**Q17:** 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*.

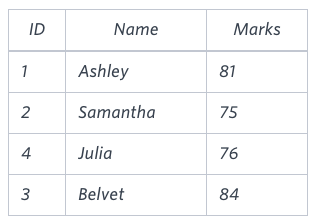
Input Format

The STUDENTS table is described as follows:



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

Sample Input



Sample Output

Ashley

Julia

Belvet

Explanation

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

**Solution:**

SELECT Name

FROM STUDENTS

WHERE MARKS > 75

ORDER BY RIGHT(NAME, 3) ASC, ID;

**Q18:** Write a query that prints a list of employee names (i.e.: the *name* attribute) from the Employee table in alphabetical order.

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



Sample Output

Angela

Bonnie

Frank

Joe

Kimberly

Lisa

Michael

Patrick

Rose

Todd

**Solution:**

SELECT name FROM Employee ORDER BY name;

**Q19:**  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*.

**Input Format**

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 the their monthly salary.

**Sample Input**



**Sample Output**

Angela

Michael

Todd

Joe

**Explanation**

*Angela* has been an employee for  month and earns  $3443 per month.

*Michael* has been an employee for  months and earns  $2017 per month.

*Todd* has been an employee for  months and earns  $3396 per month.

*Joe* has been an employee for  months and earns $3573 per month.

We order our output by ascending *employee\_id*.

**Solution:**

SELECT name FROM Employee WHERE salary > 2000 AND months < 10 ORDER BY employee\_id;

**Q20:** 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 3  sides of equal length.
* Isosceles: It's a triangle with 2 sides of equal length.
* Scalene: It's a triangle with 3 sides of differing lengths.
* Not A Triangle: The given values of *A*, *B*, and *C* don't form a triangle.

Input Format

The TRIANGLES table is described as follows:



Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input



Sample Output

Isosceles

Equilateral

Scalene

Not A Triangle

Explanation Values in the tuple (20, 20, 23) form an Isosceles triangle, because A = B.

Values in the tuple (20, 20, 20) form an Equilateral triangle, because A = B = C. Values in the tuple (20, 21, 22) form a Scalene triangle, because A / B ≠ C. Values in the tuple (13, 14, 30) cannot form a triangle because the combined value of sides A and B is not larger than that of side C.

**Solution:**

SELECT

CASE

WHEN A + B <= C OR A + C <= B OR B + C <= A THEN 'Not A Triangle'

WHEN A = B AND B = C THEN 'Equilateral'

WHEN A = B OR B = C OR A = C THEN 'Isosceles'

ELSE 'Scalene'

END AS triangle\_type

FROM TRIANGLES;