**SQL challenge**

**Rohini reddy**

**21wu0101047**

**Team3**

**Q1.)** Query all columns for all American cities in the CITY table with populations larger than 100000.

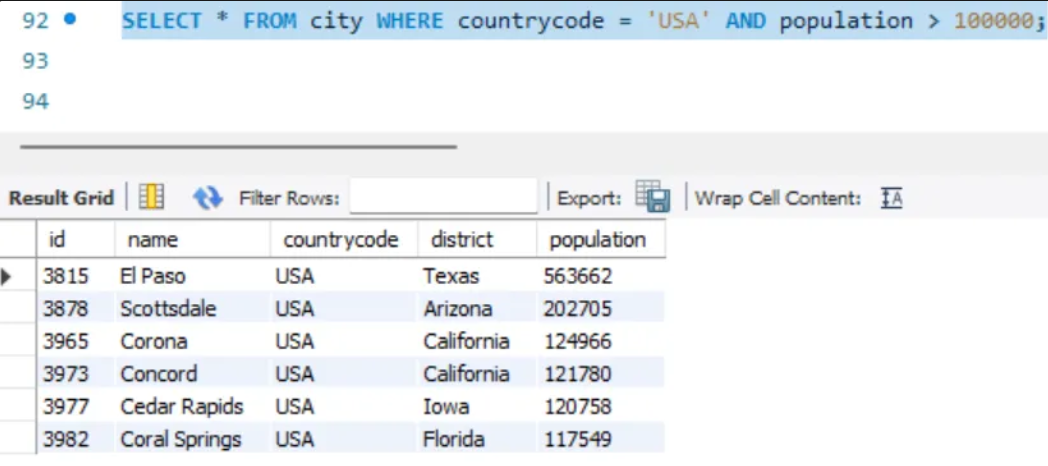
The CountryCode for America is USA.

SELECT \*

FROM CITY

WHERE CountryCode = 'USA' AND Population > 100000;

Output:



**Q2.)** Query the NAME field for all American cities in the CITY table with populations larger than 120000.

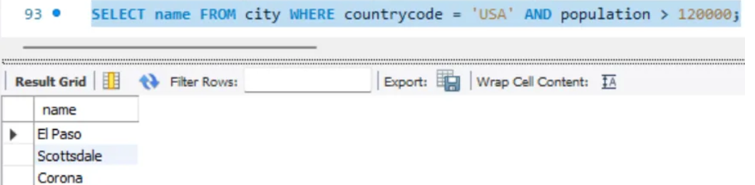
The CountryCode for America is USA.

SELECT NAME

FROM CITY

WHERE CountryCode = 'USA' AND Population > 120000;

Output:



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

SELECT \*

FROM CITY;

Output:

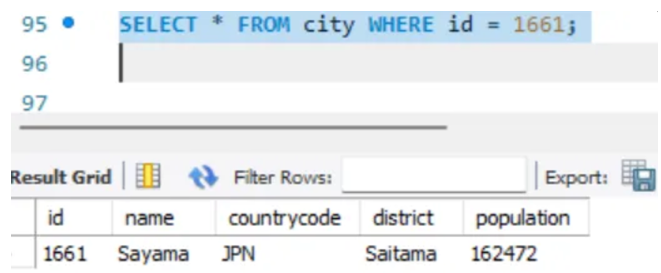


**Q4.)** Query all columns for a city in CITY with the ID 1661.

SELECT \*

FROM CITY

WHERE ID = 1661;

Output: 

**Q5.)** Query all attributes of every Japanese city in the CITY table.

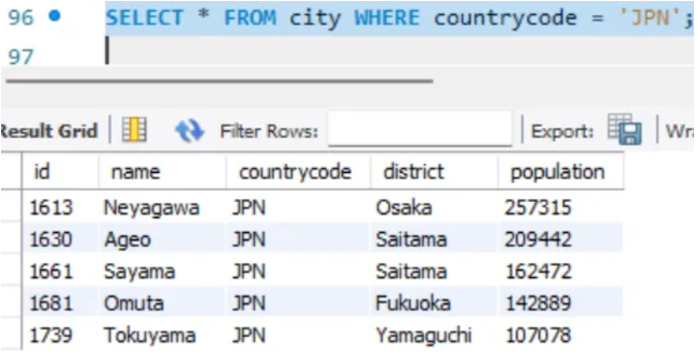
The CountryCode for Japan is JPN.

SELECT \*

FROM CITY

WHERE CountryCode = 'JPN';

Output:



**Q6.)** Query the NAME field of all the Japanese cities in the CITY table.

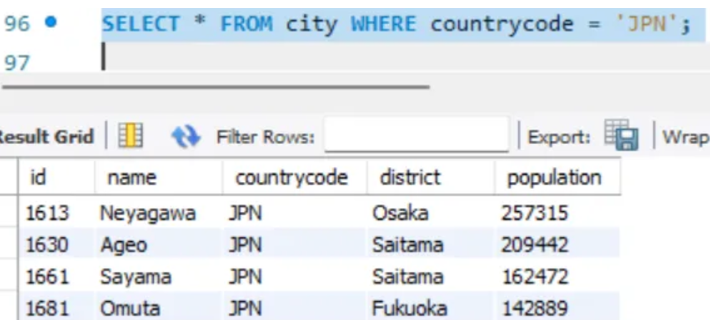
The CountryCode for Japan is JPN.

SELECT NAME

FROM CITY

WHERE CountryCode = 'JPN';

Output:

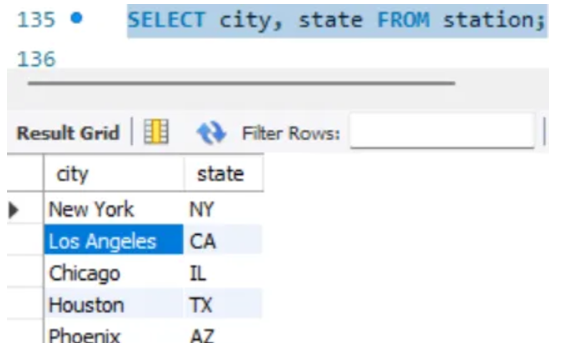


**Q7.)** Query a list of CITY and STATE from the STATION table.

SELECT CITY, STATE

FROM STATION;

Output:



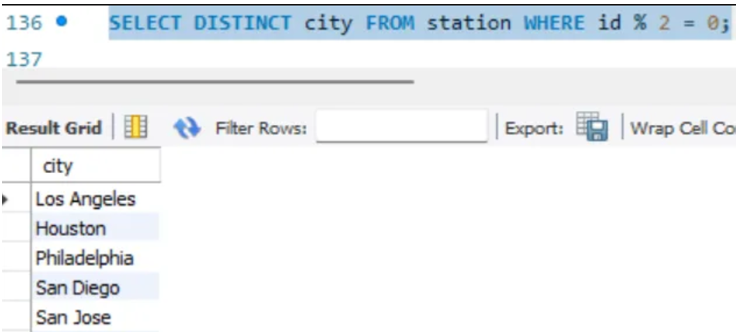
**Q8.)** Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

SELECT DISTINCT CITY

FROM STATION

WHERE MOD(ID, 2) = 0;

Output:



**Q9.)**

SELECT COUNT(CITY) - COUNT(DISTINCT CITY) AS Difference

FROM STATION;

Output:

A screenshot of a computer

Description automatically generated

**Q10.)**

Query for the shortest CITY name:

SELECT CITY, LENGTH(CITY) AS Length

FROM STATION

ORDER BY LENGTH(CITY), CITY

LIMIT 1;

Query for the longest CITY name:

SELECT CITY, LENGTH(CITY) AS Length

FROM STATION

ORDER BY LENGTH(CITY) DESC, CITY

LIMIT 1;

Output:

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

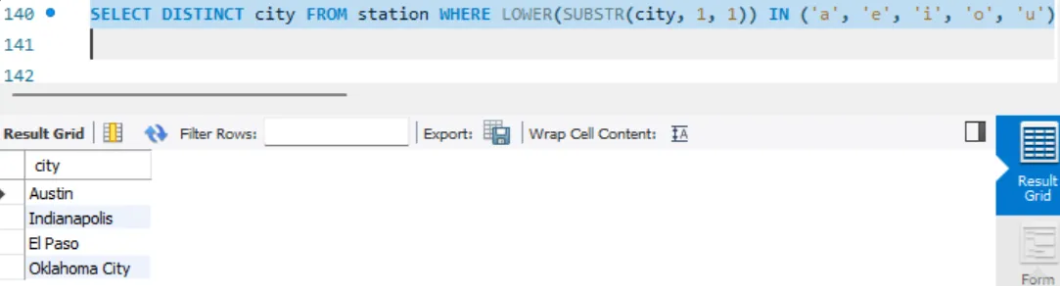
**Q11.)** Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates

SELECT DISTINCT CITY

FROM STATION

WHERE CITY REGEXP '^[AEIOUaeiou]';

Output:



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

SELECT DISTINCT CITY

FROM STATION

WHERE CITY REGEXP '[AEIOUaeiou]$';

Output:

A screenshot of a computer

Description automatically generated

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

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '^[AEIOUaeiou]';

Output:

A screenshot of a computer

Description automatically generated

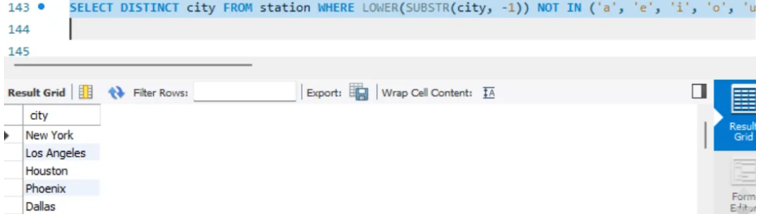
**Q14.)** Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '[AEIOUaeiou]$';

Output:



**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.

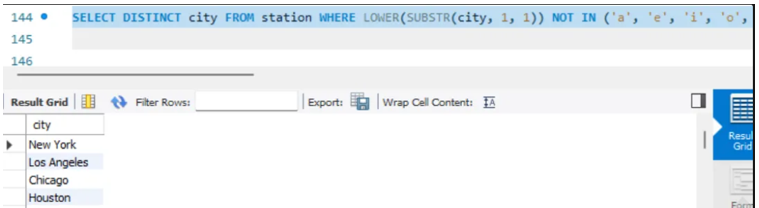
SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '^[AEIOUaeiou]'

OR CITY NOT REGEXP '[AEIOUaeiou]$';

Output:



**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.

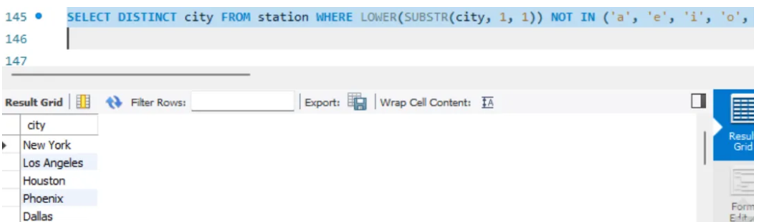
SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '^[AEIOUaeiou]'

AND CITY NOT REGEXP '[AEIOUaeiou]$';

Output:



**Q17.)** WITH SalesInQ1 AS (

SELECT DISTINCT product\_id

FROM Sales

WHERE sale\_date BETWEEN '2019-01-01' AND '2019-03-31'

),

SalesOutsideQ1 AS (

SELECT DISTINCT product\_id

FROM Sales

WHERE sale\_date NOT BETWEEN '2019-01-01' AND '2019-03-31'

)

SELECT p.product\_id, p.product\_name

FROM Product p

JOIN SalesInQ1 si ON p.product\_id = si.product\_id

WHERE p.product\_id NOT IN (SELECT product\_id FROM SalesOutsideQ1);

Output:

A screenshot of a computer

Description automatically generated

**Q18.)** SELECT DISTINCT author\_id AS id

FROM Views

WHERE author\_id = viewer\_id

ORDER BY id ASC;

Output:

**A screenshot of a computer

Description automatically generated**

**Q19.)** SELECT

ROUND(

(SUM(CASE WHEN order\_date = customer\_pref\_delivery\_date THEN 1 ELSE 0 END) \* 100.0) / COUNT(\*),

2

) AS immediate\_percentage

FROM

Delivery;

Output:

A screenshot of a computer

Description automatically generated

**Q20.)** SELECT

ad\_id,

ROUND(

CASE

WHEN SUM(CASE WHEN action IN ('Clicked', 'Viewed') THEN 1 ELSE 0 END) = 0 THEN 0

ELSE (SUM(CASE WHEN action = 'Clicked' THEN 1 ELSE 0 END) \* 100.0) /

SUM(CASE WHEN action IN ('Clicked', 'Viewed') THEN 1 ELSE 0 END)

END,

2

) AS ctr

FROM

Ads

WHERE

action IN ('Clicked', 'Viewed') -- Exclude 'Ignored' actions

GROUP BY

ad\_id

ORDER BY

ctr DESC,

ad\_id ASC;

Output:

A screenshot of a computer

Description automatically generated

**Q21.)** SELECT

e.employee\_id,

COUNT(t.employee\_id) AS team\_size

FROM

Employee e

JOIN

Employee t

ON

e.team\_id = t.team\_id

GROUP BY

e.employee\_id;

Output: The query returns a table with employee\_id and their respective team\_size.

A screenshot of a computer

Description automatically generated

**Q22.)** SELECT

c.country\_name,

CASE

WHEN AVG(w.weather\_state) <= 15 THEN 'Cold'

WHEN AVG(w.weather\_state) >= 25 THEN 'Hot'

ELSE 'Warm'

END AS weather\_type

FROM

Countries c

JOIN

Weather w

ON

c.country\_id = w.country\_id

WHERE

w.day BETWEEN '2019-11-01' AND '2019-11-30'

GROUP BY

c.country\_id, c.country\_name;

Output: The query returns a table with country\_name and the corresponding weather\_type, as shown in the example. Countries without any weather data in November 2019 are excluded from the results.

A screenshot of a computer program

Description automatically generated

**Q23.)** SELECT

p.product\_id,

ROUND(SUM(p.price \* u.units) / SUM(u.units), 2) AS average\_price

FROM

Prices p

JOIN

UnitsSold u

ON

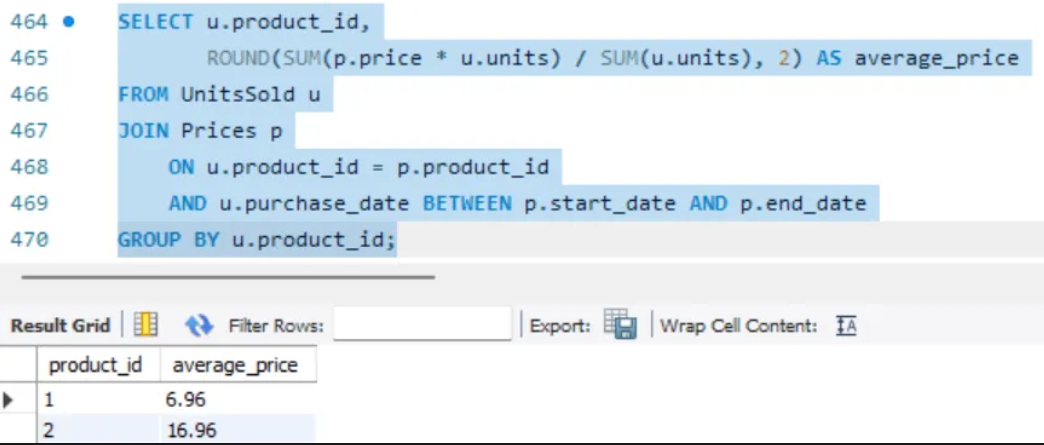
p.product\_id = u.product\_id

AND u.purchase\_date BETWEEN p.start\_date AND p.end\_date

GROUP BY

p.product\_id;

Output:



**Q24.)** SELECT

player\_id,

MIN(event\_date) AS first\_login

FROM

Activity

GROUP BY

player\_id;

Output:

**A screenshot of a computer

Description automatically generated**

**Q25.)** SELECT

player\_id,

device\_id

FROM

Activity

WHERE

(player\_id, event\_date) IN (

SELECT

player\_id,

MIN(event\_date) AS first\_login

FROM

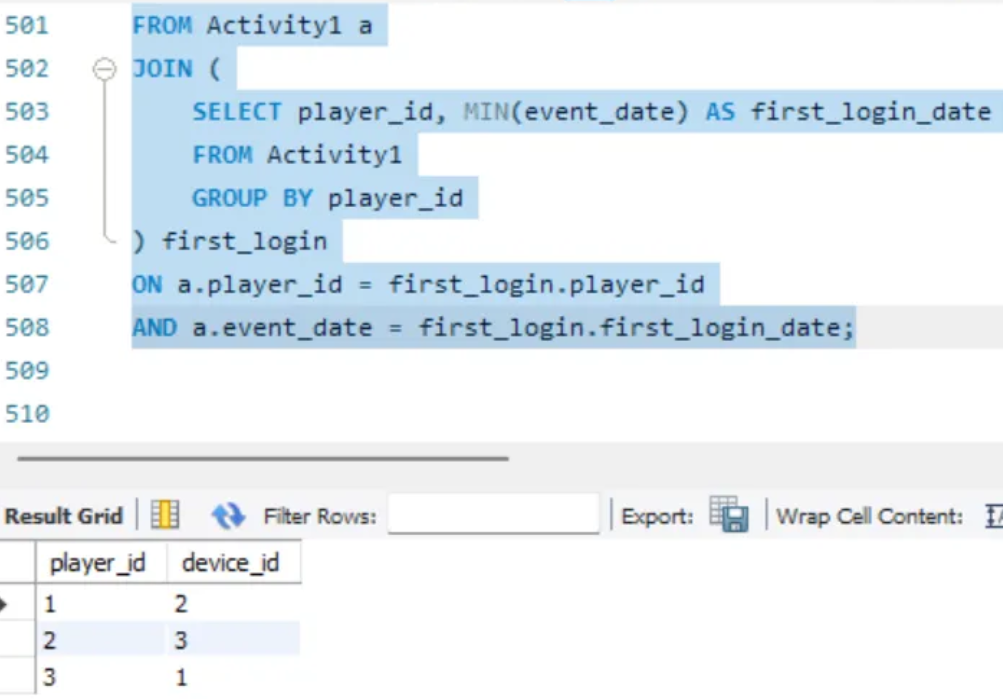
Activity

GROUP BY

player\_id

);

Output: This query will return a table with player\_id and the device\_id that the player first used to log in, as shown in the example.

****

**Q26.)** SELECT p.product\_name, SUM(o.unit) AS amount FROM Products p

JOIN Orders o ON p.product\_id = o.product\_id

WHERE o.order\_date BETWEEN '2020-02-01' AND '2020-02-29'

GROUP BY p.product\_name

HAVING SUM(o.unit) >= 100;

Output:

A screenshot of a computer

Description automatically generated