Project Employees I

✅ **Task:**

For each project_id, calculate the **average experience** (in years) of all employees working on that project. Round the result to **2 decimal places**.

🧩 **Tables:**

- Project(project_id, employee_id)
- Employee(employee_id, name, experience_years)

✅ **SQL Query:**

```
SELECT
  p.project_id,
  ROUND(AVG(e.experience_years), 2) AS average_experience
FROM
  Project p
JOIN
  Employee e ON p.employee_id = e.employee_id
GROUP BY
  p.project_id;
```

🔍 **Explanation:**

- We **join** the Project and Employee tables on employee_id to access each employee's experience.
- Then we **group by project_id** and calculate the **average experience**.
- ROUND(..., 2) ensures the result is rounded to **2 decimal places** as required.

Accepted Runtime: 106 ms

• Case 1

Input

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

Output

project_id	average_years
1	2
2	2.5

Expected

project_id	average_years
1	2
2	2.5

Percentage of Users Attended a Contest,

your task is to calculate the percentage of total users who registered for **each contest**, rounded to **2 decimal places**.

🧩 **Tables:**

- **Users(user_id, user_name)**
- **Register(contest_id, user_id)**

✅ **SQL Query:**

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

🔍 **Explanation:**

- COUNT(DISTINCT r.user_id) → counts how many unique users registered for a contest.
- (SELECT COUNT(*) FROM Users) → gets the total number of users.
- Multiply by 100.0 to convert to percentage, and ROUND(..., 2) for 2 decimal places.

- Ordered by percentage DESC and contest_id ASC as required.

Case 1

Input

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

View less

Output

contest_id	percentage
208	100
209	100
210	100
215	66.67
207	33.33

Expected

contest_id	percentage
208	100
209	100
210	100
215	66.67
207	33.33

Queries Quality and Percentage

we need to compute the **query quality** and the **poor query percentage** for each query_name.

📘 **Definitions:**

1. **Query Quality** = Average of (rating / position) for a given query name
2. **Poor Query Percentage** = (Number of queries with rating < 3) ÷ (Total queries for that query_name) × 100

✅ **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_percentage
FROM
  Queries
GROUP BY
  query_name;
```

🔍 **Explanation:**

- AVG(rating * 1.0 / position) computes the quality for each query_name.
- CASE WHEN rating < 3 THEN 1 ELSE 0 END counts poor queries.
- Divide by total count and multiply by 100 to get a percentage.
- Both are rounded to 2 decimal places using ROUND(..., 2).

Accepted Runtime: 89 ms

• Case 1

Input

Queries =

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

Output

query_name	quality	poor_query_percentage
Dog	2.5	33.33
Cat	0.66	33.33

Expected

query_name	quality	poor_query_percentage
Dog	2.5	33.33
Cat	0.66	33.33

Monthly Transactions I

we need to aggregate transaction data by **month** and **country**, and provide:

- Total number of transactions
- Total amount of all transactions
- Number of **approved** transactions
- Total amount of **approved** transactions

✅ **SQL Query:**

```
SELECT
  DATE_FORMAT(trans_date, '%Y-%m') AS month,
  country,
  COUNT(*) AS trans_count,
  SUM(amount) AS trans_total_amount,
  SUM(CASE WHEN state = 'approved' THEN 1 ELSE 0 END) AS approved_count,
  SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END) AS approved_amount
FROM
  Transactions
GROUP BY
  month, country;
```

🔍 **Explanation:**

- DATE_FORMAT(trans_date, '%Y-%m') extracts the **month and year** (e.g., '2019-01').
- COUNT(*) = total number of transactions.
- SUM(amount) = total transaction amount.
- SUM(CASE WHEN state = 'approved' THEN 1 ELSE 0 END) = count of approved transactions.

- SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END) = total approved amount.

Accepted Runtime: 139 ms

• Case 1

Input

Transactions =

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
2019-01	DE	1	1	2000	2000

Expected

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
2019-01	DE	1	1	2000	2000

Immediate Food Delivery II

Identify the **first order** (earliest order_date) for each customer_id.

1. Among those first orders, calculate the percentage where order_date = customer_pref_delivery_date (i.e., **immediate orders**).
2. Round the final percentage to **2 decimal places**.

✅ **SQL Query:**

```
WITH FirstOrders AS (
  SELECT *
  FROM Delivery
  WHERE (customer_id, order_date) IN (
    SELECT customer_id, MIN(order_date)
    FROM Delivery
    GROUP BY customer_id
  )
)

SELECT
  ROUND(
    SUM(CASE WHEN order_date = customer_pref_delivery_date THEN 1 ELSE 0 END) * 100.0 / COUNT(*),
    2
  ) AS immediate_percentage
FROM FirstOrders;
```

🔍 **Explanation:**

- **CTE FirstOrders** selects each customer's first order using MIN(order_date) and filters the main table.
- We then:
    - Count immediate orders with order_date = customer_pref_delivery_date.
    - Divide by total first orders using COUNT(*).
    - Multiply by 100.0 to convert to a percentage and round to 2 decimals.

Accepted Runtime: 66 ms

• Case 1

Input

Delivery =

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

View less

Output

immediate_percentage
50

Expected

immediate_percentage
50

Contribute a testcase