**SQL Challenge**

CREATE DATABASE challenge;

use challenge;

CREATE TABLE CITY

( ID INT,

NAME VARCHAR(17),

COUNTRYCODE VARCHAR(3),

DISTRICT VARCHAR(20),

POPULATION INT);

INSERT INTO CITY VALUES(6 ,'Rotterdam', 'NLD' ,'Zuid-Holland' ,593321),

(3878 ,'Scottsdale', 'USA' ,'Arizona' ,202705),

(3965 ,'Corona' ,'USA' ,'California', 124966),

(3973 ,'Concord', 'USA', 'California', 121780),

(3977, 'Cedar Rapids' ,'USA' ,'Iowa' ,120758),

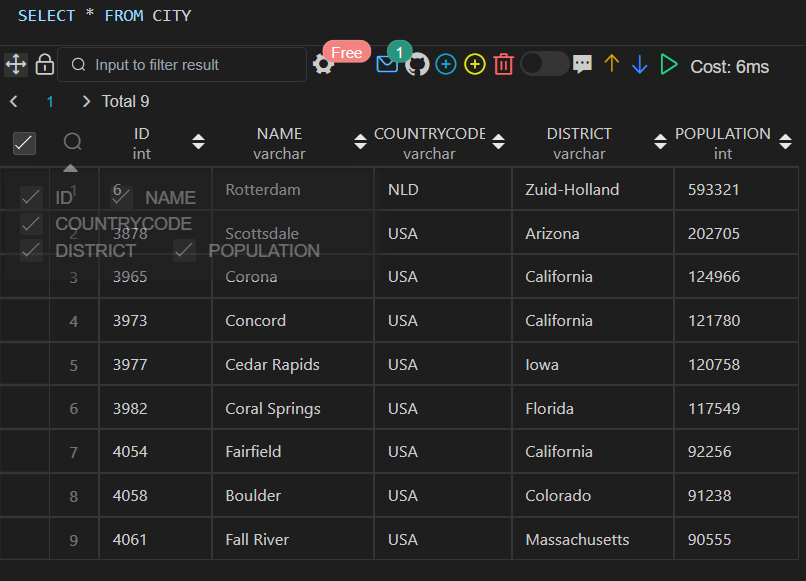
(3982 , 'Coral Springs', 'USA' ,'Florida' ,117549),

(4054 ,'Fairfield', 'USA' ,'California', 92256),

(4058 ,'Boulder' ,'USA', 'Colorado' ,91238),

(4061 ,'Fall River', 'USA', 'Massachusetts', 90555);

SELECT \* FROM CITY;



Q1. Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

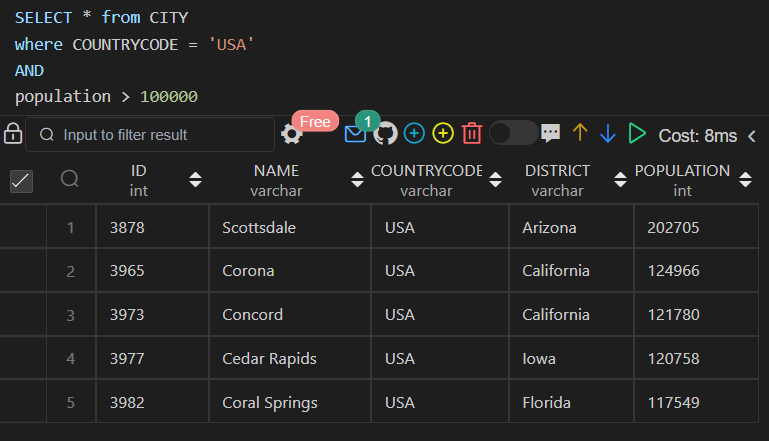
Solution:

SELECT \* from CITY

where COUNTRYCODE = 'USA'

AND

population > 100000 ;



Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

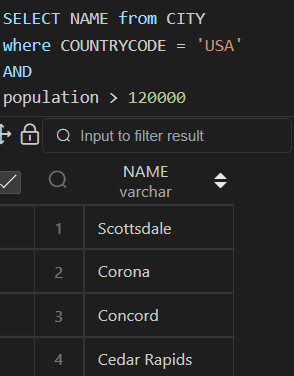
Solution:

SELECT NAME from CITY

where COUNTRYCODE = 'USA'

AND

population > 120000 ;



Q3. Query all columns (attributes) for every row in the CITY table.

Solution:

SELECT \* FROM CITY;

Chart, funnel chart

Description automatically generated

Q4. Query all columns for a city in CITY with the ID 1661.

Solution:

SELECT \* FROM CITY

WHERE ID = 1661

Graphical user interface, text, application, chat or text message

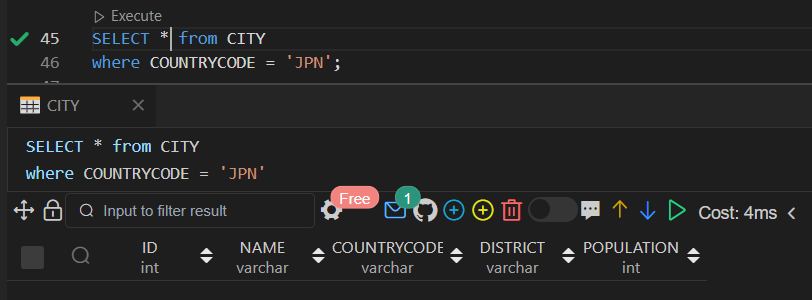
Description automatically generated

Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

Solution:

SELECT \* from CITY

where COUNTRYCODE = 'JPN’;



Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

Solution:

SELECT NAME from CITY

where COUNTRYCODE = 'JPN’;

A screenshot of a computer

Description automatically generated with medium confidence

Dataset 2

CREATE Table Station

(

ID INT,

city VARCHAR(21),

state VARCHAR(2),

LAT\_N INT,

LONG\_W INT

);

INSERT INTO Station VALUES

(794,'Kissee Mills','MO',139,73),

(824,'Loma Mar','CA',48,130),

(603,'Sandy Hook','CT',72,148),

(478,'Tipton','IN',33,97),

(619,'Arlington','CO',75,92),

(711,'Turner','AR',50,101),

(839,'Slidell','LA',85,151),

(411,'Negreet','LA',98,105),

(588,'Glencoe','KY',46,136),

(665,'Chelsea','IA',98,59),

(342,'Chignik Lagoon','AK',103,153),

(733,'Pelahatchie','MS',38,28),

(441,'Hanna City','IL',50,136),

(811,'Dorrance','KS',102,121),

(698,'Albany','CA',49,80),

(325,'Monument','KS',70,141),

(414,'Manchester','MD',73,37),

(113,'Prescott','IA',39,65),

(971,'Graettinger','IA',94,150),

(266,'Cahone','CO',116,127);

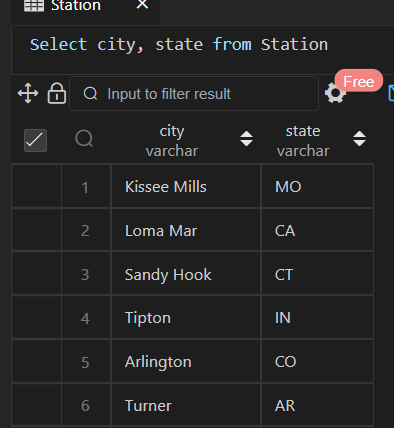
Select \* from Station;



Q7. Query a list of CITY and STATE from the STATION table.

Solution:

Select city, state from Station;



Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

Solution:

Select Distinct city

from Station

WHERE ID %2 = 0;

Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

Solution:

Select count(city) - count(Distinct city)

from Station;

Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

Solution:

select \* from (SELECT city ,LENGTH(City) as city\_length

from Station

order by LENGTH(City) desc ,city asc limit 1) as longest

UNION

select \* from (SELECT city , LENGTH(city) as city\_length

from Station

order by LENGTH(City) asc, city asc limit 1) as shortest;

Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '^[aeiouAEIOU].\*');

Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '[aeiouAEIOU]$');

Q13. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '^[^aeiouAEIOU].\*');

Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '[^aeiouAEIOU]$');

Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '^[^aeiouAEIOU].\*|[^aeiouAEIOU]$');

Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

Solution:

Select distinct city from Station

Where RegexP\_like(city, '^[^aeiouAEIOU].\*')

AND RegexP\_like(city, '[^aeiouAEIOU]$');

Q17. Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive. Return the result table in any order.

Solution:

Select product\_id, product\_name from Product

where product\_id not in

(select p.product\_id from Product p join Sales s

on p.product\_id = s.product\_id

WHERE sale\_date NOT between '2019-01-01' and '2019-03-31' )

Q18. Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order.

Solution:

select distinct a.author\_id from Views a join Views v

on a.author\_id = v.viewer\_id

order by author\_id asc;

Q19. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

Solution:

select

round((SUM(order\_date = customer\_pref\_delivery\_date)\*100/count(order\_date)),2) as immediate\_percentage

from Delivery;

Q20. Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points. Return the result table ordered by ctr in descending order and by ad\_id in ascending order in case of a tie.

Solution:

select ad\_id,

case

when SUM(action = "Clicked" or action = "Viewed") = 0 then 0

else round((SUM(action = "Clicked")\*100/SUM(action = "Clicked" or action = "Viewed")),2)

end as ctr

from Ads

Group by ad\_id

ORDER BY ctr desc, ad\_id asc;

Q21. Write an SQL query to find the team size of each of the employees. Return result table in any order.

Solution:

select employee\_id,

count(\*) over (Partition by team\_id) as team\_size

from Employee

order by employee\_id;

Q22. Write an SQL query to find the type of weather in each country for November 2019. The type of weather is:

* Cold if the average weather\_state is less than or equal 15,
* Hot if the average weather\_state is greater than or equal to 25, and
* Warm otherwise. Return result table in any order.

Solution:

Select distinct country\_name, W.weather\_type from Countries C

join

(select country\_id,

case when AVG(weather\_state) over (partition by country\_id) <= 15 then "Cold"

when AVG(weather\_state) over (partition by country\_id) >= 25 then "Hot"

else "Warm"

end as weather\_type

from Weather

where Year(day) = "2019" and Month(day) = "11") W

on C.country\_id = W.country\_id

order by W.weather\_type;

Q23. Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places. Return the result table in any order.

Solution:

Select product\_id,

round((Sum(price)/SUM(units)),2) as average\_price From

(SELECT P.product\_id,U.units\*P.price as price, U.units as units

From Prices P Join UnitsSold U

on P.product\_id = U.product\_id

where U.purchase\_date BETWEEN P.start\_date and P.end\_date) PU

group by product\_id;

Q25. Write an SQL query to report the device that is first logged in for each player. Return the result table in any order.

Solution:

select player\_id, device\_id from

(select \*,

row\_number() over(partition by player\_id order by event\_date) as logged\_rank

from Activity) A

where A.logged\_rank =1;

Q26. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.

Solution:

Select distinct product\_name, unit

FROM

(Select p.\*

, SUM(o.unit) over(partition by o.product\_id) as unit

 from

 Products p join Orders o

 on p.product\_id = o.product\_id

 Where

  Year(o.order\_date) ="2020" and Month(o.order\_date)='02') OP

WHERE unit >=100

;

Q27. Write an SQL query to find the users who have valid emails. A valid e-mail has a prefix name and a domain where:

* The prefix name is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'. The prefix name must start with a letter.
* The domain is '@leetcode.com'.

Solution:

Select \* from Users

where  REGEXP\_LIKE(mail, '^[A-Za-z]+[A-Za-z\_.-]')

AND mail LIKE '%@leetcode.com';

Q28. Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020. Return the result table in any order.

Solution:

select o.customer\_id, c.name

from Customers c, Products p, Orders o

where c.customer\_id = o.customer\_id and p.product\_id = o.product\_id

group by o.customer\_id

having

(

    sum(case when o.order\_date like '2020-06%' then o.quantity\*p.price else 0 end) >= 100

    and

    sum(case when o.order\_date like '2020-07%' then o.quantity\*p.price else 0 end) >= 100

)

;

Q29. Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020. Return the result table in any order.

Solution:

Select c.title from Content c join TVProgram t

on c.content\_id = t.content\_id

where

c.Kids\_content = 'Y'

and c.content\_type = 'Movies'

and t.program\_date like '2020-06%';

Q30. Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order.

Solution:

Select distinct q.\* , IFNULL(n.npv,0)

FROM Queries AS q

LEFT JOIN NPV as n ON q.id = n.id AND q.year = n.year

;

Q31. Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order.

Solution:

Select distinct q.\* , IFNULL(n.npv,0)

FROM Queries AS q

LEFT JOIN NPV as n ON q.id = n.id AND q.year = n.year

;

Q32. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null. Return the result table in any order.

Solution: SELECT eu.unique\_id, e.name

from Employee e left join EmployeeUNI eu

on e.id = eu.id

order by e.name;

Q33. Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.

Solution:

SELECT u.name, IFNULL(SUM(r.distance) ,0) as travelled\_distance

From Users u left join Rides r on u.id = r.user\_id

Group by (u.name)

order by SUM(r.distance) DESC, u.name ASC;

Q34. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount. Return result table in any order.

Solution:

Table Schema is not shared correctly.

Q35. Write an SQL query to:

* Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
* Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.

Solution:

select name from (select u.name ,

count(movie\_id) over(partition by mr.user\_id) as results

from MovieRating mr join Users u

on mr.user\_id = u.user\_id

order by results desc, u.name asc

limit 1) first\_results

UNION

select title from (select m.title ,

AVG(rating) over(partition by mr.movie\_id) as results

from MovieRating mr join Movies m

on mr.movie\_id = m.movie\_id

where created\_at like '2020-02%'

order by results desc, m.title asc

limit 1) second\_results

;

Q36. Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.

Solution:

SELECT u.name, IFNULL(SUM(r.distance) ,0) as travelled\_distance

From Users u left join Rides r on u.id = r.user\_id

Group by (u.name)

order by SUM(r.distance) DESC, u.name ASC;

Q37. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null. Return the result table in any order

Solution:

SELECT eu.unique\_id, e.name

from Employee e left join EmployeeUNI eu

on e.id = eu.id

order by e.name;

Q38. Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist. Return the result table in any order.

Solution:

Select id, name from Students

where

department\_id not in (select distinct id from Departments)

order by id;

Q39. Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1 < person2. Return the result table in any order.

Solution:

 select distinct person1, person2,

 count(\*) over(partition by person1, person2) as call\_count,

 SUM(duration) over(partition by person1, person2) as total\_duration

 from (select from\_id as person1, to\_id as person2, duration

    from calls

    UNION ALL

    select to\_id as person1, from\_id as person2, duration

    from calls) dd

    where person1 < person2;

Q40. Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places. Return the result table in any order.

Solution:

Select product\_id,

round((Sum(price)/SUM(units)),2) From

(SELECT P.product\_id,U.units\*P.price as price, U.units as units

From Prices P Join UnitsSold U

on P.product\_id = U.product\_id

where U.purchase\_date BETWEEN P.start\_date and P.end\_date) PU

group by product\_id;

Q41. Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse. Return the result table in any order.

Solution:

Select distinct w.name,

SUM(w.units\*p.vol) over(partition by w.name)

from

(Select \*,Width\* Length \*Height as vol

from Products) p join Warehouse w

on w.product\_id = p.product\_id

;

Q42. Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale\_date.

Solution:

select sd.sale\_date,

(Apples-Oranges) as diff

from(select sale\_date,

SUM(case when fruit = 'apples' then sold\_num end) as Apples,

SUM(case when fruit = 'oranges' then sold\_num end) as Oranges

 from Sales

 group by sale\_date) sd;

Q43. Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.

Solution:

SELECT

round((count(distinct c.player\_id) / (select count(distinct player\_id) from Activity)),2)as fraction

FROM

(

SELECT

player\_id, min(event\_date) as event\_start\_date

from

Activity

group by player\_id ) c

JOIN Activity a

on c.player\_id = a.player\_id

and datediff(c.event\_start\_date, a.event\_date) = -1

;

Q44. Write an SQL query to report the managers with at least five direct reports. Return the result table in any order.

Solution:

select name

FROM Employee e

where id in(select managerId

from Employee

group by managerId

having  count(id) >= 5);

Q45. Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student\_number in descending order. In case of a tie, order them by dept\_name alphabetically.

Solution:

select distinct d.dept\_name,

Count(s.student\_id) over(partition by d.dept\_id) as student\_number

from Department d left join Students s on d.dept\_id = s.dept\_id;

Q46. Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table. Return the result table in any order.

Solution: Select customer\_id from Customer

GROUP by customer\_id

having

count(distinct product\_key) =

(select count(product\_key) from Product);

Q47. Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years. Return the result table in any order.

Solution:

with

cte as (select p.\*, e.experience\_years,

rank() over(partition by project\_id order by experience\_years desc) as rank\_exp

FROM Project

 p join Employee e on

p.employee\_id = e.employee\_id

order by project\_id)

select project\_id,employee\_id

from cte where rank\_exp =1 ;