**Question\_1:**

The election is conducted in a city and everyone can vote for one or more candidates, or choose not to vote at all. Each person has 1 vote so if they vote for multiple candidates, their vote gets equally split across these candidates. For example, if a person votes for 2 candidates, these candidates receive an equivalent of 0.5 vote each.

Find out who got the most votes and won the election. Output the name of the candidate or multiple names in case of a tie. To avoid issues with a floating-point error you can round the number of votes received by a candidate to 3 decimal places.

**Table\_Details:**

Voting\_details ---> Voter,Candidates

**Solution**

select candidate

from

(select candidate,

round(sum(vote\_value),3) n\_votes,

dense\_rank()over(order by round(sum(vote\_value),3)DESC) place

from

(select candidate,voter, round(1.0 /count(\*) over(partition by voter),3) vote\_value

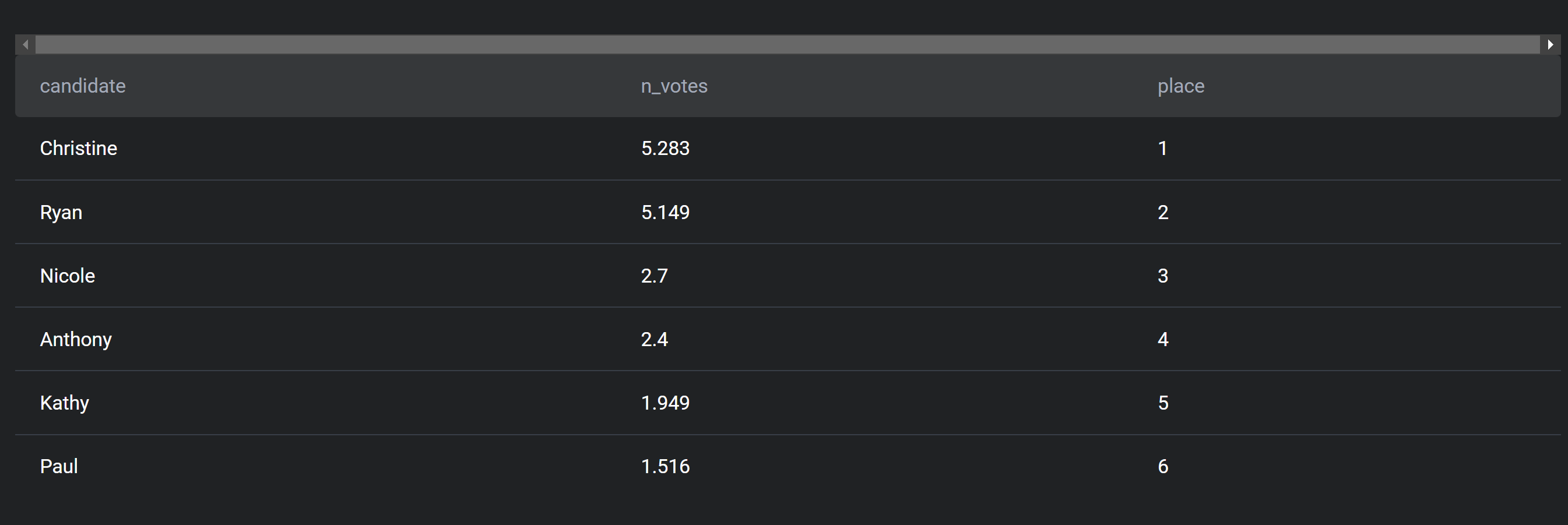
from voting\_results

where candidate is not null) a

group by candidate) results

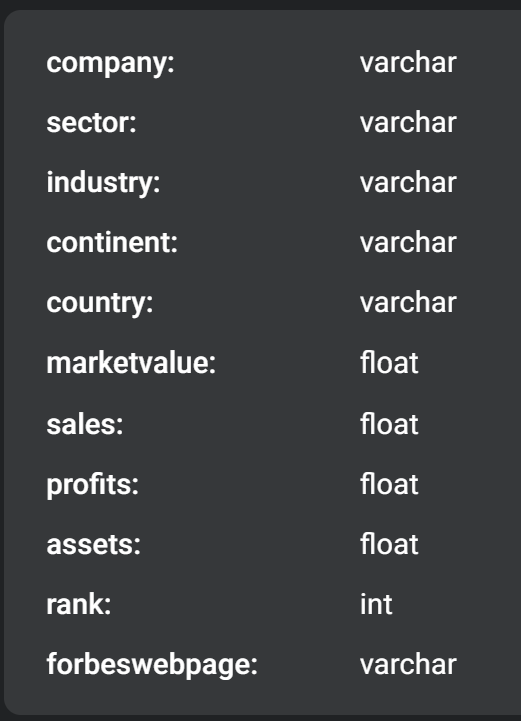
where place = 1

**Solution\_Answer**



**Question\_2:** Find the 3 most profitable companies in the entire world. Output the result along with the corresponding company name. Sort the result based on profits in descending order.

Table\_Details:



**Query\_That\_Solved\_Question**:

select company,profits,`rank`

from forbes\_global\_2010\_2014

order by profits desc;

limit 3;

or

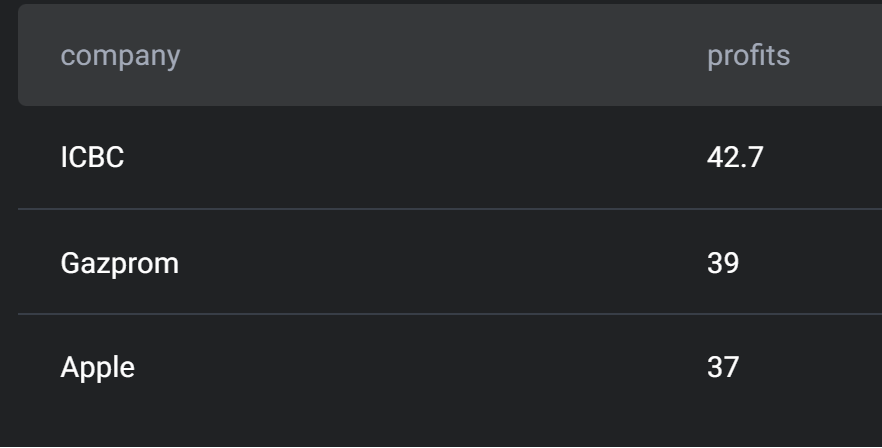
select company,sum(profits) as Total\_profits

from forbes\_global\_2010\_2014

group by company

order by Total\_profits desc;

**Result**:

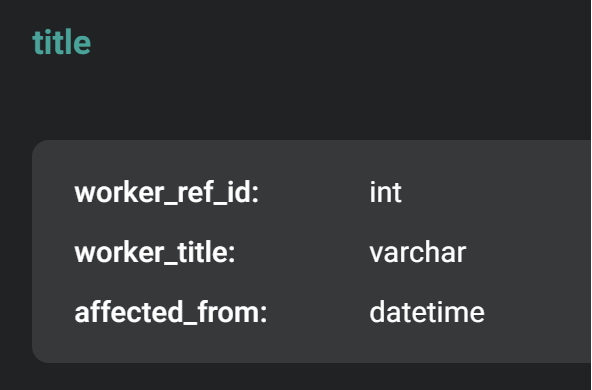
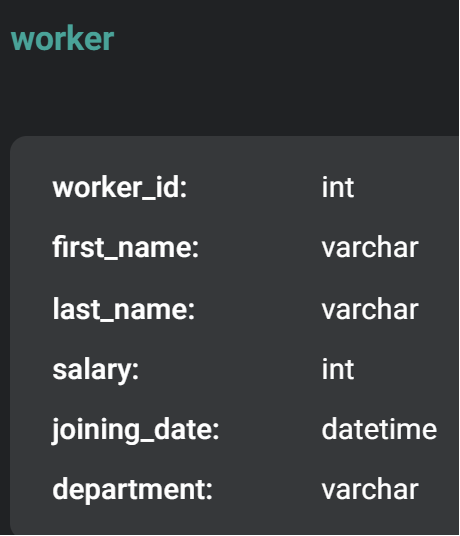


**Question\_3: Worker with highest salaries**

You have been asked to find the job titles of the highest-paid employees.

Your output should include the highest-paid title or multiple titles with the same salary.

**Table\_Details:**



**Query\_That\_Solved\_Question:**

select worker\_id,concat(first\_name," ",last\_name) as name,worker\_title,salary

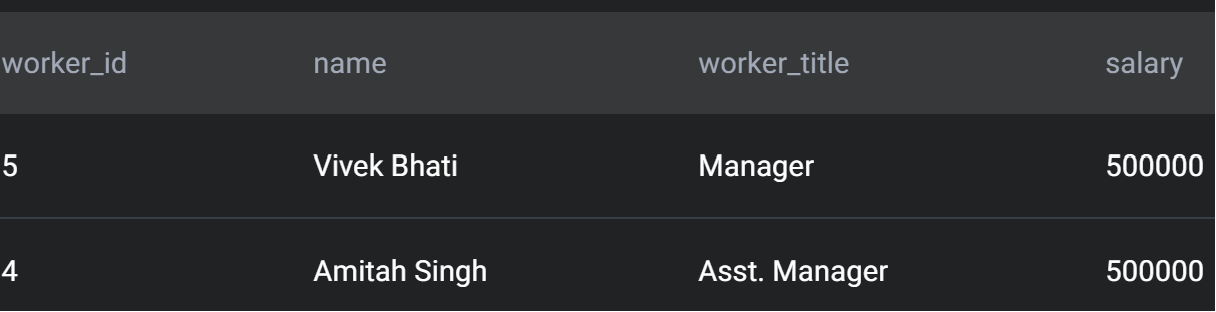
from worker as w

join title as t

on w.worker\_id = t.worker\_ref\_id

where salary = (select max(salary) from worker)

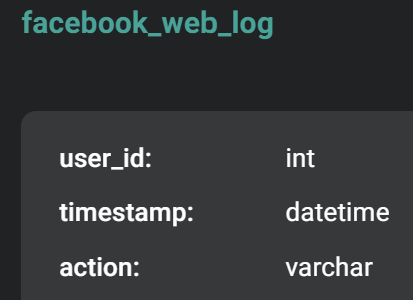
**Result:**

****

**Question\_3: Users by Average Usage Time**

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.

**Table\_Details:**

****

**Solution\_Query:**

select A.user\_id , avg(timestampdiff(second,Late\_load,Early\_exit)) as Average\_session

from

(select user\_id,date(timestamp) as Date, max(timestamp) as Late\_load

from facebook\_web\_log

where action = 'page\_load'

group by 1,2) as A

join

(select user\_id,date(timestamp) as Date, min(timestamp) as Early\_exit

from facebook\_web\_log

where action = 'page\_exit'

group by 1,2) as B

on A.user\_id = B.user\_id

group by 1;

Or

with temp as (

select user\_id, date(timestamp) as date,

max(case when action = 'page\_load' then timestamp else null end)

as late\_load,

min(case when action = 'page\_exit' then timestamp else null end)

as early\_exit

from facebook\_web\_log

group by user\_id, date

)

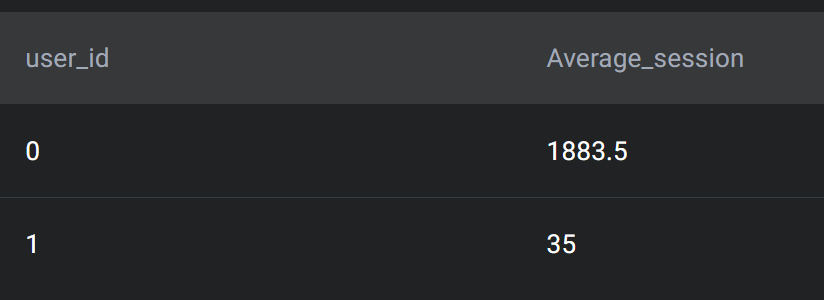
select user\_id, avg(timestampdiff(second,late\_load,early\_exit)) as avg\_time

from temp

where early\_exit is not null and late\_load is not null

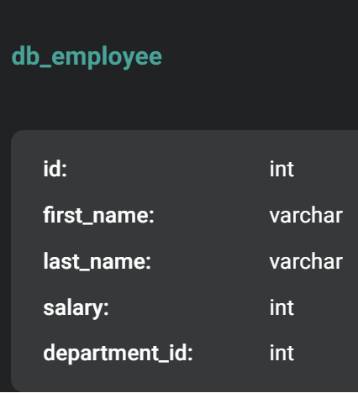
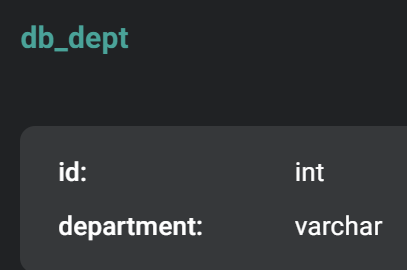
group by user\_id;

**Result:**

****

**Question\_4: Salaries Difference**

Write a Query that calculates that in difference between the highest salaries found in marketing, engineering departments. Output must be absolute difference in salaries

**Tables\_Details:**

**Solution\_Query\_to\_Question:**

with A as

(select max(salary) as marketing\_high

from db\_employee as de

join db\_dept as dd

on de.department\_id = dd.id

where dd.department like "marketing"

group by de.department\_id),

B as

(select max(salary) as engineering\_high

from db\_employee as de

join db\_dept as dd

on de.department\_id = dd.id

where dd.department like "engineering"

group by de.department\_id)

select abs(marketing\_high-engineering\_high) as salary\_difference from A,B;

select

abs(max(case when dd.department = 'marketing' then de.salary

else 0

end ) -

max(case when dd.department = 'engineering' then de.salary

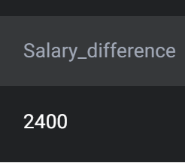
else 0

end)) as Salary\_difference

from db\_employee as de

join db\_dept as dd

on de.department\_id = dd.id

**Result**

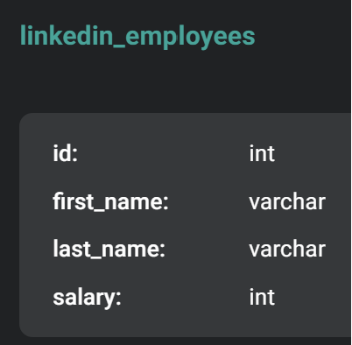
**Question\_5: Risky Projects**

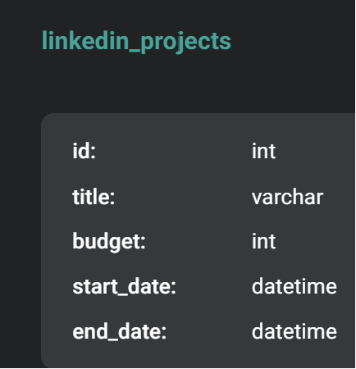
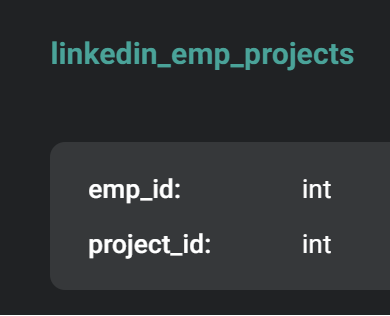
Identify projects that are at risk for going overbudget. A project is considered to be overbudget if the cost of all employees assigned to the project is greater than the budget of the project.

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.

**Table\_Details:**

**Solution\_Query\_to\_question:**

with project\_details as

(select id,title,budget,timestampdiff(day,start\_date,end\_date) as project\_duration

from linkedin\_projects),

employee\_details as

(select id,salary/365 as salary\_per\_day from linkedin\_employees)

select pd.title,budget, ceiling(sum(project\_duration \* salary\_per\_day))

as Total\_cost\_pay\_employee

from project\_details as pd

join linkedin\_emp\_projects as emp

on pd.id = emp.project\_id

join employee\_details as ed

on ed.id = emp.emp\_id

group by pd.id

having budget < Total\_cost\_pay\_employee

order by pd.title ASC

select lp.title,lp.budget,

ceiling((datediff(lp.end\_date,lp.start\_date)\*sum(le.salary)/365)) as prorated\_value

from linkedin\_projects as lp

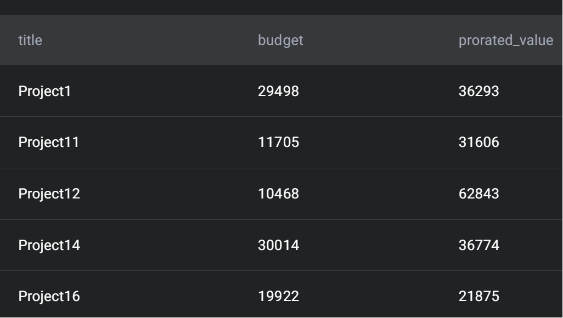
join linkedin\_emp\_projects as lep on lp.id = lep.project\_id

join linkedin\_employees as le on le.id = lep.emp\_id

group by lp.id

having prorated\_value > budget;

**Result:**

**This table shows only 5 rows because of space constraints**