Game Play Analysis IV.

✅ **Objective:**

Calculate the **fraction of players** who logged in again on the **day after their first login date**, rounded to **2 decimal places**.

✅ **Table: Activity**

Column Name	Type	Description
player_id	int	Player's unique ID
device_id	int	Device used for login
event_date	date	Date the player logged in
games_played	int	Number of games played during the session

- Primary Key: (player_id, event_date)
- Each row is **one login session**.

✅ **Approach:**

1. **Find each player's first login date** using MIN(event_date).
2. **Self-join** the Activity table to find players who logged in **exactly the next day**.
3. Count:
    - The number of such players (numerator)
    - Total number of players (denominator)
4. Calculate the **fraction** and **round** to 2 decimal places.

✅ **SQL Query:**

```
SELECT
```

```
 ROUND(COUNT(DISTINCT a.player_id) / COUNT(DISTINCT b.player_id), 2) AS fraction
FROM
 (SELECT player_id, MIN(event_date) AS first_login
  FROM Activity
  GROUP BY player_id) b
JOIN
 Activity a
ON a.player_id = b.player_id
AND a.event_date = DATE_ADD(b.first_login, INTERVAL 1 DAY);
```

✅ **Explanation:**

- b: Subquery that gets each player's first login date.
- a: Main table, used to check if the player logged in **on the next day**.
- DATE_ADD(first_login, INTERVAL 1 DAY): Computes the day after the first login.
- COUNT(DISTINCT a.player_id): Players who logged in the next day.
- COUNT(DISTINCT b.player_id): Total number of players (all who had at least one login).
- ROUND(..., 2): Rounds result to 2 decimal places.

Accepted Runtime: 122 ms

• Case 1

Input

Activity =

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

Expected

fraction
0.33

Number of Unique Subjects Taught by Each Teacher

✅ **Objective:**

Determine how many **unique subjects** each teacher teaches.

✅ **Table: Teacher**

Column Name	Type	Description
teacher_id	int	ID of the teacher
subject_id	int	ID of the subject
dept_id	int	ID of the department

- Primary Key: (subject_id, dept_id)
- This means the same subject might be taught in multiple departments, but each subject-department pair is unique.

✅ **Approach:**

1. You need to count how many **distinct subjects** (subject_id) each teacher teaches.
2. Group by teacher_id.
3. Use COUNT(DISTINCT subject_id) to avoid double-counting subjects taught in multiple departments.

✅ **SQL Query:**

```
SELECT
 teacher_id,
 COUNT(DISTINCT subject_id) AS cnt
FROM
 Teacher
```

GROUP BY

teacher_id;

✅ **Explanation:**

- GROUP BY teacher_id: Calculates results per teacher.
- COUNT(DISTINCT subject_id): Counts only unique subjects, even if the same subject is listed multiple times with different departments.

• Case 1

Input

Teacher =

teacher_id	subject_id	dept_id
1	2	3
1	2	4
1	3	3
2	1	1
2	2	1
2	3	1
2	4	1

View less

Output

teacher_id	cnt
1	2
2	4

Expected

teacher_id	cnt
1	2
2	4

User Activity for the Past 30 Days I

✅ **Objective:**

Find the number of **distinct users active** on each day in the **30-day period ending on 2019-07-27 (inclusive)**.

✅ **Table: Activity**

Column Name	Type	Description
user_id	int	ID of the user
session_id	int	ID of the session
activity_date	date	Date the activity occurred
activity_type	enum	Type of activity ('open_session', 'end_session', etc.)

- May contain duplicate rows.
- One session belongs to one user.

✅ **Definition:**

- A user is **active on a day** if they performed **at least one activity** that day.
- Need to **count distinct users per day** from **2019-06-28 to 2019-07-27**.

✅ **SQL Query:**

```
SELECT
 activity_date,
 COUNT(DISTINCT user_id) AS active_users
FROM
 Activity
```

```
WHERE
  activity_date BETWEEN '2019-06-28' AND '2019-07-27'
GROUP BY
  activity_date;
```

✅ **Explanation:**

- WHERE activity_date BETWEEN ...: Filters data to the last 30 days.
- COUNT(DISTINCT user_id): Counts unique users per day.
- GROUP BY activity_date: Groups by each date to calculate daily active users.

• Case 1

Input

Activity =

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

View more

Output

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

Expected

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

Product Sales Analysis III"

✅ **Objective:**

Find **all sales records** corresponding to the **first year** each product was sold.

- If a product was sold multiple times in its first year, **include all such records**.

✅ **Tables Involved:**

🟦 **Sales**

Column Name	Type	Description
sale_id	int	Unique sale ID (combined with year)
product_id	int	Refers to product
year	int	Year of the sale
quantity	int	Quantity sold
price	int	Price per unit

- (sale_id, year) is the primary key.
- product_id is a foreign key to the Product table.

🟦 **Product**

Column Name	Type
product_id	int
product_name	varchar

✅ **Approach:**

1. **Find the first year** each product was sold using a subquery or CTE.
2. **Join** this result back with the Sales table to get all sales from that first year.

✅ **SQL Query:**

```
SELECT
 s.product_id,
 s.year,
 s.quantity,
 s.price
FROM
 Sales s
JOIN (
 SELECT
  product_id,
  MIN(year) AS first_year
 FROM
  Sales
 GROUP BY
  product_id
) AS first_sales
ON
 s.product_id = first_sales.product_id
 AND s.year = first_sales.first_year;
```

✅ **Explanation:**

- The subquery gets the **earliest year** (MIN(year)) each product_id was sold.
- The outer query **joins** back with Sales to filter only those records that occurred in that **first year**.

- Multiple rows may be returned for the same product if it had **multiple sales in its first year**.

• Case 1

Input

Sales =

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

Product =

product_id	product_name
100	Nokia
200	Apple
300	Samsung

Output

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

Expected

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