

1. **How to select UNIQUE records from a table using a SQL Query?**

select distinct EMPLOYEE\_ID,NAME,SALARY from employee

select EMPLOYEE\_ID,NAME,SALARY from employee group by EMPLOYEE\_ID

1. **How to delete DUPLICATE records from a table using a SQL Query?**

select \* from employee

select \*,count(\*) from employee group by emp\_id,name,salary having count(\*)>1

delete from employee where emp\_id not in (select min(emp\_id) from employee group by name,salary)

1. **How to read TOP 5 records from a table using a SQL query?**

Consider below table DEPARTMENTS as the source data.



select \* from departments order by Department\_ID asc limit 5

select \* from departments limit 5

1. **How to read LAST 5 records from a table using a SQL query?**

select \* from departments order by Department\_ID desc limit 5

## here last 5 records in ascending order

(select \* from departments order by Department\_ID desc limit 5) order by Department\_ID

1. **How to find the employee with second MAX Salary using a SQL query?**

select max(salary) as second\_max\_salary from employee where salary not in (select max(salary) from employee).

## If we need all the columns too

with

temp as(

select max(salary) as salary from employee where salary not in

(select max(salary) as salary from employee)

)

select a.\* from employee a join temp b on a.salary = b.salary

1. **How to find the employee with third MAX Salary using a SQL query without using Analytic Functions?**

Consider the same EMPLOYEES table as source discussed in previous question

SELECT name, salary FROM Employee e1

WHERE 3-1 =

(SELECT COUNT(DISTINCT salary) FROM Employee e2 WHERE e2.salary > e1.salary)

1. **Assume you have the below tables on sessions that users have, and a users table. Write a query to get the active user count of daily cohorts.**



By definition, daily cohorts are active users from a particular day. First, we can use a subquery to get the sessions of new users by day using an inner join with users. This is to filter for only active users by a particular join date for the cohort. Then we can get a distinct count to return the active user count:

with new\_users\_by\_date as(

select s.\* from sessions s join users u on s.user\_id = u.user\_id s.date = u.date)

select date,count(distinct user\_id) as active\_user\_count from new\_users\_by\_date group by date order by asc

1. **Assume you are given the below table on transactions from users for purchases. Write a query to get the list of customers where their earliest purchase was at least $50.**



we can also use the RANK() window function to get the ordering of purchase by customer, and then use that subquery to filter on customers where the first purchase (rank one) is at least 50 dollars. Note that this requires the subquery to include spend as well.

with purchase\_rank as

( select user\_id,spend,rank() over ( partition by user\_id order by transaction\_date asc)

as rank\_ from user\_transactions)

select user\_id,spend from purchase\_rank where rank =1 and spend>=50

1. **Assume you are given the below table on transactions from users. Write a query to get the number of users and total products bought per latest transaction date where each user is bucketed into their latest transaction date.**



First, we need to get the latest transaction date for each user, along with the number of products they have purchased. This can be done in a subquery where we GROUP BY user\_id and take a COUNT(DISTINCT product\_id) to get the number of products they have purchased, and a MAX(transaction\_date) to get the latest transaction date (while casting to a date). Then, using this subquery, we can simply do an aggregation by the transaction date column in the previous subquery, while doing a COUNT() on the number of users, and a SUM() on the number of products:

with latest\_transaction as(

select user\_id, count(distinct product\_id) as product\_num,

max(transaction\_date::Date) as transactions from user\_transaction

group by user\_id)

select user\_transaction, count(user\_id) as num\_users,

sum(num\_products) as total\_products from latest\_transaction group by 1

1. **Assume you are given the below tables on users and their time spent on sending and opening Snaps. Write a query to get the breakdown for each age breakdown of the percentage of time spent on sending versus opening snaps.**

### activities Table:

| **Column Name** | **Type** |
| --- | --- |
| activity\_id | integer |
| user\_id | integer |
| activity\_type | string ('send', 'open', 'chat') |
| time\_spent | float |
| activity\_date | Datetime |

### age\_breakdown Table:

| **Column Name** | **Type** |
| --- | --- |
| user\_id | integer |
| age\_bucket | string ('21-25', '26-30', '31-25') |

We can get the breakdown of total time spent on each activity by each user by filtering out for the activity\_type and taking the sum of time spent. In doing this, we want to do an outer join with the age bucket to get the total time by age bucket for both activity types. This results in the below two subqueries. Then, we can use these two subqueries to sum them by joining on the appropriate age bucket and take the proportion for send time and the proportion for open time per age bucket :

## send activity\_type total time spent on each activity by each user

with send\_timespent as(

select age\_breakdown.age\_bucket, sum(activities.time\_spent) from age\_breakdown

left join on age\_breakdown.user\_id = activities.user\_id

where activity\_type = 'send'

group by 1)

## open activity\_type total time spent on each activity by each user

open open\_timespent as(

select age\_breakdown.age\_bucket, sum(activities.time\_spent) from age\_breakdown

left join on age\_breakdown.user\_id = activities.user\_id

where activity\_type = 'open'

group by 1)

## finally for percentage of time spent on sending versus opening snaps.

select a.age\_bucket,

s.send\_timespent /(s.send\_timespent + o.open\_timespent) as pct\_send,

o.open\_timespent / (s.send\_timespent + o.open\_timespent) as pct\_open,

from age\_breakdown a

left join send\_timespent s on a.age\_bucket = s.age\_bucket

left join open\_timespent o on a.age\_bucket = o.age\_bucket

group by 1

1. **Assume you are given the below table on reviews from users. Define a top-rated place as a business whose reviews only consist of 4 or 5 stars. Write a query to get the number and percentage of businesses that are top-rated places.**



First, we need to get the places where the reviews are all 4 or 5 stars. We can do this using a HAVING clause, instead of a WHERE clause since the reviews need to all be 4 stars or above. For the HAVING condition, we can use a CASE statement that filters for 4 or 5 stars and then take a SUM over them. This can then be compared with the total row count of the particular business\_id reviews to ensure that the count of top reviews matches with the total review count. With the relevant businesses, we can then do an outer join with the original table on business\_id to get a COUNT of distinct business\_id matches, and then the percentage by comparing the COUNT from the top places with the overall COUNT of business\_id :-

with top\_places as( select business\_id from reviews group by 1 having

sum(

case

when review\_stars >=4 then 1 else 0 end) = count(\*)

)

select count(distinct t.business\_id) as top\_places,

count(distinct t.business\_id/count(r.business\_id) as top\_places\_pct

from reviews r

left join top\_places t

on r.business\_id = t.business\_id)

**12.**







**13.** 



