Tricky SQL queries

**Unpivot Example:**

StudentMarks ( name varchar2(20),

Maths varchar2(20),

Science varchar2(20),

English varchar2(20)

);

Name, Maths, Science, English

Tilak, 90, 40, 60

Raj, 30, 20, 10

I want to get it arranged like the following:

Name, Subject, Marks

Tilak, Maths, 90

Tilak, Science, 40

Tilak, English, 60

select u.name, u.subject, u.marks from StudentMarks s unpivot (marks for subject in (Maths, Science, English) ) u;

Tilak MATHS 90

Tilak SCIENCE 40

Tilak ENGLISH 60

Raj MATHS 30

Raj SCIENCE 20

Raj ENGLISH 10

**Pivot Example**

Select \* from OCCUPATIONS;

Samantha Doctor

Julia Actor

Maria Actor

Meera Singer

Ashely Professor

Ketty Professor

Christeen Professor

Jane Actor

Jenny Doctor

Priya Singer

I want out put:

Jenny Ashley Meera Jane

Samantha Christeen Priya Julia

NULL Ketty NULL Maria

The first column is an alphabetically ordered list of Doctor names.

The second column is an alphabetically ordered list of Professor names.

The third column is an alphabetically ordered list of Singer names.

The fourth column is an alphabetically ordered list of Actor names.

The empty cell data for columns with less than the maximum number of names per occupation (in this case, the Professor and Actor columns) are filled with NULL values.

SELECT doctor,

professor,

singer,

actor

FROM

(SELECT \*

FROM

(SELECT Name,

occupation,

(ROW\_NUMBER() OVER (PARTITION BY occupation ORDER BY name)) AS row\_num

FROM occupations

ORDER BY name ASC

) pivot ( MIN(name) FOR occupation IN ('Doctor' AS doctor,'Professor' AS professor,'Singer' AS singer,'Actor' AS actor))

ORDER BY row\_num

);

**Select \* from TRIANGLES**

select A,B,C,(case

when (T.A=T.B and T.B=T.C and T.A+ T.B >T.C) then 'Equilateral'

when ((T.A=T.B OR T.B=T.C OR T.A=T.C) and T.A+ T.B >T.C) then 'Isosceles'

when (T.A <> T.B and T.B <> T.C and T.A <> T.C and T.A + T.B >T.C) then 'Scalene'

else 'Not A Triangle'

end) as "Type" from triangles T;

**output:**

20 20 23 Isosceles

20 20 20 Equilateral

20 21 22 Scalene

13 14 30 Not A Triangle

**🡪**

**select \* from BST**



Write a query to find the node type of *BST* ordered by the value of the node. Output one of the following for each node:

* *Root*: If node is root node.
* *Leaf*: If node is leaf node.
* *Inner*: If node is neither root nor leaf node.

select N, nvl2(P,(CASE WHEN N IN(select P from BST) THEN 'Inner' ELSE 'Leaf' END),'Root') from BST order by N;

output:

1 Leaf

2 Inner

3 Leaf

5 Root

6 Leaf

8 Inner

9 Leaf

**The SQL WITH** clause was introduced by Oracle in the Oracle 9i release 2 database. The SQL WITH clause allows you to give a sub-query block a name (a process also called sub-query refactoring), which can be referenced in several places within the main SQL query.

**Note:**When a query with a WITH clause is executed, first the query mentioned within the  clause is evaluated and the output of this evaluation is stored in a temporary relation. Following this, the main query associated with the WITH clause is finally executed that would use the temporary relation produced

Find all the employee whose salary is more than the average salary of all employees.

WITH temporaryTable(averageValue) as

(SELECT avg(Salary) from Employee),

SELECT EmployeeID,Name, Salary

FROM Employee, temporaryTable

WHERE Employee.Salary > temporaryTable.averageValue;

It is very helpful when you need the same set of results data multiple times. In such a case you can simply define a CTE for this data and reuse the same again and again by referencing it. It’s a kind of code reuse.

**Natural Join :**  
Natural Join joins two tables based on same attribute name and datatypes. The resulting table will contain all the attributes of both the table but keep only one copy of each common column.

**Inner Join :**  
Inner Join joins two table on the basis of the column which is explicitly specified in the ON clause. The resulting table will contain all the attributes from both the tables including common column also.

Duplicate deletes

DELETE FROM <table\_name> WHERE rowid not in

(SELECT MIN(rowid) FROM <table\_name> GROUP BY column1, column2, column3);

Another way:

DELETE FROM table\_name A

WHERE a.rowid >

ANY (

SELECT B.rowid FROM table\_name B

WHERE A.col1 = B.col1 AND A.col2 = B.col2);