LeetCode SQL Practice - Basic

Aggregate Functions(By sanjana thakur)

Question 1: 620. Not Boring Movies

Problem:

Given a table `Cinema` with columns `id`, `movie`, `description`, and `rating`, find the movies with odd-numbered IDs and a description that is not "boring". The result should be ordered by `rating` in descending order.

Input Table:

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

Expected Output:

id	movie	description	rating
5	House card	Interesting	9.1
1	War	great 3D	8.9

SQL Query:

```
SELECT
    id,
    movie,
    description,
    rating FROM
    Cinema WHERE
    id % 2 = 1
```

```
AND description != 'boring' ORDER BY
rating DESC;
```

Question 2: 1251. Average Selling Price

Problem:

Given tables `Prices` and `UnitsSold`, find the average selling price for each product. The `average_price` should be rounded to 2 decimal places.

Input Tables:

`Prices`:

product_id	start_date	end_date	price
1	2019-02-17	2019-02-28	5
1	2019-03-01	2019-03-22	20
2	2019-02-01	2019-02-20	15
2	2019-02-21	2019-03-31	30

`UnitsSold`:

product_id	purchase_date	units
1	2019-02-25	100
1	2019-03-01	15
2	2019-02-10	200
2	2019-03-22	30

Expected Output:

product_id	average_price
1	6.96
2	16.96

SQL Query:

```
SELECT
    p.product_id,
    ROUND(COALESCE(SUM(u.units * p.price) / SUM(u.units), 0), 2) AS
average_priceFROM
    Prices pLEFT JOIN
    UnitsSold uON
    p.product_id = u.product_id
    AND u.purchase_date BETWEEN p.start_date AND p.end_dateGROUP BY
    p.product_id;
```

Question 3: 1075. Project Employees I

Problem:

Given tables `Project` and `Employee`, find the average experience years of all the employees for each project. The `average_years` should be rounded to 2 digits.

Input Tables:

`Project`:

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

`Employee`:

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	1
4	Doe	2

Expected Output:

project_id	average_years
1	2.00
2	2.50

```
SELECT
```

```
p.project_id,
ROUND(AVG(e.experience_years), 2) AS average_yearsFROM
Project pJOIN
Employee eON
p.employee_id = e.employee_idGROUP BY
p.project_id;
```

Question 4: 1633. Percentage of Users Attended a Contest

Problem:

Given tables `Users` and `Register`, find the percentage of the users registered in each contest rounded to two decimals. The result should be ordered by `percentage` in descending order and by `contest_id` in ascending order in case of a tie.

Input Tables:

`Users`:

user_id	user_name
6	Alice
2	Bob
7	Alex

`Register`:

contest_id	user_id
215	6
209	2
208	2
210	6
208	6
209	7
209	6
215	7
208	7
210	2
207	2
210	7

Expected Output:

contest_id	percentage
208	100.0
209	100.0
210	100.0
215	66.67
207	33.33

SQL Query:

```
sql
SELECT
    r.contest_id,
    ROUND(COUNT(DISTINCT r.user_id) * 100.0 / (SELECT COUNT(*) FROM
Users), 2) AS percentageFROM
    Register rGROUP BY
    r.contest_idORDER BY
    percentage DESC,
    r.contest_id ASC;
```

Question 5: 1211. Queries Quality and Percentage

Problem:

Given a table `Queries` with columns `query_name`, `result`, `position`, and `rating`, calculate `quality` as the average of the ratio between `rating` and `position`, and `poor_query_percentage` as the percentage of all queries with `rating` less than 3. Both should be rounded to 2 decimal places.

Input Table:

query_name	result	position	rating
Dog	Golden Retriever	1	5
Dog	German Shepherd	2	5
Dog	Mule	200	1
Cat	Shirazi	1	3
Cat	Persian	10	2
Bird	Sparrow	1	4

Expected Output:

query_name	quality	poor_query_percentage
Dog	2.50	33.33
Cat	0.30	50.00
Bird	4.00	0.00

SQL Query:

```
SELECT
    query_name,
    ROUND(AVG(rating * 1.0 / position), 2) AS quality,
    ROUND(SUM(CASE WHEN rating < 3 THEN 1 ELSE 0 END) * 100.0 /
COUNT(*), 2) AS poor_query_percentageFROM
    QueriesGROUP BY
    query_name;
```

6. Question 1193: Monthly Transactions I

Problem Statement:

Write an SQL query to find, for each month and country, the number of transactions and their total amount, the number of approved transactions, and their total amount. Return the result table in any order.

Schema:

- **Table:** `Transactions`
  - `id` (int)
  - `country` (varchar)
  - `state` (enum: ["approved", "declined"])
  - `amount` (int)
  - `trans_date` (date)

Example:

Input:

`Transactions` table:

id	country	state	amount	trans_date
121	US	approved	1000	2018-12-18
122	US	declined	2000	2018-12-19
123	US	approved	2000	2019-01-01
124	DE	approved	2000	2019-01-07

Output:

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2018-12	US	2	1	3000	1000
2019-01	US	1	1	2000	2000

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2019-01	DE	1	1	2000	2000

Solution:

```
SELECT

    DATE_FORMAT(trans_date, '%Y-%m') AS month,
    country,
    COUNT(id) AS trans_count,
    SUM(state = 'approved') AS approved_count,
    SUM(amount) AS trans_total_amount,
    SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END) AS
approved_total_amountFROM
    TransactionsGROUP BY
    month, country;
```

7. Question 1174: Immediate Food Delivery II

Problem Statement:

Write a solution to find the percentage of immediate orders in the first orders of all customers, rounded to 2 decimal places.

Schema:

- **Table:** `Delivery`
  - `delivery_id` (int)
  - `customer_id` (int)
  - `order_date` (date)
  - `customer_pref_delivery_date` (date)

Example:

Input:

`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

Output:

immediate_percentage
50.00

Solution:

```
WITH FirstOrders AS (

    SELECT
        customer_id,
        MIN(order_date) AS first_order_date
    FROM
        Delivery
    GROUP BY
        customer_id
),
ImmediateOrders AS (
    SELECT
        f.customer_id,
        CASE
            WHEN d.order_date = d.customer_pref_delivery_date THEN 1
            ELSE 0
        END AS is_immediate
    FROM
        FirstOrders f
    JOIN
        Delivery d
    ON
        f.customer_id = d.customer_id AND f.first_order_date = d.order_date
)SELECT
    ROUND(SUM(is_immediate) * 100.0 / COUNT(*), 2) AS
immediate_percentageFROM
    ImmediateOrders;
```

8. Question 550: Game Play Analysis IV

Problem Statement:

Write a solution to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.

Schema:

- **Table:** `Activity`
  - `player_id` (int)
  - `device_id` (int)
  - `event_date` (date)
  - `games_played` (int)

Example:

Input:

`Activity` table:

player_id	device_id	event_date	games_played
1	2	2016-03-01	5

player_id	device_id	event_date	games_played
1	2	2016-03-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

Output:

fraction
0.33

Solution

```
WITH FirstLogin AS (

    SELECT
        player_id,
        MIN(event_date) AS first_login_date
    FROM
        Activity
    GROUP BY
        player_id
),
NextDayLogin AS (
    SELECT
        f.player_id,
        CASE
            WHEN a.event_date = DATE_ADD(f.first_login_date, INTERVAL 1 DAY)
THEN 1
            ELSE 0
        END AS logged_next_day
    FROM
        FirstLogin f
```

```
    LEFT JOIN
        Activity a
    ON
        f.player_id = a.player_id
        AND a.event_date = DATE_ADD(f.first_login_date, INTERVAL 1 DAY)
)SELECT
    ROUND(SUM(logged_next_day) * 1.0 / COUNT(*), 2) AS fractionFROM
    NextDayLogin;
```