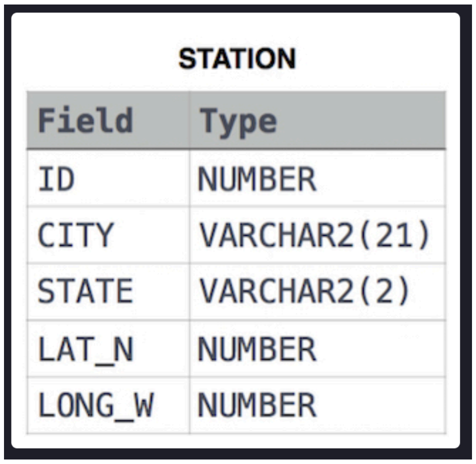
# SQL – Task 1

*Weather Observations*



**STATION**

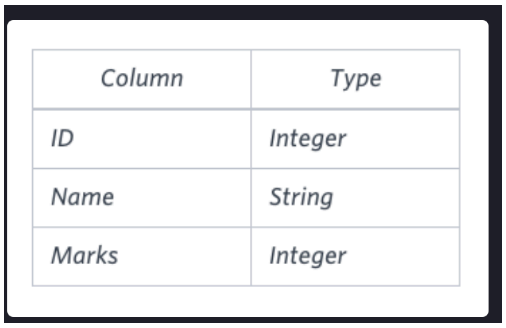
| Field | Type |
|---|---|
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT_N | NUMBER |
| LONG_W | NUMBER |

1. Query a list of **CITY** and **STATE** from the **STATION** table.
   The **STATION** table is described as follows: where **LAT_N** is the northern latitude and **LONG_W** is the western longitude.
2. 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.
3. Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.
   For example, if there are three records in the table with **CITY** values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns 1, because total number of records – number of unique city names = 3-2 = 1
4. Query the list of *CITY* names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result *cannot* contain duplicates.
5. Query the list of *CITY* names ending with vowels (a, e, i, o, u) from **STATION**. Your result *cannot* contain duplicates.
6. 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.
7. Query the list of *CITY* names from **STATION** that *do not end* with vowels. Your result cannot contain duplicates.
8. Query the list of *CITY* names from **STATION** that *do not start* with vowels. Your result cannot contain duplicates.

9. 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.
10. 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.

## *Higher than 75 Marks*

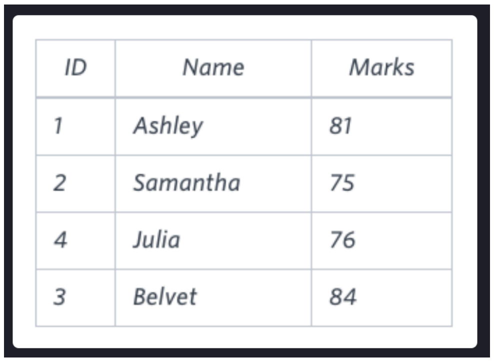
STUDENTS



| Column | Type |
|---|---|
| ID | Integer |
| Name | String |
| Marks | Integer |

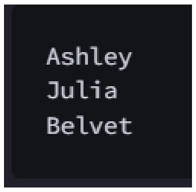
The *Name* column only contains uppercase (A-Z) and lowercase (a-z) letters.

Sample Input



| ID | Name | Marks |
|---|---|---|
| 1 | Ashley | 81 |
| 2 | Samantha | 75 |
| 4 | Julia | 76 |
| 3 | Belvet | 84 |

Sample Output



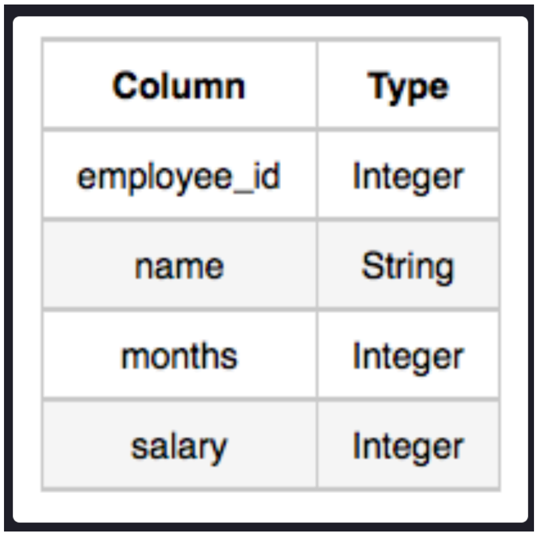
```
Ashley
Julia
Belvet
```

**Explanation**

Only Ashley, Julia, and Belvet have *Marks* > 75 . If you look at the last three characters of each of their names, there are no duplicates and 'ley' < 'lia' < 'vet'.

1. 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*.

## *Employee Names*



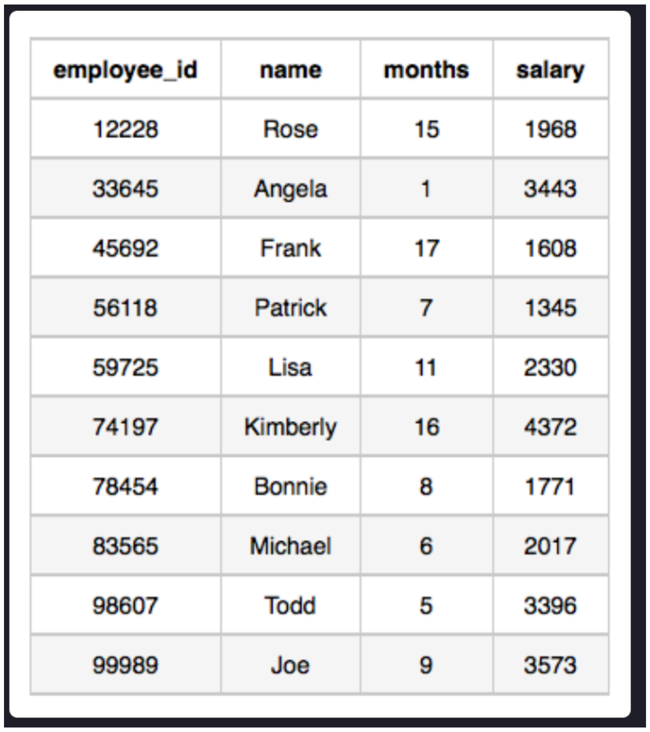
| Column | Type |
|---|---|
| employee_id | Integer |
| name | String |
| months | Integer |
| salary | Integer |

Employee Table

The **Employee** table containing employee data for a company is described as follows:

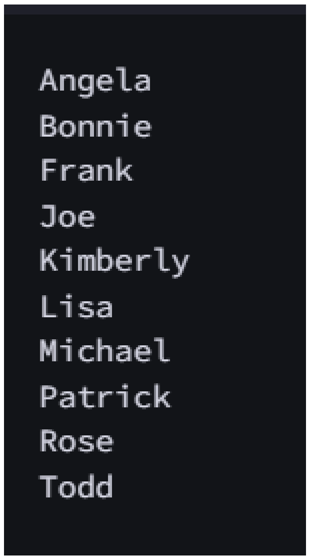
where *employee_id* is an employee's ID number, *name* is their name, *months* is the total number of months they've been working for the company, and *salary* is their monthly salary.

Sample Input



| employee_id | name | months | salary |
|---|---|---|---|
| 12228 | Rose | 15 | 1968 |
| 33645 | Angela | 1 | 3443 |
| 45692 | Frank | 17 | 1608 |
| 56118 | Patrick | 7 | 1345 |
| 59725 | Lisa | 11 | 2330 |
| 74197 | Kimberly | 16 | 4372 |
| 78454 | Bonnie | 8 | 1771 |
| 83565 | Michael | 6 | 2017 |
| 98607 | Todd | 5 | 3396 |
| 99989 | Joe | 9 | 3573 |

Sample Output

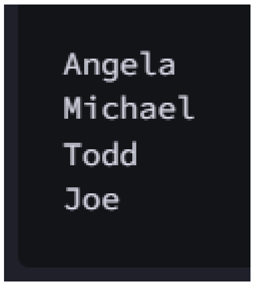


```
Angela
Bonnie
Frank
Joe
Kimberly
Lisa
Michael
Patrick
Rose
Todd
```

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

2. 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*.

   Sample Input is given above same in query 1
   Sample Output



```
Angela
Michael
Todd
Joe
```

   **Explanation**
   *Angela* has been an employee for 1 month and earns $3443 per month.
   *Michael* has been an employee for 6 months and earns $2017 per month.
   *Todd* has been an employee for 5 months and earns $3396 per month.
   *Joe* has been an employee for 9 months and earns $3573 per month.
   We order our output by ascending *employee_id*.

## *Population Census & African Cities*

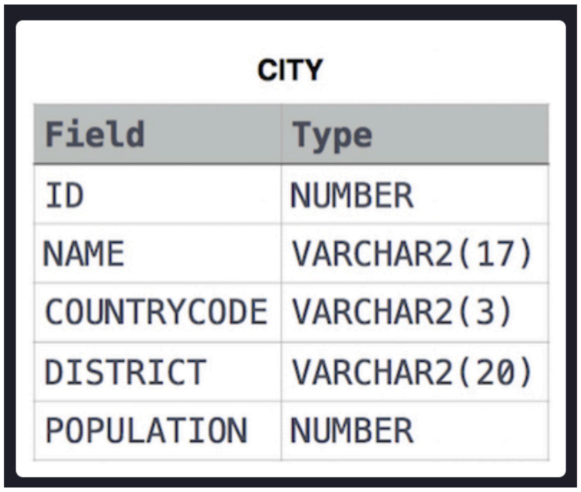
1. Given the **CITY** and **COUNTRY** tables, query the sum of the populations of all cities where the *CONTINENT* is *'Asia'*.

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

2. Given the **CITY** and **COUNTRY** tables, query the names of all cities where the *CONTINENT* is *'Africa'*.
3. 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.

**Input Format**

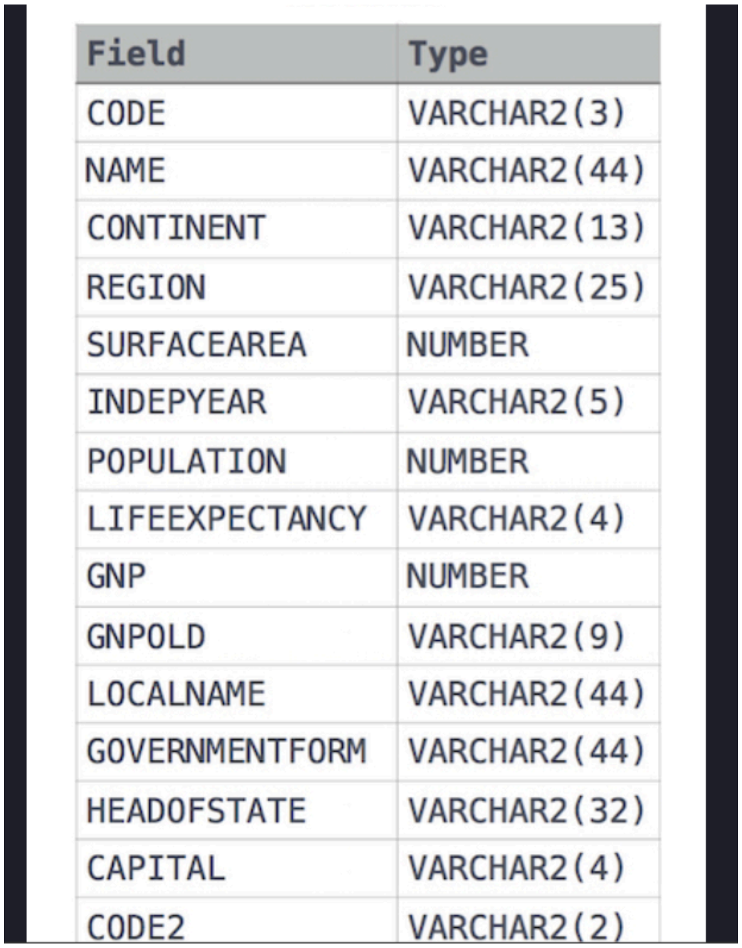
The **CITY** and **COUNTRY** tables are described as follows:



CITY

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

COUNTRY



| Field | Type |
|---|---|
| CODE | VARCHAR2(3) |
| NAME | VARCHAR2(44) |
| CONTINENT | VARCHAR2(13) |
| REGION | VARCHAR2(25) |
| SURFACEAREA | NUMBER |
| INDEPYEAR | VARCHAR2(5) |
| POPULATION | NUMBER |
| LIFEEXPECTANCY | VARCHAR2(4) |
| GNP | NUMBER |
| GNPOLD | VARCHAR2(9) |
| LOCALNAME | VARCHAR2(44) |
| GOVERNMENTFORM | VARCHAR2(44) |
| HEADOFSTATE | VARCHAR2(32) |
| CAPITAL | VARCHAR2(4) |
| CODE2 | VARCHAR2(2) |