1) 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 LOWER (SUBSTRING (CITY, 1, 1)) NOT IN ('a', 'e', 'i', 'o', 'u')

 AND LOWER (SUBSTRING (CITY, -1, 1)) NOT IN ('a', 'e', 'i', 'o', 'u');
```

2) 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.

```
SELECT NAME FROM STUDENTS

WHERE Marks>75

order by Right (name, 3), ID;
```

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

```
Select Name from Employee

order by name;
```

4) 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.

```
SELECT Name from employee

where Salary > 2000 AND Months < 10

order by employee_id;
```

5) Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:

Equilateral: It's a triangle with sides of equal length.

Isosceles: It's a triangle with sides of equal length.

Scalene: It's a triangle with sides of differing lengths.

Not A Triangle: The given values of A, B, and C don't form a triangle.

```
SELECT

CASE

    when a=b and b=c then 'Equilateral'

    when (a+b <= c) or (a+c <= b) or (b+c <= a) then 'Not A Triangle'

    when a=b or b=c or c=a then 'Isosceles'
    else 'Scalene'
end as Triangle_type
from Triangles;
```

6) Query a count of the number of cities in CITY having a Population larger than 100000

```
Select count(*) FROM CITY
Where Population > 100000;
```

7) Query the total population of all cities in CITY where District is California.

```
SELECT sum(Population) From CITY
where District = 'California';
```

8) Query the average population of all cities in CITY where District is California.

```
Select AVG(Population) From CITY
Where District = 'California';
```

9) Query the average population for all cities in CITY, rounded down to the nearest integer.

```
Select Round (AVG(Population)) From CITY;
```

10) Query the sum of the populations for all Japanese cities in CITY. The COUNTRYCODE for Japan is JPN.

```
Select SUM(Population) From city
where CountryCode = 'JPN';
```

11) Query the difference between the maximum and minimum populations in CITY.

```
Select MAX(POPULATION) - MIN(POPULATION)
FROM CITY;
```

12) Write a query calculating the amount of error (i.e.: average monthly salaries), and round it up to the next integer.

```
Select CEIL(AVG(Salary)- AVG (Replace(salary, '0',"))
FROM EMPLOYEES;
CEIL (Rounds a number up to nearest integer e.g. 3.13 → 4)
```

13) Query the greatest value of the Northern Latitudes (LAT_N) from STATION that is less than 137.23445 Truncate your answer to decimal places.

```
Select MAX(ROUND(LAT_N,4)) FROM Station
where LAT_N < 137.2345;
```