STRATASCRATCH MEDIUM QUESTIONS

# Find the top 10 ranked songs in 2010

```
select distinct year_rank,
group_name,
song_name
from billboard_top_100_year_end
where year = 2010
group by 1
limit 10;
```

| year_rank | group_name | song_name |
|---|---|---|
| 1 | Ke$ha | TiK ToK |
| 2 | Lady Antebellum | Need You Now |
| 3 | Train | Hey, Soul Sister |
| 4 | Katy Perry feat. Snoop Dogg | California Gurls |

Find songs that have ranked in the top position. Output the track name and the number of times it ranked at the top. Sort your records by the number of times the song was in the top position in descending order.

```
select trackname,
count(*)
from spotify_worldwide_daily_song_ranking AS top_ranking
where position = 1
group by 1
order by 2 DESC;
```

| trackname | count(*) |
|---|---|
| HUMBLE. | 7 |
| Bad and Boujee (feat. Lil Uzi Vert) | 1 |
| Look What You Made Me Do | 1 |

Output share of US users that are active. Active users are the ones with an "open" status in the table.

```
WITH CTE1 AS (
    SELECT *
    FROM fb_active_users
    WHERE country = 'USA' AND status = 'open'
),
CTE2 AS (
    SELECT *
    FROM fb_active_users
    WHERE country = 'USA'
)
SELECT (SELECT COUNT(*) FROM CTE1) / (SELECT COUNT(*) FROM CTE2);
```

| (SELECT COUNT(*) FROM CTE1) / (SELECT COUNT(*) FROM CTE2) |
|---|
| 0.5 |

---

Calculate each user's average session time. A session is defined as the time difference between a page_load and page_exit. For simplicity, assume a user has only 1 session per day and if there are multiple of the same events on that day, consider only the latest page_load and earliest page_exit, with an obvious restriction that load time event should happen before exit time event . Output the user_id and their average session time.

```
SELECT user_id,

       AVG(TIMESTAMPDIFF(SECOND, p_load_time, p_exit_time)) AS
avg1

FROM

(

    SELECT user_id,

           DATE(timestamp),

           MAX(CASE WHEN action = 'page_load' THEN timestamp ELSE
NULL END) AS p_load_time,

           MIN(CASE WHEN action = 'page_exit' THEN timestamp ELSE
NULL END) AS p_exit_time

    FROM facebook_web_log

    GROUP BY user_id, DATE(timestamp)

) AS t

GROUP BY user_id

HAVING avg1 IS NOT NULL;
```

| user_id | avg1 |
|---|---|
| 0 | 1883.5 |
| 1 | 35 |

Identify projects that are at risk for going overbudget. A project is considered to be overbudget if the cost of all employees assigned to the project is greater than the budget of the project.

You'll need to prorate the cost of the employees to the duration of the project. For example, if the budget for a project that takes half a year to complete is $10K, then the total half-year salary of all employees assigned to the project should not exceed $10K. Salary is defined on a yearly basis, so be careful how to calculate salaries for the projects that last less or more than one year.

Output a list of projects that are overbudget with their project name, project budget, and prorated total employee expense (rounded to the next dollar amount).

HINT: to make it simpler, consider that all years have 365 days. You don't need to think about the leap years.

```
WITH cte1 AS (

    SELECT p.title, p.budget, e.salary,

    (e.salary / 365) AS 1_day_of_sal,

    DATEDIFF(p.end_date, p.start_date) AS total_days

    FROM linkedin_projects AS p

    JOIN linkedin_emp_projects AS ep ON p.id = ep.project_id

    JOIN linkedin_employees AS e ON e.id = ep.emp_id

)



SELECT title, budget, prorated_employee_expense

FROM (

    SELECT title, budget, CEIL(SUM(1_day_of_sal * total_days)) AS
prorated_employee_expense

    FROM cte1

    GROUP BY title, budget

) AS t

WHERE budget < prorated_employee_expense;
```

| title | budget | prorated_employee_expense |
|---|---|---|
| Project1 | 29498 | 36293 |
| Project2 | 32487 | 52870 |
| Project4 | 15776 | 30656 |

```
WITH cte1 AS (
    SELECT p.title, p.budget, e.salary,
    (e.salary / 365) AS 1_day_of_sal,
    DATEDIFF(p.end_date, p.start_date) AS total_days
    FROM linkedin_projects AS p
    JOIN linkedin_emp_projects AS ep ON p.id = ep.project_id
    JOIN linkedin_employees AS e ON e.id = ep.emp_id
)

SELECT title, budget, prorated_employee_expense
FROM (
    SELECT title, budget, CEIL(SUM(1_day_of_sal * total_days)) AS
prorated_employee_expense
    FROM cte1
    GROUP BY title, budget
) AS t
WHERE budget > prorated_employee_expense;
```

| title | budget | prorated_employee_expense |
|---|---|---|
| Project3 | 43909 | 7300 |
| Project5 | 36268 | 24043 |
| Project7 | 34003 | 22305 |
| Project8 | 49284 | 16232 |