal{F}_{\text{COUNT *}}$ (RESTAURANT $\bowtie_{\text{RESTAURANT.restaurant\_id = REVIEW.restaurant\_id}}$ REVIEW)

REVIEW_MAX $\leftarrow \mathcal{F}_{\text{MAX review\_count}}$ (REVIEW_COUNTS)

MAX_CUISINE $\leftarrow \pi_{\text{cuisine}}$ ($\sigma_{\text{review\_count = REVIEW\_MAX}}$ (REVIEW_COUNTS))

CUISINE_RESTAURANTS $\leftarrow \pi_{\text{restaurant\_id}}$ ($\sigma_{\text{cuisine = MAX\_CUISINE}}$ (RESTAURANT)

CUSTOMER_COUNTS(restaurant_id, customers) $\leftarrow {}_{\text{restaurant\_id}}\mathcal{F}_{\text{COUNT user\_id}}$ ($\pi_{\text{restaurant\_id, user\_id}}$ (ORDER))

ORDER_COUNTS(restaurant_id, orders) $\leftarrow {}_{\text{restaurant\_id}}\mathcal{F}_{\text{COUNT user\_id}}$ ($\pi_{\text{restaurant\_id, user\_id, order\_id}}$ (ORDER))

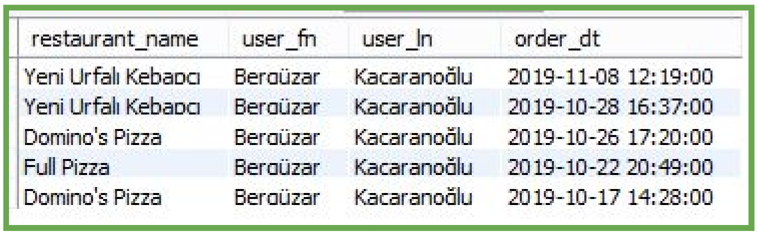
RESULT $\leftarrow$ CUISINE_RESTAURANTS $\bowtie$ CUSTOMER_COUNTS $\bowtie$ ORDER_COUNTS

B) Write the following queries in <u>SQL</u> and include your queries <u>in a plain text format that can be easily copied and pasted</u>. Please use MySQL to make sure your queries actually work.

1. We want to display the last five orders for each customer on the main page to encourage them review their past orders. For example, retrieve the name of the user <u>whose user_id is 5</u> and the last five restaurants' names from which they made an order along with their order dates. (7 points)

   ```
   SELECT r.restaurant_name, u.user_fn, u.user_ln, o.order_dt FROM is_ys.order o
   INNER JOIN is_ys.user u USING (user_id) INNER JOIN restaurant r USING
   (restaurant_id) WHERE u.user_id = 5 GROUP BY user_id, restaurant_id, order_id
   ORDER BY o.order_dt DESC LIMIT 5;
   ```

   Result:

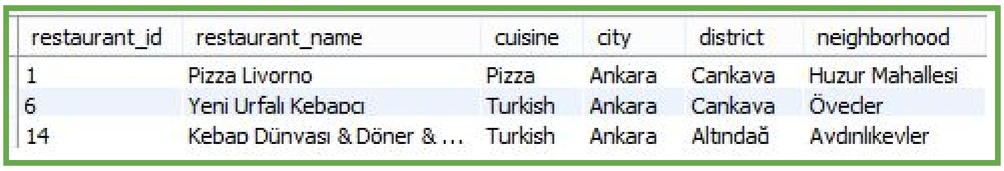
   

restaurant_name	user_fn	user_ln	order_dt
Yeni Urfalı Kebapcı	Bergüzar	Kacaranoğlu	2019-11-08 12:19:00
Yeni Urfalı Kebapcı	Bergüzar	Kacaranoğlu	2019-10-28 16:37:00
Domino's Pizza	Bergüzar	Kacaranoğlu	2019-10-26 17:20:00
Full Pizza	Bergüzar	Kacaranoğlu	2019-10-22 20:49:00
Domino's Pizza	Bergüzar	Kacaranoğlu	2019-10-17 14:28:00

2. We want to analyze restaurants and their performances. Find the restaurants that are closed on at least one day and could not receive more than 10 orders with a significant price (at least 50 TL or more) ordered in the last 30 days (take 10/30/2019 as the reference day). Retrieve these restaurants' IDs, names, cuisines, and location details. Ignore promotions but not cancellations. Take the order quantities into consideration. (12 points) (Tip: A restaurant that is closed for a specific day would not have working hours record for that day.)

   ```
   SELECT t1.restaurant_id, t1.restaurant_name, t1.cuisine, t1.city, t1.district,
   t1.neighborhood FROM (SELECT r.restaurant_id, r.restaurant_name, r.cuisine, l.city,
   l.district, l.neighborhood FROM is_ys.restaurant r INNER JOIN is_ys.location l
   USING (location_id) WHERE EXISTS(SELECT restaurant_id FROM
   is_ys.working_hours w WHERE w.restaurant_id = r.restaurant_id GROUP BY
   w.restaurant_id HAVING COUNT(day_id) < 7)) t1 INNER JOIN
   (SELECT o.restaurant_id FROM is_ys.order o INNER JOIN menu_item m USING
   (restaurant_id, item_id) WHERE o.cancelled = 0 AND o.quantity*m.item_price >= 50
   AND o.order_dt BETWEEN ("2019-10-30" - INTERVAL 30 DAY) AND "2019-10-30"
   GROUP BY o.restaurant_id, o.user_id, o.order_id) t2 USING (restaurant_id) GROUP
   BY restaurant_id HAVING COUNT(*) <= 10;
   ```

   Result:

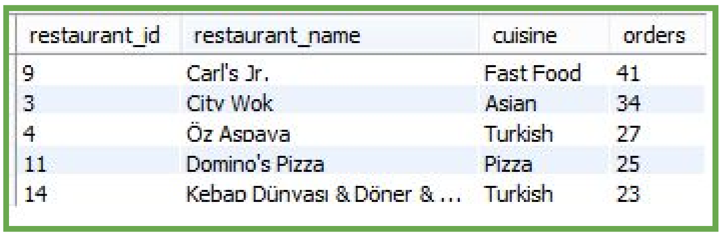


restaurant_id	restaurant_name	cuisine	city	district	neighborhood
1	Pizza Livorno	Pizza	Ankara	Cankava	Huzur Mahallesi
6	Yeni Urfalı Kebapcı	Turkish	Ankara	Cankava	Övecler
14	Kebap Dünvası & Döner & ...	Turkish	Ankara	Altındağ	Avdınlıkevler

3. Find the top five restaurants (limit your results to five restaurants no matter what) that received the highest numbers of orders given that their number of responses is greater than half of their number of reviews. Retrieve their IDs, names, cuisines, and numbers of orders. Order the restaurants by their number of orders in a descending order. (12 points) (Tip: For example, if a restaurant has the highest number of orders while not responding to more than 50% of their reviews, that restaurant should not be retrieved. Therefore, you might want to filter those restaurants first.)

```
SELECT t2.restaurant_id, t2.restaurant_name, t2.cuisine, COUNT(DISTINCT
o.restaurant_id, o.user_id, o.order_id) AS orders FROM (SELECT r.restaurant_id,
r.restaurant_name, r.cuisine FROM (SELECT a.restaurant_id, a.reviews/b.responses
response_rate
FROM (SELECT restaurant_id, COUNT(*) reviews FROM is_ys.review GROUP BY
restaurant_id) a
INNER JOIN (SELECT restaurant_id, COUNT(*) responses FROM is_ys.response
GROUP BY restaurant_id) b USING (restaurant_id)
WHERE a.reviews/b.responses < 2) t1, is_ys.restaurant r WHERE t1.restaurant_id =
r.restaurant_id) t2
INNER JOIN is_ys.order o USING (restaurant_id) GROUP BY restaurant_id ORDER
BY orders DESC LIMIT 5;
```

Result:



restaurant_id	restaurant_name	cuisine	orders
9	Carl's Jr.	Fast Food	41
3	Citv Wok	Asian	34
4	Öz Aspava	Turkish	27
11	Domino's Pizza	Pizza	25
14	Kebap Dünvası & Döner & ...	Turkish	23

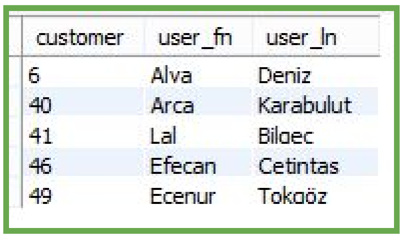
4. Find people who ordered from a restaurant whose owner had ordered from their restaurant as well, given that these two restaurants belong to the same cuisine. (12 points) In other words, find user pairs who ordered from each other's restaurants given that the restaurants share the cuisine. For example, if John Doe (user_id = 1) ordered from Restaurant X that is owned by Jane Roe, Jane Roe (user_id = 2) ordered from Restaurant Y that is owned by John Doe, and Restaurant X and Restaurant Y are of the same cuisine, you should retrieve their IDs, first names, and last names. Your query should bring a person only once. For this scenario, the result would look like this:

user_id	**user_fn**	**user_ln**

1	John	Doe
2	Jane	Roe

SELECT DISTINCT d1.customer, u2.user_fn, u2.user_ln FROM (SELECT distinct o.user_id as customer FROM is_ys.order o INNER JOIN is_ys.user u USING (user_id) INNER JOIN is_ys.restaurant r USING (restaurant_id)
WHERE EXISTS (SELECT o.user_id, r.owner_id FROM is_ys.order o1 INNER JOIN is_ys.user u1 USING (user_id) INNER JOIN is_ys.restaurant r1 USING (restaurant_id) WHERE o.user_id=r1.owner_id AND r.owner_id=o1.user_id AND r.cuisine = r1.cuisine)) d1 INNER JOIN is_ys.user u2 ON (d1.customer = u2.user_id);

Result:

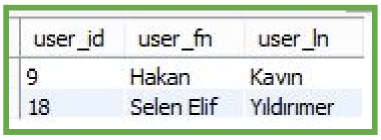


customer	user_fn	user_ln
6	Alva	Deniz
40	Arca	Karabulut
41	Lal	Bilgec
46	Efecan	Cetintas
49	Ecenur	Tokgöz

5. Find the users who reviewed all the Asian restaurants yet never reviewed a Turkish restaurant (by looking at the cuisine column). Retrieve their IDs, first names, and last names. (12 points)

SELECT * FROM (SELECT u.user_id, u.user_fn, u.user_ln FROM is_ys.user u, is_ys.restaurant r, is_ys.review re WHERE u.user_id = re.user_id AND re.restaurant_id = r.restaurant_id AND r.cuisine = "Asian" GROUP BY u.user_id HAVING COUNT(DISTINCT re.restaurant_id) = (SELECT COUNT(restaurant_id) FROM is_ys.restaurant WHERE cuisine = "Asian")) t1
WHERE NOT EXISTS (SELECT r.restaurant_id from is_ys.restaurant r INNER JOIN is_ys.review re using (restaurant_id) WHERE r.cuisine = "Turkish" and t1.user_id = re.user_id);

Result:



user_id	user_fn	user_ln
9	Hakan	Kavın
18	Selen Elif	Yıldırımer