Today’s agenda

- Suggested answers of SQL Challenge
- More SQL examples
- Advanced SQL examples

SQL Challenge

- In this challenge, you need to write ten SQL queries to complete ten questions. Submit your answers in a PDF file.
- A database of train stations with five tables, including **Customer**, **Card**, **Station**, **Usage**, and **Owner**, are given in the next slide.
  - The **Owner** table stores customers who have cards to pay tickets.
  - The **Usage** table stores customers' visits to the train station, the latest expense, and latest date.
  - The **Card** table stores the card balance.

COMP3278B

Introduction to Database Management Systems

Dr. Ping Luo
Email : pluo@cs.hku.hk

Department of Computer Science, The University of Hong Kong

customer_id	name
1	Alan
2	Bob
3	Cindy
4	David

Customer

card_id	balance
1	-10
2	-4
3	30
4	-2
5	-1
6	10

Card

station_id	name
1	Wan Chai
2	Central
3	Admiralty

Station

customer_id	card_id	station_id	lastest_expense	lastest_date
1	1	2	12	2021-02-25
1	2	2	8	2021-02-25
2	3	3	4	2021-02-26
2	4	3	6	2021-02-24
3	6	3	4	2021-02-24
4	5	2	12	2021-02-24
4	5	1	10	2021-02-24
4	5	1	12	2021-02-24

Usage

customer_id	card_id
1	1
1	2
2	3
2	4
3	6
4	5

Owner

Question 1

- Write a SQL query to find the customer who has the maximum number of cards.

Requirements:

1. Return customer_id
2. Evaluate your answer in MySQL

customer_id	card_id
1	1
1	2
2	3
2	4
3	6
4	5

Owner

```
SELECT O.customer_id
FROM Owner O
GROUP BY O.customer_id
HAVING COUNT(O.card_id) >= ALL(
            SELECT COUNT(card_id)
            FROM Owner
            GROUP BY customer_id);
```

customer_id
1
2

Question 2

- Write a SQL query to find the card that has the maximum balance.

Requirements:

1. Return card_id, card balance
2. Evaluate your answer in MySQL

card_id	balance
1	-10
2	-4
3	30
4	-2
5	-1
6	10

Card

```
SELECT C.card_id, MAX(C.balance)
FROM Card AS C
GROUP BY C.card_id
HAVING MAX(C.balance) >= ALL(
            SELECT MAX(balance)
            FROM Card
            GROUP BY card_id);
```

card_id	max_balance
3	30

Question 3

Common error for GROUP BY: any column 'col' is invalid in the select list, because it is not contained in either an aggregate function (e.g. MAX) or the GROUP BY clause. [MySQL v5.7.5 or higher.]

- Write a SQL query to find each customer's card, which has the largest balance. (Hint: a customer may have multiple cards)

Requirements:

1. Return customer_id, card_id, card balance
2. Evaluate your answer in MySQL

```
SELECT O1.customer_id, O1.card_id, C.balance
FROM Card C, Owner O1
WHERE C.card_id = O1.card_id
AND C.balance IN (
        SELECT MAX(C.balance)
        FROM Card C, Owner O2
        WHERE C.card_id = O2.card_id
        GROUP BY O2.customer_id
        HAVING O1.customer_id = O2.customer_id);
```

customer_id	card_id	balance
1	2	-4
2	3	30
4	5	-1
3	6	10

Question 4

- Write a SQL query to find the third highest card balance. (Hint: use the LIMIT clause)

Requirements:

1. Return card balance
2. You are required to use the SELECT…FROM syntax once only
3. Evaluate your answer in MySQL

```
SELECT DISTINCT balance
FROM Card
ORDER BY balance DESC LIMIT 2,1;
```

balance
-1

LIMIT x, y means the offset is x and then return y number of records.

Question 5

customer_id	station_name	latest_date	count
1	Central	2021/2/25	2
4	Wan Chai	2021/2/24	2

- Write a SQL query to find the list of customers, who took train at the same station more than once in the same day, and date must be ordered from the most recent date to oldest date. (Hint: Use the GROUP BY syntax)

Requirements:

1. Return customer_id, station_name, latest_date, number of visits
2. Evaluate your answer in MySQL

```
SELECT U.customer_id, S.station_name, U.latest_date, COUNT(U.station_id) AS count
FROM Usage AS U, Station AS S
WHERE U.station_id = S.station_id
GROUP BY U.customer_id, U.station_id, S.station_name, U.latest_date
HAVING COUNT(U.station_id) > 1
ORDER BY U.latest_date DESC;
```

Question 6

- Suppose the customer_id is a unique numeric increasing value, write a SQL query to select the first two even customer_id values.

Requirements:

1. You are required to use the SELECT…FROM syntax once only
2. Evaluate your answer in MySQL

```
SELECT TOP 2 customer_id
FROM Customer
WHERE customer_id % 2 = 0
ORDER BY customer_id
```

customer_id
2
4

Question 7

- What are the outputs of the below SQL queries?

    1. SELECT COUNT(*), SUM(1) FROM Usage GROUP BY customer_id
    2. SELECT COUNT(*), SUM(2) FROM Usage GROUP BY station_id
    3. SELECT COUNT(*), SUM(3) FROM Usage GROUP BY latest_expense

Requirements:

1. Explain why.

COUNT(*)	SUM(1)
2	2
2	2
1	1
3	3

Question 8

- Write a SQL query to calculate the sum of all positive card balances, and the sum of all negative card balances.

Requirements:

1. You are allowed to use the SELECT…FROM syntax once only
2. Return the sum of all positive balances and the sum of all negative balances

```
SELECT SUM(CASE WHEN balance>0 THEN balance ELSE 0 END) AS sum_pos,
SUM(case WHEN balance<0 THEN balance ELSE 0 END) AS sum_neg
FROM Card;
```

sum_pos	sum_neg
40	-17

```
SELECT SUM(balance)
FROM Card
GROUP BY SIGN(balance)
```

Question 9

- Write a SQL query to add 20 to negative card balance, add 5 to positive card balance. (Hint: use the SELECT clause, not the UPDATE clause)

Requirements:

1. Evaluate your answer in MySQL

```
UPDATE Card C SET C.balance = CASE
WHEN C.balance < 0
THEN C.balance + 20
ELSE C.balance + 5
END;

SELECT (CASE WHEN C.balance < 0
THEN C.balance + 20
ELSE C.balance + 5 END) AS balance
FROM Card C;
```

balance
10
16
35
18
19
15

Question 10

- Write a SQL query to find the customer, who has the maximum number of cards, and all his/her cards have been deducted to negative balance at the same station within the last ten days. (Hint: (1) partition the query to different subqueries; (2) write SQL for each subquery; (3) write main query by using the SELECT… FROM… WHERE syntax)

Requirements:

1. Return customer_id, customer_name, number of all his/her cards satisfied the above query, station_name
2. Evaluate your answer in MySQL

Answer Q10

- Subqueries
    1. Return a list of latest card usages
    2. Return maximum number of cards of the customers
    3. Return number of cards that have been deducted to negative balance in the last ten days
    4. Return number of cards that have been deducted to negative balance in the same station
    5. …

Answer Q10

Usage

customer_id	card_id	station_id	lastest_expense	lastest_date
1	1	2	12	2021-02-25
1	2	2	8	2021-02-25
2	3	3	4	2021-02-26
2	4	3	6	2021-02-24
3	6	3	4	2021-02-24
4	5	2	12	2021-02-24
4	5	1	10	2021-02-24
4	5	1	12	2021-02-24

Latest Usage

customer_id	card_id	station_id	lastest_expense	lastest_date
1	1	2	12	2021-02-25
1	2	2	8	2021-02-25
2	3	3	4	2021-02-26
2	4	3	6	2021-02-24
3	6	3	4	2021-02-24
4	5	2	12	2021-02-24

[Latest_Usage]:

```
SELECT U1.customer_id, U1.card_id, U1.station_id, U1.latest_expense, U2.max_date
FROM Usage AS U1
INNER JOIN
        (SELECT card_id, MAX(latest_date) AS max_date
        FROM Usage
        GROUP BY card_id) AS U2
ON U2.card_id = U1.card_id
AND U2.max_date = U1.latest_date
```

Answer Q10

- Main query

```
SELECT DISTINCT U.customer_id, CM.customer_name, [Num_of_Latest_Neg_Card].num_card, S.station_name
FROM Usage AS U, Station AS S, Customer AS CM, [Num_of_Card], [Num_of_Latest_Neg_Card],
        [Num_of_Latest_Card_at_Station]
WHERE U.customer_id = [Num_of_Card].customer_id
        AND  U.customer_id = [Num_of_Latest_Neg_Card].customer_id
        AND CM.customer_id = U.customer_id
        AND  U.station_id = S.station_id
        AND  U.customer_id= [Num_of_Latest_Card_at_Station].customer_id
        AND  [Num_of_Card].num_card = [Num_of_Latest_Neg_Card].num_card
        AND [Num_of_Card].num_card= [Num_of_Latest_Card_at_Station].num_card
        AND [U.customer_id has the maximum number of cards];
```

Answer Q10

[Num_of_Card]:

```
(SELECT O.customer_id, COUNT(O.card_id) AS num_card
FROM Owner AS O
GROUP BY O.customer_id)
AS Num_of_Card
```

[Num_of_Latest_Neg_Card]:

```
(SELECT U.customer_id, COUNT(U.card_id) AS num_card
FROM [Latest_Usage] AS U, Card AS C
WHERE U.latest_date >= DATE_SUB(CURDATE(), INTERVAL 10 DAY)
AND C.balance < 0
AND U.card_id = C.card_id
GROUP BY U.customer_id)
AS Num_of_Latest_Neg_Card
```

Answer Q10

customer_id	name	num_card	station_name
1	Alan	2	Central

[Num_of_Latest_Card_at_Station]:

```
(SELECT U.customer_id, COUNT(U.card_id) AS num_card
FROM [Latest_Usage] AS U
WHERE U.latest_date >= DATE_SUB(CURDATE(), INTERVAL 10 DAY)
GROUP BY U.customer_id, U.station_id)
AS Num_of_Latest_Card_at_Station
```

[Latest_Usage]:

```
SELECT U1.customer_id, U1.card_id, U1.station_id, U1.latest_expense, U2.max_date
FROM Usage AS U1
INNER JOIN
        (SELECT card_id, MAX(latest_date) AS max_date
        FROM Usage
        GROUP BY card_id) AS U2
ON U2.card_id = U1.card_id
AND U2.max_date = U1.latest_date
```

More SQL Examples

Dataset

You can download **`L5_tables.sql`** and **`L5_data.sql`** on Moodle and import them to MySQL for testing.

- **Restaurant** (*restaurant_id*, *name*))
- **RestaurantCategory** (*restaurant_id*, *category*)
  - Foreign key: {*restaurant_id*} references **Restaurant**
- **Branch** (*restaurant_id*, *branch_no*, *location*, *seats*)
  - Foreign key: {*restaurant_id*} references **Restaurant**
- **Member** (*member_id*, *name*, *birthday, joined, points*)
  - (*joined* is the year the member joined)
- **Visits** (*visit_id*, *member_id*, *restaurant_id*, *branch_no*, *date*, *score*)
  - Foreign keys: {*restaurant_id, branch_no*} references **Branch**,
    {*member_id*} references **Member**

Example 1 – Basic query

- List all branches showing the restaurant name and branch location, order by the number of seats in the branches.

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

Example 1 – join tables

First, we find and join all necessary tables that gives all information needed for the query.

- Where can I find all the information needed?
  - Restaurant name – **Restaurant**
  - Branch location – **Branch**

This suggests:
`SELECT … FROM Restaurant, Branch` or
`SELECT … Restaurant inner/outer join Branch`

- What is the relationship between the two tables?
  - They are connected by the key *restaurant_id*

This gives a condition:
`Restaurant.restaurant_id = Branch.restaurant_id`

```
SELECT …
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id
      = Branch.restaurant_id
```

OR

```
SELECT …
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id
   = Branch.restaurant_id
```

Example 1- Select/Order

Then we apply the constraints needed for the query.

SELECT… showing the restaurant name and branch location

```
SELECT Restaurant.name, Branch.location …
```

ORDER BY… order by seat numbers

```
… ORDER BY Branch.seats
```

```
SELECT Restaurant.name,
       Branch.location
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id
      = Branch.restaurant_id
ORDER BY Branch.seats
```

OR

```
SELECT Restaurant.name,
       Branch.location
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id
   = Branch.restaurant_id
ORDER BY Branch.seats
```

Example 1 – answer

```
SELECT Restaurant.name,
       Branch.location
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id
      = Branch.restaurant_id
ORDER BY Branch.seats
```

OR

```
SELECT Restaurant.name,
       Branch.location
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id
   = Branch.restaurant_id
ORDER BY Branch.seats
```

name	location
PizzaHub	Causeway Bay
McRonalds	Admiralty
McRonalds	Central
DeliItaly	Admiralty
DeliItaly	Causeway Bay
Starducks	Wan Chai
DeliItaly	Wan Chai
Starducks	Central
UltraSandwich	Admiralty
UltraSandwich	North Point
UltraSandwich	Central

Exercise 1

- List all visiting records showing the member's name and visiting date, order by the score.

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Exercise 1

- List all visiting records showing the member’s name and visiting date, order by the score.

Please draft the corresponding SQL before you continue

Exercise 1 - answer

```
SELECT
    Member.name,
    visits.date
FROM
    visits,
    Member
WHERE
    visits.member_id = Member.member_id
ORDER BY
    score
```

name	date
Lonny	2018-05-10
Mavis	2018-06-07
Todd	2018-06-21
Keith	2018-12-04
Royce	2018-03-06
Todd	2018-01-19
Cleo	2018-12-09
Evan	2018-03-26
Lonny	2018-02-12
Alvin	2016-01-15
Royce	2018-07-27
Evan	2018-06-14
Cleo	2018-06-16
Keith	2018-01-27
Mavis	2018-06-11
Evan	2018-11-25
Cleo	2015-12-30
Keith	2018-05-08
Todd	2018-01-16
Lonny	2018-08-02
Keith	2018-04-22

Example 2 – Inner join and group by

- Find the name of restaurant branches in Admiralty and the corresponding number of visits. List only those with at least 2 visits.

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Example 2 – Join tables

- We need the following information
  - Restaurant name – **Restaurant**
  - Branch location – **Branch**
  - Number of visits – **visits**
- How should we join **Branch** and **visits**?
  - We must join the table by the corresponding key.

```
SELECT …
FROM Restaurant, Branch, visits
WHERE Restaurant.restaurant_id = Branch.restaurant_id
      AND Branch.restaurant_id = visits.restaurant_id
      AND Branch.branch_no = visits.branch_no
```

{restaurant_id, branch_no} is the primary key of **Branch** table.
We must match **both** columns when joining such table.

Example 2 – using inner join

- Alternatively we can use INNER JOIN

```
SELECT …
FROM Restaurant, Branch, visits
WHERE Restaurant.restaurant_id = Branch.restaurant_id
      AND Branch.restaurant_id = visits.restaurant_id
      AND Branch.branch_no = visits.branch_no
```

OR

```
SELECT …
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id = Branch.restaurant_id
INNER JOIN visits
ON Branch.restaurant_id = visits.restaurant_id
   AND Branch.branch_no = visits.branch_no
```

Example 2 – SELECT…

- We need to extract two information from the joined table
  - Restaurant.name from Restaurant table
  - Number of visits from visits table – we need to use an aggregate function

```
SELECT Restaurant.name, COUNT(*)
FROM …
```

Without a GROUP BY clause, this will count all result as one single group

- Aggregate function `COUNT()` in our query is used to count the number of visits **per branch**, therefore we must add a GROUP BY clause to group data by branches.

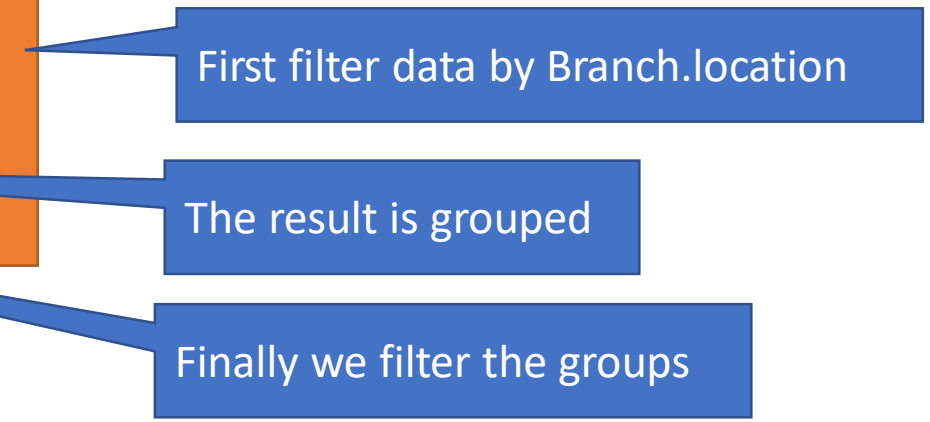
```
SELECT Restaurant.name, COUNT(*)
FROM …
GROUP BY Branch.restaurant_id, Branch.branch_no
```

Again, we need BOTH columns here.
When we refer to a table, we always use the whole set of primary key

Example 2 - filter

- Finally we filter the data. There are two filters:
  - Location is Admiralty – Filter on the raw data, we use the WHERE clause
  - Number of visits is at least 2 – Filter on the aggregated data, we need to use HAVING

```
SELECT …
WHERE Branch.location = 'Admiralty'
…
GROUP BY …
HAVING COUNT(*) >= 2
```



Example 2 - answer

```
SELECT Restaurant.name, COUNT(*) AS visit_count
FROM Restaurant, Branch, visits
WHERE Restaurant.restaurant_id = Branch.restaurant_id
      AND Branch.restaurant_id = visits.restaurant_id
      AND Branch.branch_no = visits.branch_no
      AND Branch.location = 'Admiralty'
GROUP BY Branch.restaurant_id, Branch.branch_no
HAVING visit_count >= 2
```

name	visit_count
UltraSandwich	3

OR

```
SELECT Restaurant.name, COUNT(*) AS visit_count
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id = Branch.restaurant_id
INNER JOIN visits
ON Branch.restaurant_id = visits.restaurant_id
   AND Branch.branch_no = visits.branch_no
WHERE Branch.location = 'Admiralty'
GROUP BY Branch.restaurant_id, Branch.branch_no
HAVING visit_count >= 2
```

Exercise 2

- Find the maximum score given by each member, considering only visits on or after year 2018. Show only the members who have given a maximum score of at least 3.
  - Hint: you can compare a date string, e.g., `` `date`>='2018-01-01' ``

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Exercise 2

- Find the maximum score given by each member, considering only visits on or after year 2018. Show only the members who have given a maximum score of at least 3.
  - Hint: you can compare a date string,
    e.g., `` `date`>='2018-01-01' ``

Please draft the corresponding SQL before you continue

Exercise 2 - answer

```
SELECT
    Member.name,
    MAX(score) AS max_score
FROM
    Member,
    visits
WHERE
    Member.member_id = visits.member_id
    AND visits.date >= '2018-01-01'
GROUP BY
    Member.member_id
HAVING
    max_score >= 3
```

name	max_score
Cleo	5
Evan	5
Todd	6
Lonny	7
Keith	8
Royce	4
Mavis	5

Example 3 – Outer Join

- Find the name of restaurant branches in Admiralty and the corresponding number of visits. **Show a count of 0 for branches with no visiting record.**

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Example 3 – outer join

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
3	1	Admiralty	25
4	2	Admiralty	100

Branch in Admiralty

- This question is very similar to example 2, but there is one requirement that cannot be achieved with inner join.
- Consider joining branch in Admiralty and visits.

restaurant_id	branch_no	location	seats	visit_id	member_id	restaurant_id	branch_no	date	score
3	1	Admiralty	25	9	5	3	1	2018-04-22	8
4	2	Admiralty	100	15	7	4	2	2018-06-07	1
4	2	Admiralty	100	16	5	4	2	2018-12-04	2
4	2	Admiralty	100	17	3	4	2	2018-01-19	3

Branches in Admiralty INNER JOIN visits

- Not all of the branches appears in the joined table! To get a zero count, we must use an outer join

restaurant_id	branch_no	location	seats	visit_id	member_id	restaurant_id	branch_no	date	score
1	1	Admiralty	10	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*
3	1	Admiralty	25	9	5	3	1	2018-04-22	8
4	2	Admiralty	100	15	7	4	2	2018-06-07	1
4	2	Admiralty	100	16	5	4	2	2018-12-04	2
4	2	Admiralty	100	17	3	4	2	2018-01-19	3

Branches in Admiralty LEFT OUTER JOIN visits

Example 3 - counting

restaurant_id	branch_no	location	seats	visit_id	member_id	restaurant_id	branch_no	date	score
1	1	Admiralty	10	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*
3	1	Admiralty	25	9	5	3	1	2018-04-22	8
4	2	Admiralty	100	15	7	4	2	2018-06-07	1
4	2	Admiralty	100	16	5	4	2	2018-12-04	2
4	2	Admiralty	100	17	3	4	2	2018-01-19	3

Branches in Admiralty LEFT OUTER JOIN visits

- To count the number of visits, we must count the columns in the visits table. Consider the following SQL query:

```
SELECT Branch.restaurant_id, Branch.branch_no,
COUNT(*), COUNT(visits.visit_id)
FROM Branch
LEFT OUTER JOIN visits
ON Branch.restaurant_id = visits.restaurant_id
   AND Branch.branch_no = visits.branch_no
WHERE Branch.location = 'Admiralty'
GROUP BY Branch.restaurant_id, Branch.branch_no
```

restaurant_id	branch_no	COUNT(*)	COUNT(visits.visit_id)
1	1	1	0
3	1	1	1
4	2	3	3

This count is incorrect as the first row is counted by “*”

Example 3 - answer

name	visit_count
McRonalds	0
DeliItaly	1
UltraSandwich	3

```
SELECT Restaurant.name, COUNT(visits.visit_id) AS visit_count
FROM Restaurant
INNER JOIN Branch
ON Restaurant.restaurant_id = Branch.restaurant_id
LEFT OUTER JOIN visits
ON Branch.restaurant_id = visits.restaurant_id
   AND Branch.branch_no = visits.branch_no
WHERE Branch.location = 'Admiralty'
GROUP BY Branch.restaurant_id, Branch.branch_no
```

Exercise 3

- List all name of the members who have joined after 2010 and find the number of visits for each of them.

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Exercise 3

- List all name of the members who have joined after 2010 and find the number of visits for each of them.

Please draft the corresponding SQL before you continue

Exercise 3 - answer

```
SELECT
    Member.name,
    COUNT(visits.visit_id) AS visit_count
FROM
    Member
LEFT OUTER JOIN
    visits
ON
    Member.member_id = visits.member_id
WHERE
    Member.joined > 2010
GROUP BY
    Member.member_id
```

name	visit_count
Cleo	3
Lonny	3
Ira	0
Dino	0
A. Bella	0

Example 4 – Nested query

- For any restaurant that is a cafe, or that serves light meals, find the restaurant name and total number of seats available.

restaurant_id	category
1	Fast Food
1	Take Away
2	Italian
3	Cafe
3	Italian
4	Light meal
5	Cafe
5	Light meal
5	Take Away

RestaurantCategory

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

Example 4 – identifying sub-problems

- The problem could be broken down into smaller problems:
  - Find all restaurants that is a cafe, or serve light meals

```
SELECT restaurant_id FROM RestaurantCategory
WHERE category = 'Cafe' OR category = 'Light meal'
```

  - Find the total number of seats of those restaurants.

```
SELECT Restaurant.name, SUM(Branch.seats) FROM
Restaurant, Branch
WHERE Restaurant.restaurant_id = Branch.restaurant_id
GROUP BY Restaurant.restaurant_id
```

The first sub-problem can be used as a filter condition of the second sub-problem

Example 4 – using IN

name	total_seats
DeliItaly	105
UltraSandwich	390
Starducks	120

```
SELECT restaurant_id FROM RestaurantCategory
WHERE category = 'Cafe' OR category = 'Light meal'
```

+

```
SELECT Restaurant.name, SUM(Branch.seats)
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id = Branch.restaurant_id
GROUP BY Restaurant.restaurant_id
```

=

```
SELECT R.name, SUM(B.seats) AS total_seats
FROM Restaurant R, Branch B
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id IN (
        SELECT restaurant_id FROM RestaurantCategory RC
        WHERE RC.category = 'Cafe' OR RC.category = 'Light meal'
      )
GROUP BY R.restaurant_id
```

Example 4 – why subquery

- It breaks the problem down into smaller ones, which is easier to manage
- Avoid duplicated immediate results
  - Can you identify the problem in the following query?

```
SELECT R.name, SUM(B.seats) AS total_seats
FROM Restaurant R, Branch B, RestaurantCategory RC
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id = RC.restaurant_id
      AND (RC.category = 'Cafe' OR RC.category = 'Light meal')
GROUP BY R.restaurant_id
```

Example 4 – problem

name	SUM(Branch.seats)
DeliItaly	105
UltraSandwich	390
Starducks	**240**

```
SELECT R.name, SUM(B.seats) AS total_seats
FROM Restaurant R, Branch B, RestaurantCategory RC
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id = RC.restaurant_id
      AND (RC.category = 'Cafe' OR RC.category = 'Light meal')
GROUP BY R.restaurant_id
```

- As there could be multiple matches in RestaurantCategory, some restaurants is double-counted!

restaurant_id	category
1	Fast Food
1	Take Away
2	Italian
3	Cafe
3	Italian
4	Light meal
5	Cafe
5	Light meal
5	Take Away

RestaurantCategory

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	name	category
3	DeliItaly	Cafe
4	UltraSandwich	Light meal
5	Starducks	Cafe
5	Starducks	Light meal

Restaurant INNER JOIN RestaurantCategory filtered by "Cafe" and "Light meal"

Example 4 – other choices (FROM clause)

```
SELECT DISTINCT restaurant_id FROM RestaurantCategory
WHERE category = 'Cafe' OR category = 'Light meal'
```

Use DISTINCT keyword to ensure that there is no duplicated records

+

```
SELECT Restaurant.name, SUM(Branch.seats)
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id = Branch.restaurant_id
GROUP BY Restaurant.restaurant_id
```

=

```
SELECT R.name, SUM(B.seats) AS total_seats
FROM Restaurant R, Branch B, (
        SELECT DISTINCT restaurant_id FROM RestaurantCategory RC
        WHERE RC.category = 'Cafe' OR RC.category = 'Light meal'
    ) R2
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id = R2.restaurant_id
GROUP BY R.restaurant_id
```

Must use alias for sub-query

Example 4 – Other choice (EXISTS)

```
SELECT restaurant_id FROM RestaurantCategory
WHERE category = 'Cafe' OR category = 'Light meal'
```

+

```
SELECT Restaurant.name, SUM(Branch.seats)
FROM Restaurant, Branch
WHERE Restaurant.restaurant_id = Branch.restaurant_id
GROUP BY Restaurant.restaurant_id
```

=

```
SELECT R.name, SUM(B.seats) AS total_seats
FROM Restaurant R, Branch B
WHERE R.restaurant_id = B.restaurant_id
      AND EXISTS (
        SELECT restaurant_id FROM RestaurantCategory RC
        WHERE (RC.category = 'Cafe' OR RC.category = 'Light meal')
              AND R.restaurant_id = RC.restaurant_id
      )
GROUP BY R.restaurant_id
```

EXISTS will check the subquery for EACH of the matches in the main query.

Sub-query must correlates to the main query so that every time the sub-query is checked, the corresponding record is checked.

Example 4 - answer

```
SELECT R.name, SUM(B.seats) AS total_seats FROM Restaurant R, Branch B
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id IN (
        SELECT restaurant_id FROM RestaurantCategory RC
        WHERE (RC.category = 'Cafe' OR RC.category = 'Light meal')
      )
GROUP BY R.restaurant_id
```

Sub-query as condition

```
SELECT R.name, SUM(B.seats) AS total_seats FROM Restaurant R, Branch B, (
        SELECT DISTINCT restaurant_id FROM RestaurantCategory RC
        WHERE (RC.category = 'Cafe' OR RC.category = 'Light meal')
    ) R2
WHERE R.restaurant_id = B.restaurant_id
      AND R.restaurant_id = R2.restaurant_id
GROUP BY R.restaurant_id
```

Sub-query as temporary table

```
SELECT R.name, SUM(B.seats) AS total_seats FROM Restaurant R, Branch B
WHERE R.restaurant_id = B.restaurant_id
      AND EXISTS (
        SELECT restaurant_id FROM RestaurantCategory RC
        WHERE (RC.category = 'Cafe' OR RC.category = 'Light meal')
              AND R.restaurant_id = RC.restaurant_id
      )
GROUP BY R.restaurant_id
```

Correlated sub-query

Exercise 4

- Find the average score of all visits given by each of the members, who have ever visited Starducks.

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Exercise 4

- Find the average score of all visits given by each of the members, who have ever visited Starducks.

Please draft the corresponding SQL before you continue

Exercise 4 - answer

```
SELECT
    M.name,
    AVG(V.score) AS average_score
FROM
    Member M,
    visits V
WHERE
    M.member_id = V.member_id AND M.member_id IN(
    SELECT
        V.member_id
    FROM
        visits V
    WHERE
        V.restaurant_id IN(
        SELECT
            restaurant_id
        FROM
            Restaurant
        WHERE NAME
            = 'Starducks'
    )
)
GROUP BY
    M.member_id
```

name	average_score
Todd	3.3333
Lonny	3.3333
Keith	5.2500
Royce	3.0000

You are encouraged to try other methods to achieve the same result

Example 5 – more nested query

- Find the average score of the restaurant(s) with the greatest number of branches.

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Example 5 – sub queries

- The problem could be broken down as follow.
  - Find the number of branches of restaurant

```
SELECT restaurant_id, COUNT(*) FROM Branch
GROUP BY restaurant_id
```

  - Find the restaurants that has the maximum number of branches

```
SELECT restaurant_id FROM Branch GROUP BY restaurant_id
HAVING COUNT(*) >= ALL(…find count…)
```

  - Find the average score of those restaurants

```
SELECT restaurant_id, AVG(score) FROM visits
WHERE restaurant_id IN (…find list of restaurants…)
GROUP BY restaurant_id
```

Example 5 – answer

restaurant_id	average_score
3	5.6250
4	3.0000

```
SELECT restaurant_id, COUNT(*) FROM Branch
GROUP BY restaurant_id
```

+

```
SELECT restaurant_id FROM Branch GROUP BY restaurant_id
HAVING COUNT(*) >= ALL(…find count…)
```

+

```
SELECT restaurant_id, AVG(score) FROM visits
WHERE restaurant_id IN (…find list of restaurants…)
GROUP BY restaurant_id
```

=

```
SELECT R.restaurant_id, AVG(score) AS average_score
FROM Restaurant R, visits V
WHERE R.restaurant_id = V.restaurant_id
      AND R.restaurant_id IN (
        SELECT restaurant_id FROM Branch B GROUP BY restaurant_id
        HAVING COUNT(*) >= ALL(
          SELECT COUNT(*) FROM Branch GROUP BY restaurant_id
        )
      )
GROUP BY R.restaurant_id
```

Exercise 5

- Find the number of visits by each member, who has given the greatest number of scores that is less than or equals to 5.

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Exercise 5

- Find the number of visits by each member, who has given the greatest number of scores that is less than or equals to 5.

Please draft the corresponding SQL before you continue

Exercise 5 – answer

```
SELECT
    M.name, COUNT(*) AS visit_count
FROM
    Member M, visits V
WHERE
    M.member_id = V.member_id AND M.member_id IN(
    SELECT
        member_id
    FROM
        visits
    WHERE
        score <= 5
    GROUP BY
        member_id
    HAVING
        COUNT(*) >= ALL(
        SELECT
            COUNT(*)
        FROM
            visits
        WHERE
            score <= 5
        GROUP BY
            member_id
    )
)
GROUP BY
    M.member_id
```

name	visit_count
Cleo	3
Evan	3

Note that these two parts must have the same conditions as we are comparing the count to the max count under the same condition.

Advanced SQL

Table Schemas

- **Restaurant** (*restaurant_id*, *name*))
- **RestaurantCategory** (*restaurant_id*, *category*)
  - Foreign key: {*restaurant_id*} references **Restaurant**
- **Branch** (*restaurant_id*, *branch_no*, *location*, *seats*)
  - Foreign key: {*restaurant_id*} references **Restaurant**
- **Member** (*member_id*, *name*, *birthday, joined, points*)
  - (*joined* is the year the member joined)
- **visits** (*visit_id*, *member_id*, *restaurant_id*, *branch_no*, *date*, *score*)
  - Foreign keys: {*restaurant_id, branch_no*} references **Branch**,
    {*member_id*} references **Member**

restaurant_id	name
1	McRonalds
2	PizzaHub
3	DeliItaly
4	UltraSandwich
5	Starducks

Restaurant

restaurant_id	branch_no	location	seats
1	1	Admiralty	10
1	2	Central	20
2	1	Causeway Bay	5
3	1	Admiralty	25
3	2	Wan Chai	45
3	3	Causeway Bay	35
4	1	Central	170
4	2	Admiralty	100
4	3	North Point	120
5	1	Central	80
5	2	Wan Chai	40

Branch

restaurant_id	category
1	Fast Food
1	Take Away
2	Italian
3	Cafe
3	Italian
4	Light meal
5	Cafe
5	Light meal
5	Take Away

RestaurantCategory

member_id	name	birthday	joined	points
1	Cleo	1983-09-25	2017	690
2	Evan	1988-03-28	1989	130
3	Todd	1966-06-22	1967	190
4	Lonny	1973-04-05	2016	10
5	Keith	1991-06-19	1992	380
6	Royce	1965-06-14	2006	300
7	Mavis	1977-08-21	1981	840
8	Alvin	1998-04-17	2008	900
9	Ira	1993-07-17	2015	100
10	Dino	1968-12-26	2012	150
11	A. Bella	1989-11-26	2016	20

Member

visit_id	member_id	restaurant_id	branch_no	date	score
1	1	4	3	2015-12-30	5
2	8	4	3	2016-01-15	4
3	1	2	1	2018-06-16	5
4	2	3	2	2018-06-14	5
5	1	2	1	2018-12-09	3
6	3	3	2	2018-01-16	6
7	2	4	1	2018-03-26	3
8	4	3	2	2018-08-02	7
9	5	3	1	2018-04-22	8
10	4	5	2	2018-05-10	0
11	5	5	2	2018-05-08	6
12	3	5	1	2018-06-21	1
13	6	5	1	2018-03-06	2
14	4	5	1	2018-02-12	3
15	7	4	2	2018-06-07	1
16	5	4	2	2018-12-04	2
17	3	4	2	2018-01-19	3
18	5	3	3	2018-01-27	5
19	7	3	3	2018-06-11	5
20	2	3	3	2018-11-25	5
21	6	3	3	2018-07-27	4

visits

Questions

1. Find all branches with at least one visit scoring less than 4.
   - List the restaurant name, the branch location, and the best score among the visits to the branch.
   - Order the list by the restaurant name, and then the location.
2. Find all members who has visited the restaurant branch(/branches) that received the lowest average score ever.
   - Show only the member name, order by the member name.
3. Find all members who have made at least two visits, show only the member name, the name of the restaurant of the member's first visit, and that of the last visit.
   - Show any of the restaurant names if the member visited more than one restaurant on the same day.
   - Order the list by the member name.
   - Show only the first 5 results.
4. Count the number of unique members visiting each of the restaurants. List the result by showing the restaurant name, and the number of members. A restaurant should be listed even if it is not visited by any member, in that case the count should be zero. Order the list by the number of unique members (descending), then by restaurant name.