**BASIC SELECT**

1. **REVISITING THE SELECT QUERY 1**

Query all columns for all American cities in the **CITY** table with populations larger than 100000. The **CountryCode** for America is USA. The **CITY** table is described as follows:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol:

select \*from city

where population > 100000 and countrycode = 'USA';

1. **REVISITING THE SELECT QUERY 2**

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:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol: SELECT name FROM city WHERE population > 120000 and countrycode = 'USA';

1. **SELECT ALL**

Query all columns (attributes) for every row in the **CITY** table. The **CITY** table is described as follows:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol: select \* from city;

1. **SELECT BY ID**

Query all columns for a city in **CITY** with the ID 1661. The **CITY** table is described as follows:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol: select \* from city where id = 1661;

1. **JAPANESE CITIES’ ATTRIBUTES**

Query all attributes of every Japanese city in the **CITY** table. The **COUNTRYCODE** for Japan is JPN. The **CITY** table is described as follows:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol: select \* from city where countrycode = 'JPN';

1. **JAPANESE CITIES’ NAMES**

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

The city table is described as follows:

|  |  |
| --- | --- |
| Field | Type |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRYCODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

Sol:

Select name from city where countrycode = 'JPN';

1. **WEATHER OBSERVATION STATION 1**

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

Table

Description automatically generated

Sol:

Select city, state from station;

1. **WEATHER OBSERVATION STATION 3**

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

Table

Description automatically generated

Sol:- select distinct city

From station where id % 2 =0;

1. **WEATHER OBSERVATION STATION 4**

Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.  
The **STATION** table is described as follows:

// Table is same as above question

Sol: select count(city)-count(distinct city) from station;

1. **WEATHER OBSERVATION STATION 5**

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

// Table is same as above question

Sol: select city, length(city) from station

Order by length(City) asc, city asc limit 1;

select city, length(city) from station

Order by length(City) desc, city desc limit 1;

1. **WEATHER OBSERVATION STATION 6**

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

// Table is same as above question

Sol: select distinct city from station where city rlike '^[aeiouAEIOU]';

1. **WEATHER OBSERVATION STATION 7**

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

// Table is same as above question

Sol: select DISTINCT city from station where city rlike '[aeiouAEIOU]$';

1. **WEATHER OBSERVATION STATION 8**

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.

// Table is same as above question

**Sol: select DISTINCT city from station where city rlike '^[aeiouAEIOU].\*[aeiouAEIOU]$';**

1. **WEATHER OBSERVATION STATION 9**

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

// Table is same as above question

Sol: select distinct city from station where city not rlike '^[aeiouAEIOU]';

1. **WEATHER OBSERVATION STATION 10**

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

// Table is same as above question

**Sol:** select DISTINCT city from station where city not rlike '[aeiouAEIOU]$';

1. **WEATHER OBSERVATION STATION 11**

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.

// Table is same as above question

Sol: **select DISTINCT city from station where city not rlike '^[aeiouAEIOU].\*[aeiouAEIOU]$';**

1. **WEATHER OBSERVATION STATION 12**

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.

// Table is same as above question

Sol: SELECT DISTINCT CITY FROM STATION WHERE CITY NOT REGEXP '^[aeiou]'

AND CITY NOT REGEXP '[aeiou]$'

1. **HIGHER THAN 75 MARKS**

Query the Name of any student in **STUDENTS** who scored higher than  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.

**Table

Description automatically generated**

Sol: select name from students where marks > 75 order by right(name,3),id asc;

1. **EMPLOYEE NAMES**

Write a query that prints a list of employee names (i.e.: the name attribute) from the **Employee** table in alphabetical order.

Table

Description automatically generated

Sol: select name from employee order by name;

1. **EMPLOYEE SALARIES**

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

// Table is same as above

Sol: select name from employee where salary > 2000 and months < 10 order by employee\_id;