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

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



SOLUTION 1: SELECT name

FROM city

WHERE countrycode = “USA” And population > 120000

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

Solution 2: SELECT \*

FROM city

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

Solution 3: SELECT \*

FROM city

WHERE id = 1661

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

Solution 4: SELECT \*

FROM city

WHERE countrycode = “JPN”

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

Solution 5: SELECT name

FROM city

WHERE countrycode = “JPN”

6. Query a list of **CITY** and **STATE** from the **STATION** table.  
The **STATION** table is described as follows:



Solution 6: SELECT city, state

FROM station

7. 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 7: SELECT DISTINCT city

FROM station

WHERE id % 2 = 0

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

Solution 8: SELECT (COUNT(city) - COUNT(DISTINCT city))

FROM station

9. 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 9: (SELECT city, LENGTH (city)

FROM station

ORDER BY LENGTH (city) ASC, city ASC

LIMIT 1)

UNION

(SELECT city, LENGTH (city)

FROM station

ORDER BY LENGTH (city) DESC, city ASC

LIMIT 1)

(Note : to use LIMIT in each query we have to use parenthesis, because in general LIMIT used once at the end.

10. 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 : 10 SELECT DISTINCT CITY

FROM station

WHERE city LIKE 'a%'

OR city LIKE 'e%'

OR city LIKE 'i%'

OR city LIKE 'u%'

OR city LIKE 'o%'

11. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION.

Your result cannot contain duplicates. \*/

Solution : 11 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'

12. /\* Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u)

as both their first and last characters.

Your result cannot contain duplicates. \*/

Solution 12: SELECT DISTINCT city

FROM

(SELECT city FROM station

WHERE city LIKE 'a%' OR city LIKE 'e%' OR city LIKE 'o%' OR city LIKE 'i%' OR city LIKE 'u%') AS subquery

WHERE city LIKE '%a' OR

city LIKE '%e' OR

city LIKE '%i' OR

city LIKE '%o' OR

city LIKE '%u' ;

(note here we used AS subquery alias it is must to do other wise error occur)

Note we can using Second Method : AND operator

SELECT DISTINCT city

FROM station

WHERE (city LIKE 'a%' OR city LIKE 'e%' OR city LIKE 'o%' OR city LIKE 'i%' OR city LIKE 'u%') AND

(city LIKE '%a' OR

city LIKE '%e' OR

city LIKE '%i' OR

city LIKE '%o' OR

city LIKE '%u') ;

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

Solution 13:

SELECT DISTINCT city

FROM station

WHERE NOT ( city LIKE 'a%' OR

city LIKE 'e%' OR

city LIKE 'i%' OR

city LIKE 'o%' OR

city LIKE 'u%');

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

Solution 14: SELECT DISTINCT city

FROM station

WHERE NOT (city LIKE '%a' OR

city LIKE '%e' OR

city LIKE '%i' OR

city LIKE '%o' OR

city LIKE '%u');

15. 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 15: SELECT DISTINCT city

FROM STATION

WHERE (NOT ( city LIKE '%a' OR

city LIKE '%e' OR

city LIKE '%i' OR

city LIKE '%o' OR

city LIKE '%u')

OR NOT ( city LIKE 'a%' OR

city LIKE 'e%' OR

city LIKE 'i%' OR

city LIKE 'o%' OR

city LIKE 'u%'));

16. 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 16: SELECT DISTINCT city

FROM station

WHERE NOT (city LIKE 'a%' OR city LIKE 'e%' OR city LIKE 'o%' OR city LIKE 'i%' OR city LIKE 'u%')

AND NOT

(city LIKE '%a' OR

city LIKE '%e' OR

city LIKE '%i' OR

city LIKE '%o' OR

city LIKE '%u') ;

17. Query the Name of any student in **STUDENTS** who scored higher than 75  Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

Solution 17: SELECT name

FROM students

WHERE marks > 75

ORDER BY SUBSTR (name, -3, 3), id;

18. Write a query that prints a list of employee names (i.e.: the name attribute)

from the Employee table in alphabetical order.

Solution 18: SELECT name

FROM employees

ORDER BY name;

19. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in **Employee** having a salary greater than 2000  per month who have been employees for less than 10 months. Sort your result by ascending employee\_id.

Solution 19: SELECT name

FROM employees

WHERE salary > 2000 AND months < 10

ORDER BY employee\_id ASC;

20. Given the **CITY** and **COUNTRY** tables, query the names of all cities where the CONTINENT is 'Africa'.

**Note:** CITY.CountryCode and COUNTRY.Code are matching key columns.

Solution 20: SELECT ci.NAME

from city ci

JOIN country co

ON ci.COUNTRYCODE = co.CODE

WHERE co.CONTINENT = 'africa'

21. Given the **CITY** and **COUNTRY** tables, query the names of all the continents (COUNTRY.Continent) and their respective average city populations (CITY.Population) rounded down to the nearest integer.

**Note:** CITY.CountryCode and COUNTRY.Code are matching key columns.

Solution 21:

SELECT country.CONTINENT, round(AVG(city.POPULATION)) as population

FROM country

JOIN city

ON city.COUNTRYCODE = country.CODE

GROUP BY city.COUNTRYCODE;

-- second method

SELECT country.CONTINENT, Floor(AVG(city.POPULATION)) as population

FROM country

JOIN city

ON city.COUNTRYCODE = country.CODE

GROUP BY country.CONTINENT;

-- third method

SELECT Country.Continent, FLOOR(AVG(City.Population))

FROM Country, City

WHERE Country.Code = City.CountryCode

GROUP BY Country.Continent ;

22. \*Ketty gives Eve a task to generate a report containing three columns:

Name, Grade and Mark. Ketty doesn't want the NAMES of those students

who received a grade lower than 8. The report must be in descending order by grade

-- i.e. higher grades are entered first. If there is more than one student with the s

ame grade (8-10) assigned to them, order those particular students by their name alphabetically.

Finally, if the grade is lower than 8, use "NULL" as their name and list them by their grades in

descending order. If there is more than one student with the same grade (1-7) assigned to them,

order those particular students by their marks in ascending order.

Write a query to help Eve.\*/

Solution 22:

SELECT ( CASE WHEN GRADE < 8 THEN NULL ELSE NAME END) AS NAME, GRADE, MARKS

FROM (

SELECT ID, NAME, MARKS, (CASE

WHEN MARKS >= 0 AND MARKS <= 9 THEN 1

WHEN MARKS >= 10 AND MARKS <= 19 THEN 2

WHEN MARKS >= 20 AND MARKS <= 29 THEN 3

WHEN MARKS >= 30 AND MARKS <= 39 THEN 4

WHEN MARKS >= 40 AND MARKS <= 49 THEN 5

WHEN MARKS >= 50 AND MARKS <= 59 THEN 6

WHEN MARKS >= 60 AND MARKS <= 69 THEN 7

WHEN MARKS >= 70 AND MARKS <= 79 THEN 8

WHEN MARKS >= 80 AND MARKS <= 89 THEN 9

ELSE 10 END) AS GRADE

FROM STUDENTS) AS subquery

ORDER BY GRADE DESC,

CASE WHEN GRADE >= 8 AND GRADE <= 10 THEN GRADE END DESC,

CASE WHEN GRADE >= 8 AND GRADE <= 10 THEN NAME END ASC,

CASE WHEN GRADE >= 1 AND GRADE <= 7 THEN MARKS END ASC;

23. Julia just finished conducting a coding contest, and she needs your help assembling the leaderboard! Write a query to print the respective hacker\_id and name of hackers who achieved full scores for more than one challenge. Order your output in descending order by the total number of challenges in which the hacker earned a full score. If more than one hacker received full scores in same number of challenges, then sort them by ascending hacker\_id.

Solution 23:

-- Method -1

select h.hacker\_id, h.name

from Submissions as s

join Hackers as h

on s.hacker\_id = h.hacker\_id

join Challenges as c

on s.challenge\_id = c.challenge\_id

join Difficulty as d

on c.Difficulty\_level = d.Difficulty\_level

where s.score = d.score

group by h.hacker\_id, h.name

having count(\*) > 1

order by count(\*) desc, h.hacker\_id;

-- method 2

SELECT HACKER\_ID, NAME

FROM(

SELECT HACKER\_ID, NAME, COUNT(CHALLENGE\_ID) AS COUNT\_C

FROM(

SELECT T2.HACKER\_ID, T2.NAME, T2.CHALLENGE\_ID

FROM(

SELECT T1.HACKER\_ID, T1.NAME, T1.SCORE, T1.CHALLENGE\_ID, C.DIFFICULTY\_LEVEL

FROM (

SELECT H.HACKER\_ID, H.NAME, S.CHALLENGE\_ID, S.SCORE

FROM HACKERS H

JOIN SUBMISSIONS S

ON H.HACKER\_ID = S.HACKER\_ID) T1

JOIN CHALLENGES C

ON T1.CHALLENGE\_ID = C.CHALLENGE\_ID) T2

JOIN DIFFICULTY D

ON T2.DIFFICULTY\_LEVEL = D.DIFFICULTY\_LEVEL

WHERE T2.SCORE = D.SCORE)

HAVING COUNT(CHALLENGE\_ID) > 1

GROUP BY HACKER\_ID, NAME

ORDER BY COUNT\_C DESC, HACKER\_ID ASC);

24. /\*

Harry Potter and his friends are at Ollivander's with Ron, finally replacing Charlie's old broken wand.

Hermione decides the best way to choose is by determining the minimum number of gold galleons needed to buy each non-evil wand of high power and age. Write a query to print the id, age, coins\_needed, and power of the wands that Ron's interested in, sorted in order of descending power. If more than one wand has same power, sort the result in order of descending age.

\*/

SELECT id, age, coins\_needed, power

FROM wands as w

JOIN wands\_property as wp

ON w.code = wp.code

WHERE wp.is\_evil =0 AND

w.coins\_needed = SELECT(MIN(w.coins\_needed)

FROM wands w1

JOIN wands\_property wp1

ON w1.code = wp1.code

WHERE wp.age = wp1.age AND

w.power = w1.power)

ORDER BY w.power DESC, wp.age DESC;