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*.

Input Format

Column	Type
ID	Integer
Name	String
Marks	Integer

The STUDENTS table is described as follows: 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

Solution:

```
SELECT NAME FROM STUDENTS WHERE MARKS>75 ORDER BY SUBSTR(NAME,-3),ID;
```

2. Placements

You are given three tables: *Students*, *Friends* and *Packages. Students* contains two columns: *ID* and *Name*. *Friends* contains two columns: *ID* and *Friend_ID* (*ID* of the ONLY best friend). *Packages* contains two columns: *ID* and *Salary* (offered salary in $ thousands per month).

Column	Type
ID	Integer
Name	String

Students

Column	Type
ID	Integer
Friend_ID	Integer

Friends

Column	Type
ID	Integer
Salary	Float

Packages

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

Sample Input

ID	Name
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Students

ID	Friend_ID
1	2
2	3
3	4
4	1

Friends

ID	Salary
1	15.20
2	10.06
3	11.55
4	12.12

Packages

Sample Output

Samantha

Julia

Scarlet

Solution:

```
SELECT NAME FROM STUDENTS S,FRIENDS F,PACKAGES P1,PACKAGES P2 WHERE
S.ID=F.ID AND S.ID=P1.ID AND F.FRIEND_ID=P2.ID AND P1.SALARY < P2.SALARY
ORDER BY P2.SALARY;
```

3. The Report

You are given two tables: *Students* and *Grades*. *Students* contains three columns *ID*, *Name* and *Marks*.

Column	Type
ID	Integer
Name	String
Marks	Integer

Grades contains the following data:

Grade	Min_Mark	Max_Mark
1	0	9
2	10	19
3	20	29
4	30	39
5	40	49
6	50	59
7	60	69
8	70	79
9	80	89
10	90	100

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 same 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.

Sample Input

ID	Name	Marks
1	Julia	88
2	Samantha	68
3	Maria	99
4	Scarlet	78
5	Ashley	63
6	Jane	81

Sample Output

Maria 10 99

Jane 9 81

Julia 9 88

Scarlet 8 78

NULL 7 63

NULL 7 68

Solution:

```
SELECT CASE
    WHEN G.GRADE >= 8 THEN S.NAME
    ELSE 'NULL' END,G.GRADE,S.MARKS
    FROM STUDENTS S,GRADES G
    WHERE S.MARKS >= MIN_MARK AND S.MARKS <=MAX_MARK
    ORDER BY G.GRADE DESC,S.NAME,S.MARKS;
```

4. Draw The Triangle 1

P(R) represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

```
* * * * *
* * * *
* * *
* *
*
```

Write a query to print the pattern *P(20)*.

Solution:

```
set serveroutput on;
declare
 res clob;
begin
for i in reverse 1..20 loop
     res := '';
    for j in 1..i loop
       res := res || '*' || ' ';
   end loop;
   dbms_output.put_line(res);
 end loop;
end;
/
```

5. Draw The Triangle 2

P(R) represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

```
*
* *
* * *
* * * *
* * * * *
```

Write a query to print the pattern *P(20)*.

Solution:

```
set serveroutput on;
declare
 res clob;
begin
for i in 1..20 loop
     res := '';
    for j in 1..i loop
       res := res || '*' || ' ';
   end loop;
   dbms_output.put_line(res);
 end loop;
end;
/
```

6. Print Prime Numbers

Write a query to print all *prime numbers* less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

For example, the output for all prime numbers <=10 would be:

2&3&5&7

Solution:

```
set serveroutput on;
declare
output clob := '';
co number;
begin
 for i in 2..1000 loop
    co := 0;
    for j in 2..(i/2) loop
      if mod(i,j)=0
        then
          co := 1;
          exit;
       end if;
   end loop;
   if co = 0
    then
      output :=  output || i ||'&';
   end if;
   end loop;
   dbms_output.put_line(substr(output,1,length(output)-1));
end;
/
```