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

**The CountryCode for America is USA.**

Answer:

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.**

Answer:

SELECT NAME FROM CITY WHERE CountryCode = ‘USA’ AND Population >120000;

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

Answer:

SELECT \* FROM CITY;

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

Answer:

SELECT \* FROM CITY WHERE ID = 1661;

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

**JPN.**

Answer:

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.**

Answer:

SELECT NAME FROM CITY WHERE CountryCode = 'JPN';

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

**The STATION table is described as follows:**

Answer:

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.**

Answer:

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.**

Answer:

SELECT COUNT(\*) - COUNT(DISTINCT CITY) AS Difference FROM CITY;

**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.**

Answer:

SELECT CITY, LENGTH(CITY) AS NAME\_LENGTH FROM STATION

ORDER BY NAME\_LENGTH, CITY

LIMIT 1

UNION ALL

SELECT CITY, LENGTH(CITY) AS NAME\_LENGTH FROM STATION

ORDER BY NAME\_LENGTH DESC, CITY

LIMIT 1;

**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.**

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE CITY LIKE 'A%' OR CITY LIKE 'E%' OR CITY LIKE 'I%' OR CITY LIKE 'O%' OR CITY LIKE 'U%';

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

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE RIGHT(CITY, 1) IN ('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U');

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

**contain duplicates.**

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE CITY NOT REGEXP '^[aeiouAEIOU]';

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

**contain duplicates.**

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE RIGHT(CITY, 1) NOT IN ('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U');

**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.**

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE CITY NOT REGEXP '^[aeiouAEIOU]' OR CITY NOT REGEXP '[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.**

Answer:

SELECT DISTINCT CITY FROM STATION

WHERE CITY NOT REGEXP '^[aeiouAEIOU]' AND CITY NOT REGEXP '[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.**

Answer:

SELECT DISTINCT product\_id FROM Sales

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

GROUP BY product\_id

HAVING COUNT(DISTINCT sale\_date) = DATEDIFF('2019-03-31', '2019-01-01') + 1;

**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.**

Answer:

SELECT DISTINCT A.id, A.name

FROM Authors A

JOIN Articles AR ON A.id = AR.author\_id

JOIN Views V ON AR.id = V.article\_id AND A.id = V.viewer\_id

ORDER BY A.id ASC;

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

Answer:

SELECT ROUND(

(SUM(CASE WHEN order\_type = 'immediate' THEN 1 ELSE 0 END) / COUNT(\*) \* 100),

2

) AS percentage\_immediate

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.**

Answer:

SELECT ad\_id,

ROUND(SUM(CASE WHEN action = 'Clicked' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) 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.**

Answer:

SELECT employee\_id,

COUNT(employee\_id) OVER(PARTITION BY team) AS team\_size

FROM Employees;

**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.**

Answer:

SELECT country,

CASE

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

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

ELSE 'Warm'

END AS weather\_type

FROM Weather

WHERE YEAR(date) = 2019 AND MONTH(date) = 11

GROUP BY country;

**Q23. Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places.**

Answer:

SELECT product\_id,

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

FROM Sales

GROUP BY product\_id;

**Q24. Write an SQL query to report the first login date for each player. Return the result table in any order.**

Answer:

SELECT player\_id, MIN(login\_date) AS first\_login\_date

FROM Activity

GROUP BY player\_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.**

Answer:

SELECT player\_id,

FIRST\_VALUE(device) OVER(PARTITION BY player\_id ORDER BY login\_date) AS first\_logged\_device

FROM Activity;

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

Answer:

SELECT p.product\_name,

SUM(o.unit \* p.price) AS amount

FROM Orders o

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

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

GROUP BY p.product\_name

HAVING SUM(o.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**](http://@leetcode.com)**'.**

Answer:

SELECT email

FROM Users

WHERE email REGEXP '^[a-zA-Z][a-zA-Z0-9.\_-]\*@leetcode\.com$';

**Q28. Write an SQL query to report the customer\_id and customer\_name of customers who have spent atleast $100 in each month of June and July 2020.**

Answer:

SELECT c.customer\_id, c.customer\_name

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

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

WHERE o.order\_date BETWEEN '2020-06-01' AND '2020-07-31'

GROUP BY c.customer\_id, c.customer\_name, YEAR(o.order\_date), MONTH(o.order\_date)

HAVING SUM(o.quantity \* p.price) >= 100

AND COUNT(DISTINCT YEAR(o.order\_date), MONTH(o.order\_date)) = 2;

**Q29. Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.**

Answer:

SELECT DISTINCT title

FROM Content

WHERE Kids\_content = 'Y'

AND content\_type = 'movies'

AND YEAR(streaming\_date) = 2020

AND MONTH(streaming\_date) = 6;

**Q30. Write an SQL query to find the npv of each query of the Queries table.**

Answer:

SELECT id,

FLOOR(SUM(profit / POW(1 + rate, year - 2020)) \* 100) / 100 AS npv

FROM Queries

GROUP BY id;

**Q31. Write an SQL query to find the npv of each query of the Queries table.**

**Return the result table in any order.**

Answer:

SELECT id,

FLOOR(SUM(profit / POW(1 + rate, year - 2020)) \* 100) / 100 AS npv

FROM Queries

GROUP BY id;

**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.**

Answer:

SELECT id,

CASE

WHEN unique\_id IS NULL THEN NULL

ELSE unique\_id

END AS unique\_id

FROM Employees;

**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.**

Answer:

SELECT u.name,

SUM(t.distance) AS travelled\_distance

FROM Users u

JOIN Trips t ON u.id = t.user\_id

GROUP BY u.id, u.name

ORDER BY travelled\_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.**

Answer:

SELECT p.product\_name,

SUM(o.unit \* p.price) AS amount

FROM Orders o

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

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

GROUP BY p.product\_name

HAVING SUM(o.unit) >= 100;

**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.**

Answer:

1. SELECT user\_id

FROM MovieRatings

GROUP BY user\_id

ORDER BY COUNT(movie\_id) DESC, user\_id ASC

LIMIT 1;

2. SELECT movie\_id

FROM MovieRatings

WHERE YEAR(created\_at) = 2020 AND MONTH(created\_at) = 2

GROUP BY movie\_id

ORDER BY AVG(rating) DESC, movie\_id ASC

LIMIT 1;

**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.**

Answer:

SELECT user\_id,

SUM(distance) AS travelled\_distance

FROM Rides

GROUP BY user\_id

ORDER BY travelled\_distance DESC, user\_id 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.**

Answer:

SELECT id,

CASE

WHEN unique\_id IS NULL THEN NULL

ELSE unique\_id

END AS unique\_id

FROM Employee;

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

Answer:

SELECT s.id, s.name

FROM Students

LEFT JOIN Departments d ON s.department\_id = d.id

WHERE d.id IS NULL;

**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.**

Answer:

SELECT CASE WHEN from\_id < to\_id THEN from\_id ELSE to\_id END AS person1,

CASE WHEN from\_id < to\_id THEN to\_id ELSE from\_id END AS person2,

COUNT(\*) AS call\_count,

SUM(duration) AS total\_duration

FROM Calls

GROUP BY 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.**

Answer:

SELECT product\_id,

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

FROM Prices

GROUP BY product\_id;

**Q41. Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.**

Answer:

SELECT warehouse\_id,

SUM(Width \* Length \* Height) AS volume

FROM Products

GROUP BY warehouse\_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.**

Answer:

SELECT sale\_date,

SUM(CASE WHEN fruit = 'apples' THEN 1 ELSE -1 END) AS diff

FROM Sales

GROUP BY sale\_date

ORDER BY sale\_date;

**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.**

Answer:

SELECT event\_date,

SUM(CASE WHEN fruit = 'apples' THEN 1 ELSE -1 END) AS diff

FROM Sales

GROUP BY event\_date

ORDER BY event\_date;

**Q44. Write an SQL query to report the managers with at least five direct reports.**

**Return the result table in any order.**

Answer:

SELECT managerId

FROM Employees

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.**

Answer:

SELECT d.dept\_name,

IFNULL(COUNT(e.student\_id), 0) AS student\_number

FROM Department d

LEFT JOIN Enrollments e ON d.dept\_id = e.dept\_id

GROUP BY d.dept\_id, d.dept\_name

ORDER BY student\_number DESC, d.dept\_name;

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

Answer:

SELECT c.customer\_id

FROM Customer c

WHERE NOT EXISTS (

SELECT p.product\_key

FROM Product p

WHERE NOT EXISTS (

SELECT cp.product\_key

FROM CustomerProduct cp

WHERE cp.customer\_id = c.customer\_id

AND cp.product\_key = p.product\_key

)

);

**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.**

Answer:

SELECT project\_id,

employee\_id,

experience\_years

FROM (

SELECT project\_id,

employee\_id,

experience\_years,

RANK() OVER (PARTITION BY project\_id ORDER BY experience\_years DESC) AS rnk

FROM ProjectEmployee

) ranked

WHERE rnk = 1;

**Q48. Write an SQL query that reports the books that have sold less than 10 copies in the last year,**

**excluding books that have been available for less than one month from today. Assume today is**

**2019-06-23.**

Answer:

SELECT b.book\_id,

b.name

FROM Books b

LEFT JOIN Orders o ON b.book\_id = o.book\_id

WHERE o.order\_date >= DATE\_SUB('2019-06-23', INTERVAL 1 YEAR)

AND o.order\_date <= '2019-06-23'

GROUP BY b.book\_id, b.name

HAVING SUM(o.quantity) < 10

AND DATEDIFF('2019-06-23', b.available\_from) >= 30;

**Q49. Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course\_id.**

**Return the result table ordered by student\_id in ascending order.**

Answer:

SELECT student\_id,

MIN(course\_id) AS course\_id,

MAX(grade) AS grade

FROM Enrollments

GROUP BY student\_id

**Q50. The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player\_id wins. Write an SQL query to find the winner in each group. Return the result table in any order.**

Answer:

SELECT m.match\_id,

CASE

WHEN t1.team\_id = m.host\_team AND (m.host\_goals > m.guest\_goals OR (m.host\_goals = m.guest\_goals AND m.host\_team < m.guest\_team)) THEN m.host\_team

ELSE m.guest\_team

END AS winner\_team

FROM Matches m

JOIN Teams t1 ON m.host\_team = t1.team\_id

JOIN Teams t2 ON m.guest\_team = t2.team\_id