SQL- Interview Preparation Questionnaire

1. How to select random 10 rows from a table?
2. Examine the following code. What will the value of the price be if the statement finds a NULL value? SELECT name, ISNULL(price, 50) FROM PRODUCTS.
3. What is COALESCE FUNCTION DOES IN SQL?
4. With SQL, how do you select all the records from a table named "Customers" where the "Last_Name" is alphabetically between (and including) "Brooks" and "Gray"?
5. What function to round a number to the smallest integer value greater than or equal to a number?
6. Which of the following is NOT TRUE about the ON clause?
7. What is a Cartesian join and how it can be converted into an inner join?
8. What is the difference between Nested & Correlated Queries?
9. What is Case Function?
10. How do you create a temporary table in MySQL?

Database queries

1. The Employee table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

```
+----+-------+--------+-----------+

| Id | Name | Salary | ManagerId |

+----+-------+--------+-----------+

| 1 | Joe | 70000 | 3 |

| 2 | Henry | 80000 | 4 |

| 3 | Sam | 60000 | NULL |

| 4 | Max | 90000 | NULL |

+----+-------+--------+---------
```

Given the Employee table, write a SQL query that finds out employees who earn

more than their managers. For the above table, Joe is the only employee who

earns more than his manager.

```
+----------+

| Employee |

+----------+

| Joe |
```

2. Given a Weather table, write a SQL query to find all dates' Ids with higher temperature compared to its previous (yesterday's) dates.

Id(INT)	RecordDate(DATE)	Temperature(INT)
1	2015-01-01	10
2	2015-01-02	25
3	2015-01-03	20
4	2015-01-04	30

For example, return the following Ids for the above Weather table:

Id
2
4

3. Table: Activity

Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

(player_id, event_date) is the primary key of this table. This table shows the activity of players of some game. Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.

Write a SQL query that reports the device that is first logged in for each player.

The query result format is in the following example:

Activity table:

```
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1 | 2 | 2016-03-01 | 5 |
Description
| 1 | 2 | 2016-05-02 | 6 |
| 2 | 3 | 2017-06-25 | 1 |
| 3 | 1 | 2016-03-02 | 0 |
| 3 | 4 | 2018-07-03 | 5 |
+-----------+-----------+------------+--------------+
```

Result table:

```
| player_id | device_id |
| 1 | 2 |
| 2 | 3 |
| 3 | 1 |
```

3. Table: Activity

```
  +--------------+---------+
  | Column Name | Type |
  +--------------+---------+
  | player_id | int |
  | device_id | int |
  | event_date | date |
  | games_played | int |
+--------------+---------+
```

(player_id, event_date) is the primary key of this table. This table shows the activity of players of some games. Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.

Write an SQL query that reports for each player and dates, how many games played

so far by the player. That is, the total number of games played by the player

until that date. Check the example for clarity.

The query result format is in the following example:

Activity table:

```
+-----------+-----------+------------+--------------+

| player_id | device_id | event_date | games_played |

+-----------+-----------+------------+--------------+

| 1 | 2 | 2016-03-01 | 5 |

| 1 | 2 | 2016-05-02 | 6 |

| 1 | 3 | 2017-06-25 | 1 |

| 3 | 1 | 2016-03-02 | 0 |

| 3 | 4 | 2018-07-03 | 5 |

+-----------+-----------+------------+--------------+
```

Result table:

```
+-----------+------------+---------------------+

| player_id | event_date | games_played_so_far |

+-----------+------------+---------------------+

| 1 | 2016-03-01 | 5 |

| 1 | 2016-05-02 | 11 |

| 1 | 2017-06-25 | 12 |

| 3 | 2016-03-02 | 0 |

| 3 | 2018-07-03 | 5 |

+-----------+------------+---------------------+
```

For the player with id 1, 5 + 6 = 11 games played by 2016-05-02, and 5 + 6 + 1 =12 games played by 2017-06-25. For the player with id 3, 0 + 5 = 5 games played by 2018-07-03. Note that for each player we only care about the days when the player logged in.

4. In social network like Facebook or Twitter, people send friend requests and accept others' requests as well. Now given two tables as below:

Table: friend_request

sender_id	send_to_id	request_date
1	2	2016_06-01
1	3	2016_06-01
1	4	2016_06-01
2	3	2016_06-02
3	4	2016-06-09

Table: request_accepted

requester_id	accepter_id	accept_date
1	2	2016_06-03
1	3	2016-06-08
2	3	2016-06-08
3	4	2016-06-09
3	4	2016-06-10

Write a query to find the overall acceptance rate of requests rounded to 2 decimals, which is the number of acceptance divide the number of requests. For the sample data above, your query should return the following result.

accept_rate
0.80

Note:

The accepted requests are not necessarily from the table friend_request. In this case, you just need to simply count the total accepted requests (no matter whether they are in the original requests), and divide it by the number of requests to get the acceptance rate. It is possible that a sender sends multiple requests to the same receiver, and a request could be accepted more than once. In this case, the 'duplicated' requests or acceptances are only counted once. If there is no requests at all, you should return 0.00 as the accept_rate.

Explanation: There are 4 unique accepted requests, and there are 5 requests in

total. So the rate is 0.80.

Follow-up:

Can you write a query to return the accept rate but for every month?

How about the cumulative accept rate for every day?

5. Several friends at a cinema ticket office would like to reserve consecutive

available seats. Can you help to query all the consecutive available seats order by the seat_id

using the following cinema table?

| seat_id | free |

| 1 | 1 |

| 2 | 0 |

| 3 | 1 |

| 4 | 1 |

| 5 | 1 |

Your query should return the following result for the sample case above.

| seat_id |

|---------|

| 3 |

| 4 |

| 5 |

Note:

The seat_id is an auto increment int, and free is bool ('1' means free, and '0'

means occupied.). Consecutive available seats are more than 2(inclusive) seats consecutively Available.

6. Given three tables: salesperson, company, orders.

Output all the names in the table salesperson, who didn’t have sales to company 'RED'.

Example

Input

Table: salesperson

sales_id	name	salary	commission_rate	hire_date
1	John	100000	6	4/1/2006
2	Amy	120000	5	5/1/2010
3	Mark	65000	12	12/25/2008
4	Pam	25000	25	1/1/2005
5	Alex	50000	10	2/3/2007

7. The table salesperson holds the salesperson’s information. Every salesperson has a sales_id and a name.

Table: Company

com_id	name	city
1	RED	Boston
2	ORANGE	New York
3	YELLOW	Boston
4	GREEN	Austin

The table company holds the company information. Every company has a com_id and

a name.

Table: orders

order_id	order_date	com_id	sales_id	amount

```
| 1 | 1/1/2014 | 3 | 4 | 100000 |

| 2 | 2/1/2014 | 4 | 5 | 5000 |

| 3 | 3/1/2014 | 1 | 1 | 50000 |

| 4 | 4/1/2014 | 1 | 4 | 25000 |

+----------+----------+---------+----------+--------+
```

The table orders holds the sales record information, salesperson and customer company are represented by sales_id and com_id.

output

```
+------+

| name |

+------+

| Amy |

| Mark |

| Alex |

+------+
```

Explanation

According to order '3' and '4' in table orders, it is easy to tell only salesperson 'John' and 'Alex' have sales to company 'RED', so we need to output all the other names in table salesperson.

8. Given a table tree, id is identifier of the tree node and p_id is its parent

node's id.-

```
| id | p_id |

+----+------+

| 1 | null |

| 2 | 1 |

| 3 | 1 |

| 4 | 2 |

| 5 | 2 |

+----+------+
```

Each node in the tree can be one of three types:

Leaf: if the node is a leaf node.

Root: if the node is the root of the tree.

Inner: If the node is neither a leaf node nor a root node.

Write a query to print the node id and the type of the node. Sort your output by the node id. The result for the above sample is:

```
| id | Type |
| 1 | Root |
| 2 | Inner|
| 3 | Leaf |
| 4 | Leaf |
| 5 | Leaf |
```

Explanation

Node '1' is root node, because its parent node is NULL and it has child node '2' and '3'.

Node '2' is inner node, because it has parent node '1' and child node '4' and '5'.

Node '3', '4' and '5' is Leaf node, because they have parent node and they don't have child node.

And here is the image of the sample tree as below:

```
 1
 / \
 2 3
 / \
 4 5
```

Note:If there is only one node on the tree, you only need to output its root attributes.

Explanation Node '1' is root node, because its parent node is NULL and it has child node '2' and '3'. Node '2' is inner node, because it has parent node '1' and child node '4' and '5'. Node '3', '4' and '5' is Leaf node, because they have parent node and they don't have child node. And here is the image of the sample tree as below: 1 / \ 2 3 / \ 4 5 Note If there is only one node on the tree, you only need to output its root attributes.

9. Table point_2d holds the coordinates (x,y) of some unique points (more than two) in a plane.

Write a query to find the shortest distance between these points rounded to 2

decimals.

x	y
-1	-1
0	0
-1	-2

The shortest distance is 1.00 from point (-1,-1) to (-1,2). So the output should

be:

shortest
1.00

Note: The longest distance among all the points is less than 10000.

10. Table my_numbers contains many numbers in column num including duplicated ones .Can you write a SQL query to find the biggest number, which only appears once?

num
8
8
3
3
1
4
5
6

For the sample data above, your query should return the following result:

11. |num|

```
+---+
| 6 |
```

Note: If there is no such number, just output null

12. X city opened a new cinema, many people would like to go to this cinema. The cinema also gives out a poster indicating the movies’ ratings and descriptions. Please write a SQL query to output movies with an odd numbered ID and a description that is not 'boring'. Order the result by rating.

For example, table cinema:

```
+---------+-----------+-------------+-----------+
| id | movie | description | rating |
+---------+-----------+-------------+-----------+
| 1 | War | great 3D | 8.9 |
| 2 | Science | fiction | 8.5 |
| 3 | irish | boring | 6.2 |
| 4 | Ice song | Fantacy | 8.6 |
| 5 | House card| Interesting| 9.1 |
+---------+-----------+-------------+-----------+
```

For the example above, the output should be:

```
+---------+-----------+-------------+-----------+
| id | movie | description | rating |
+---------+-----------+-------------+-----------+
| 5 | House card| Interesting| 9.1 |
| 1 | War | great 3D | 8.9 |
+---------+-----------+-------------+-----------+
```

13. Mary is a teacher in a middle school and she has a table seat storing students' names and their corresponding seat ids. The column id is continuous increment. Mary wants to change seats for the adjacent students.

Can you write a SQL query to output the result for Mary?

```
+---------+---------+
| id | student |
+---------+---------+
| 1 | Abbot |
| 2 | Doris |
| 3 | Emerson |
| 4 | Green |
| 5 | Jeames |
+---------+---------+
```

For the sample input, the output is:

```
+---------+---------+
| id | student |
+---------+---------+
```

14.

```
| 1 | Doris |
| 2 | Abbot |
| 3 | Green |
| 4 | Emerson |
| 5 | Jeames
```

Note: If the number of students is odd, there is no need to change the last one's seat.

15.

```
+-------------+-------+
| Column Name | Type |
+-------------+-------+
| sale_id | int |
| product_id | int |
| year | int |
| quantity | int |
| price | int |
+-------------+-------+
```

sale_id is the primary key of this table. product_id is a foreign key to Product table.

Note that the price is per unit.

Table: Product

Column Name	Type
product_id	int
product_name	varchar

product_id is the primary key of this table.

16. Description

Write an SQL query that selects the product id, year, quantity, and price for the first year of every product sold.

The query result format is in the following example:

Sales table:

sale_id	product_id	year	quantity	price
1	100	2008	10	5000
2	100	2009	12	5000
7	200	2011	15	9000

Product table:

product_id	product_name
100	Nokia
200	Apple
300	Samsung

Result table:

product_id	first_year	quantity	price
100	2008	10	5000
200	2011	15	9000

17. Table: Project

```
+-------------+---------+

| Column Name | Type |

+-------------+---------+

| project_id | int |

| employee_id | int |

+-------------+---------+
```

(project_id, employee_id) is the primary key of this table. employee_id is a foreign key to Employee table.

Table: Employee

```
| Column Name | Type |

| employee_id | int |

| name | varchar |

| experience_years | int |


+-----------------+---------+
```

employee_id is the primary key of this table.

Write an SQL query that reports all the projects that have the most employees.

The query result format is in the following example:

Project table:

```
+-------------+-------------+

| project_id | employee_id |

+-------------+-------------+

| 1 | 1 |

| 1 | 2 |

| 1 | 3 |

| 2 | 1 |

| 2 | 4 |

+-------------+-------------+
```

Employee table:

```
+-------------+--------+------------------+

| employee_id | name | experience_years |

+-------------+--------+------------------+

| 1 | Khaled | 3 |

| 2 | Ali | 2 |

| 3 | John | 1 |

| 4 | Doe | 2 |

+-------------+--------+------------------+
```

Result table:

```
+-------------+

| project_id |

+-------------+

| 1 |

+-------------+
```

The first project has 3 employees while the second one has 2.

```
      action_date  | date |
     | action         | enum |
     | extra            | varchar |
+---------------+---------+
```

There is no primary key for this table, it may have duplicate rows.
The action column is an ENUM type of ('view', 'like', 'reaction', 'comment', 'report', 'share').
The extra column has optional information about the action such as a reason for report or a type of reaction.
Write an SQL query that reports the number of posts reported yesterday for each report reason. Assume today is 2019-07-05.
The query result format is in the following example:
Actions table:

```
+---------+---------+------------+--------+--------+
| user_id | post_id | action_date | action | extra |
+---------+---------+------------+--------+--------+
| 1       | 1       | 2019-07-01   | view   | null |
| 1       | 1        | 2019-07-01 | like | null |
| 1       | 1        | 2019-07-01 | share | null |
| 2       | 4       | 2019-07-04 | view | null |
| 2        | 4      | 2019-07-04 | report | spam |
| 3       | 4        | 2019-07-04 | view | null |
| 3       | 4        | 2019-07-04 | report | spam |
| 4        | 3       | 2019-07-02 | view | null |
| 4       | 3        | 2019-07-02 | report | spam |
| 5       | 2        | 2019-07-04 | view | null |
| 5       | 2       | 2019-07-04 | report | racism |
| 5       | 5        | 2019-07-04 | view | null |
| 5       | 5       | 2019-07-04 | report | racism |
+---------+---------+------------+--------+--------+
```

Result table:

```
+---------------+--------------+
| report_reason     |       report_count |
+---------------+--------------+
| spam                    |          1          |
| racism                   |         2          |
+---------------+--------------+
```

*Note that we only care about report reasons with non-zero number of reports.

20.

```
| Column Name | Type |
+---------------+---------+
| user_id | int |
| session_id | int |
| activity_date | date |
| activity_type | enum |
+---------------+---------+
```

There is no primary key for this table, it may have duplicate rows. The activity_type column is an ENUM of type ('open_session', 'end_session', 'scroll_down', 'send_message'). The table shows the user activities for a social media website. Note that each session belongs to exactly one user. Write an SQL query to find the daily active user count for a period of 30 days ending 2019-07-27 inclusively. A user was active on some day if he/she made at least one activity on that day.The query result format is in the following example:

Activity table:

```
+---------+-----------+--------------+---------------+
| user_id | session_id | activity_date | activity_type |
+---------+-----------+--------------+---------------+
| 1 | 1 | 2019-07-20 | open_session |
| 1 | 1 | 2019-07-20 | scroll_down |
| 1 | 1 | 2019-07-20 | end_session |
| 2 | 4 | 2019-07-20 | open_session |
| 2 | 4 | 2019-07-21 | send_message |
| 2 | 4 | 2019-07-21 | end_session |
| 3 | 2 | 2019-07-21 | open_session |
| 3 | 2 | 2019-07-21 | send_message |
| 3 | 2 | 2019-07-21 | end_session |
| 4 | 3 | 2019-06-25 | open_session |
| 4 | 3 | 2019-06-25 | end_session |
+---------+-----------+--------------+---------------+
```

Result table:

day	active_users
2019-07-20	2
2019-07-21	2

*Note that we do not care about days with zero active users.

21. Delivery table:

delivery_id	customer_id	order_date	customer_pref_delivery_date
1	1	2019-08-01	2019-08-02
2	5	2019-08-02	2019-08-02
3	1	2019-08-11	2019-08-11
4	3	2019-08-24	2019-08-26
5	4	2019-08-21	2019-08-22
6	2	2019-08-11	2019-08-13

Result table:

immediate_percentage
33.33

The orders with delivery id 2 and 3 are immediate while the others are scheduled.

22. Table: Delivery

Column Name	Type
delivery_id	int
customer_id	int
order_date	date
customer_pref_delivery_date	date

delivery_id is the primary key of this table. The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the same order date or after it). If the preferred delivery date of the customer is the same as the order date then the order is called immediately otherwise it's called scheduled. The first order of a customer is the order with the earliest order date that The customer made. It is guaranteed that a customer has exactly one first order. Write an SQL query to find the percentage of immediate orders in the first orders of all customers, rounded to 2 decimal places.

The query result format is in the following example:

Delivery table:

```
+-------------+-------------+------------+-----------------------------+
| delivery_id | customer_id | order_date | customer_pref_delivery_date |
+-------------+-------------+------------+-----------------------------+
| 1 | 1 | 2019-08-01 | 2019-08-02 |
| 2 | 2 | 2019-08-02 | 2019-08-02 |
| 3 | 1 | 2019-08-11 | 2019-08-12 |
| 4 | 3 | 2019-08-24 | 2019-08-24 |
| 5 | 3 | 2019-08-21 | 2019-08-22 |
| 6 | 2 | 2019-08-11 | 2019-08-13 |
| 7 | 4 | 2019-08-09 | 2019-08-09 |
+-------------+-------------+------------+-----------------------------+
```

Result table:

```
+----------------------+
| immediate_percentage |
+----------------------+
| 50.00 |
+----------------------+
```

Customer id 1 has a first order with delivery id 1 and it is scheduled.
Customer id 2 has a first order with delivery id 2 and it is immediate.
Customer id 3 has a first order with delivery id 5 and it is scheduled.
Customer id 4 has a first order with delivery id 7 and it is immediate.
Hence, half the customers have immediate first orders.

23. Table: Department

```
+---------------+---------+
| Column Name | Type |
+---------------+---------+
| id | int |
| revenue | int |
| month | varchar |
+---------------+---------+
```

(id, month) is the primary key of this table. The table has information about the revenue of each department per month. The month has values in ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]. Write an SQL query to reformat the table such that there is a department id column and a revenue column for each month. The query result format is in the following example:

Department table:

```
+------+---------+-------+
| id | revenue | month |
+------+---------+-------+
| 1 | 8000 | Jan |
| 2 | 9000 | Jan |
| 3 | 10000 | Feb |
| 1 | 7000 | Feb |
| 1 | 6000 | Mar |
+------+---------+-------+
```

Result table:

id	Jan_Revenue	Feb_Revenue	Mar_Revenue	...	Dec_Revenue
1	8000	7000	6000	...	null
2	9000	null	null	...	null
3	null	10000	null	...	null

Note that the result table has 13 columns (1 for the department id + 12 for the months).

24. Table: Queue

Column Name	Type
person_id	int
person_name	varchar
weight	int
turn	int

person_id is the primary key column for this table.This table has the information about all people waiting for an elevator. The person_id and turn columns will contain all numbers from 1 to n, where n is the number of rows in the table. The maximum weight the elevator can hold is 1000. Write an SQL query to find the person_name of the last person who will fit in the elevator without exceeding the weight limit. It is guaranteed that the person who is first in the queue can fit in the elevator. The query result format is in the following example:

Queue table

person_id	person_name	weight	turn
5	George Washington	250	1
3	John Adams	350	2
6	Thomas Jefferson	400	3
2	Will Johnliams	200	4
4	Thomas Jefferson	175	5
1	James Elephant	500	6

Result table

person_name
Thomas Jefferson

Queue table is ordered by turn in the example for simplicity.
In the example George Washington(id 5), John Adams(id 3) and Thomas Jefferson(id 6) will enter the elevator as their weight sum is 250 + 350 + 400 = 1000.
Thomas Jefferson(id 6) is the last person to fit in the elevator because he has the last turn in these three people.

25. Table: Queries

Column Name	Type
query_name	varchar
result	varchar
position	int
rating	int

There is no primary key for this table, it may have duplicate rows.
This table contains information collected from some queries on a database.
The position column has a value from 1 to 500.
The rating column has a value from 1 to 5. Query with rating less than 3 is a poor query.
We define query quality as:
The average of the ratio between query rating and its position. We also define poor query percentage as: The percentage of all queries with rating less than 3.
Write an SQL query to find each query_name, the quality and poor_query_percentage.
Both quality and poor_query_percentage should be rounded to 2 decimal places.

The query result format is in the following example:
Queries table:

query_name	result	position	rating
Dog	Golden Retriever	1	5
Dog	German Shepherd	2	5
Dog	Mule	200	1
Cat	Shirazi	5	2
Cat	Siamese	3	3
Cat	Sphynx	7	4

Result table:

query_name	quality	poor_query_percentage
Dog	2.50	33.33
Cat	0.66	33.33

Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50
Dog queries poor_ query_percentage is (1 / 3) * 100 = 33.33
Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66
Cat queries poor_ query_percentage is (1 / 3) * 100 = 33.33