**Delete duplicate rows using a single statement and no table creation.**

=>delete from EmpDup where sno in (

select sno from (

select \*,

RANK() OVER ( PARTITION BY empid,name ORDER BY sno DESC )rank

From EmpDup )T

where rank>1 );

=>delete from <table\_name>

where rowid not in ( select min(rowid)

from <Table\_name>

group by column1..,column2,...column3..)

**Identify the second highest salary from emp table having salary related data.**

=>select \* from(

select A.\*,rownum as rn from(

select empid,salary from EMP order by salary desc) A)B

where B.rn=2;

=>select empid,salary from(

select empid,salary,row\_number()over(order by salary desc)as rank from EMP)

where rank=2

**Statement to determine the oracle version used .**

=>SELECT \* FROM V$VERSION

=>SELECT version FROM V$INSTANCE

=>BEGIN DBMS\_OUTPUT.PUT\_LINE(DBMS\_DB\_VERSION.VERSION || '.' || DBMS\_DB\_VERSION.RELEASE); END;

**Query to accept a date and return the fist date of the quarter in which the date exists.**

SELECT TO\_CHAR(TO\_DATE('12/26/2012', 'MM/DD/YYYY'), 'Q')AS MY\_QTR FROM DUAL

**Show the server where the DB instance istalled**

SELECT SYS\_CONTEXT('USERENV','SERVER\_HOST') FROM dual;

**List the objects using a particular DB link**

select \* from ALL\_SOURCE where( upper(text)like '%<DB\_LINK\_NAME>%'

**Table A has 100 rows, Table B has Zero rows so number of rows returned from below query**

select a.\* from a, b;

The Query will return no records.

**Statement to print numbers from 1 to 100 in sequence.**

=> Select Rownum r From dual Connect By Rownum <= 100

=> SELECT LEVEL n FROM DUAL CONNECT BY LEVEL <= 100;

Table Employees

|  |  |
| --- | --- |
| NAME | GENDER |
| A | MALE |
| B | FEMALE |
| BB | FEMALE |
| CC | FEMALE |
| CD | MALE |
| DD | MALE |
| DE | FEMALE |

Write a query to give the o/p like below

MALE FEMALE

---- ------

3 4

select count(case when GENDER='MALE' then 1 end) as MALE,

sum(case when GENDER='FEMALE' then 1 else 0 end) as FEMALE From Employess;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Department** | **Manager** | **DOJ** |
| 1 | John | IT | 9 | 05-08-2010 |
| 2 | Alex | Corp | 0 | 06-03-2008 |
| 3 | Linda | IT | 9 | 07-02-2010 |
| 4 | Rahul | Purchase | 8 | 08-12-2010 |
| 5 | Ismail | Purchase | 8 | 09-08-2012 |
| 6 | Zheng | Sales | 7 | 10-05-2012 |
| 7 | Reiki | Sales | 2 | 11-02-2009 |
| 8 | Aris | Sales | 2 | 12-08-2011 |
| 9 | Jena | IT | 2 | 01-01-2008 |
| 10 | Bonny | IT | 9 | 01-01-2008 |

* **list the employees who are not managers**

select name from employee where id not in (select manager from employee);

* **manager with only one reportee**

select mgr from (select e1.id as mgr ,e2.id as id from employee e1

inner join employee e2 on

e1.id=e2.manager)abc

group by mgr having count(id)=1

* **what is the month with most hiring?**

select dt from (select dt,rank() over(order by cnt desc) as rnk from

(select to\_char(doj,'mm') as dt ,count(\*) as cnt from employee

group by to\_char(doj,'mm')))

where rnk=1;

* **what is the experience gap between the first employee and the latest?**

select max(doj)-min(doj) from employee;

* **name the manager with most reportees?**

select mgr from (select mgr,rank() over( order by cnt desc) as rnk from (select mgr,count(id) as cnt from (select e1.id as mgr,e2.id as id from employee e1

inner join employee e2

on e1.id=e2.manager)

group by mgr)) where rnk=1

* **list managers who joined after the reportees**

select mgr from(

select e1.id as mgr,e2.id as id,e1.doj as mdoj, e2.doj as edoj from employee e1 join employee e2 on

e1.id=e2.manager)

group by mgr having max(mdoj)>min(edoj)

* **department with most managers and how many?**

select department,cnt from (select department,rank() over(order by cnt desc) as rnk,cnt from (select count(distinct manager) as cnt ,department from employee

group by department)) where rnk=1

**Set of statements executed in the below sequence.**

select count(\*) from a;

|  |
| --- |
| Count(\*) |
| 100 |

Then 1 row inserted in table a.

rollback;

again 3 rows inserted in the table a;

select count(\*) from a;

|  |
| --- |
| count(\*) |
| 103 |

create table b using select \* from a;

rollback;

so what would be the o/p of below select statement

select count(\*) from a;

?

**How can we swap the gender column in a employee table ie: male should replace by female and vice versa.**

=> update Table1 set "col" = (case "col" when 'male' then 'female'

else 'male' end);

=> UPDATE GENDER SET SEX = DECODE(SEX,'MALE','FEMALE','FEMALE','MALE');

**Calculating age from DOB**

select trunc(months\_between(sysdate,dob)/12) year,

trunc(mod(months\_between(sysdate,dob),12)) month,

trunc(sysdate-add\_months(dob,trunc(months\_between(sysdate,dob)))) days

from ( Select to\_date('27011988','DDMMYYYY') dob from dual);

YEAR MONTH DAY

------ --------- ----------

9 5 26

**Display Middle Record**

select \* from (select A.\*,rownum rn from FCAP\_PROJECTS\_T A) where rn=(select trunc(count(\*)/2) from FCAP\_PROJECTS\_T)

Analytic Function

analytic\_function([ arguments ]) OVER (analytic\_clause)

The analytic\_clause breaks down into the following optional elements.

[ query\_partition\_clause ] [ order\_by\_clause [ windowing\_clause ] ]

### windowing\_clause

RANGE BETWEEN start\_point AND end\_point

ROWS BETWEEN start\_point AND end\_point

Possible values for "start\_point" and "end\_point" are:

* **UNBOUNDED PRECEDING** : The window starts at the first row of the partition, or the whole result set if no partitioning clause is used. Only available for start points.
* **UNBOUNDED FOLLOWING** : The window ends at the last row of the partition, or the whole result set if no partitioning clause is used. Only available for end points.
* **CURRENT ROW** : The window starts or ends at the current row. Can be used as start or end point.
* **value\_expr PRECEDING** : A physical or logical offset before the current row using a constant or expression that evaluates to a positive numerical value. When used with RANGE, it can also be an interval literal if the order\_by\_clause uses a DATE column.
* **value\_expr FOLLOWING** : As above, but an offset after the current row

SELECT empno, deptno, sal,

FIRST\_VALUE(sal) OVER (ORDER BY sal ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS previous\_sal,

LAST\_VALUE(sal) OVER (ORDER BY sal ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) AS next\_sal

FROM emp;

**Few queries related to analytical functions. lead, lag, rank, dense\_rank**

**LAG** (value\_expression [,offset] [,default]) OVER ([query\_partition\_clause] order\_by\_clause)

**LEAD** (value\_expression [,offset] [,default]) OVER ([query\_partition\_clause] order\_by\_clause)

* **value\_expression** - Can be a column or a built-in function, except for other analytic functions.
* **offset** - The number of rows preceeding/following the current row, from which the data is to be retrieved. The default value is 1.
* **default** - The value returned if the offset is outside the scope of the window. The default value is NULL.

SELECT empno, ename, job, sal,

LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal\_prev,

sal - LAG(sal, 1, 0) OVER (ORDER BY sal) AS sal\_diff

FROM emp;

EMPNO ENAME JOB SAL SAL\_PREV SAL\_DIFF

---------- ---------- --------- ---------- ---------- ----------

7369 SMITH CLERK 800 0 800

7900 JAMES CLERK 950 800 150

7876 ADAMS CLERK 1100 950 150

7521 WARD SALESMAN 1250 1100 150

SELECT empno, ename, job, sal,

LEAD(sal, 1, 0) OVER (ORDER BY sal) AS sal\_next,

LEAD(sal, 1, 0) OVER (ORDER BY sal) - sal AS sal\_diff

FROM emp;

EMPNO ENAME JOB SAL SAL\_NEXT SAL\_DIFF

---------- ---------- --------- ---------- ---------- ----------

7369 SMITH CLERK 800 950 150

7900 JAMES CLERK 950 1100 150

7876 ADAMS CLERK 1100 1250 150

7521 WARD SALESMAN 1250 1250 0

7654 MARTIN SALESMAN 1250 1300 50

## LISTAGG

SELECT deptno, LISTAGG(ename, ',') WITHIN GROUP (ORDER BY ename) AS employees

FROM emp

GROUP BY deptno;

DEPTNO EMPLOYEES

---------- --------------------------------------------------

10 CLARK,KING,MILLER

20 ADAMS,FORD,JONES,SCOTT,SMITH

30 ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD

## FIRST\_VALUE

SELECT empno, deptno, sal,

FIRST\_VALUE(sal) IGNORE NULLS

OVER (PARTITION BY deptno ORDER BY sal) AS lowest\_in\_dept

FROM emp;

EMPNO DEPTNO SAL LOWEST\_IN\_DEPT

---------- ---------- ---------- --------------

7934 10 1300 1300

7782 10 2450 1300

7839 10 5000 1300

7369 20 800 800

7876 20 1100 800

7566 20 2975 800

7788 20 3000 800

7902 20 3000 800

SELECT empno, deptno, sal,

FIRST\_VALUE(sal) IGNORE NULLS

OVER (PARTITION BY deptno ORDER BY sal ROWS 1 PRECEDING) AS preceding\_in\_dept

FROM emp;

EMPNO DEPTNO SAL PRECEDING\_IN\_DEPT

---------- ---------- ---------- -----------------

7934 10 1300 1300

7782 10 2450 1300

7839 10 5000 2450

7369 20 800 800

7876 20 1100 800

7566 20 2975 1100

7788 20 3000 2975

7902 20 3000 3000

## LAST\_VALUE

SELECT empno, deptno, sal,

LAST\_VALUE(sal) IGNORE NULLS

OVER (PARTITION BY deptno ORDER BY sal) AS highest\_in\_dept

FROM emp;

EMPNO DEPTNO SAL HIGHEST\_IN\_DEPT

---------- ---------- ---------- ---------------

7934 10 1300 1300

7782 10 2450 2450

7839 10 5000 5000

7369 20 800 800

7876 20 1100 1100

7566 20 2975 2975

7788 20 3000 3000

7902 20 3000 3000

SELECT empno, deptno, sal,

LAST\_VALUE(sal) IGNORE NULLS

OVER (PARTITION BY deptno ORDER BY sal RANGE BETWEEN

UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS highest\_in\_dept

FROM emp;

EMPNO DEPTNO SAL HIGHEST\_IN\_DEPT

---------- ---------- ---------- ---------------

7934 10 1300 5000

7782 10 2450 5000

7839 10 5000 5000

7369 20 800 3000

7876 20 1100 3000

7566 20 2975 3000

7788 20 3000 3000

7902 20 3000 3000

## RANK & DENSE\_RANK

SELECT empno, deptno, sal,

RANK() OVER (PARTITION BY deptno ORDER BY sal) "rank"

FROM emp;

EMPNO DEPTNO SAL rank

---------- ---------- ---------- ----------

7900 30 950 1

7654 30 1250 2

7521 30 1250 2

7844 30 1500 4

SELECT empno, deptno, sal,

DENSE\_RANK() OVER (PARTITION BY deptno ORDER BY sal) "rank"

FROM emp;

EMPNO DEPTNO SAL rank

---------- ---------- ---------- ----------

7654 30 1250 2

7521 30 1250 2

7844 30 1500 3

**Is it possible to update Views? If yes, How, If Not, Why?**

We can perform DML on the view when   
1.View is built based on a single table  
2.Primary key of the table should be available in select clause used for view  
We can’t perform DML on the view when  
1. View is built on multiple tables   
However we can use trigger for DML operations on multi table view

**TRANSLATE Function**

The Oracle/PLSQL TRANSLATE function replaces a sequence of characters in a string with another set of characters. However, it replaces a single character at a time

TRANSLATE( string1, string\_to\_replace, replacement\_string )

=>**TRANSLATE**('1tech23', '123', '456')

*Result:* '4tech56'

=>**TRANSLATE**('222tech', '2ec', '3it')

*Result:* '333tith'

**START WITH and CONNECT BY in Oracle SQL**

[***select***](http://www.adp-gmbh.ch/ora/sql/select.html) ***... start with initial-condition connect by*** [***nocycle***](http://www.adp-gmbh.ch/ora/sql/connect_by_nocycle.html) ***recurse-condition***

[***select***](http://www.adp-gmbh.ch/ora/sql/select.html) ***... connect by recurse-condition***

*The start with .. connect by clause can be used to select data that has a hierarchical relationship (usually some sort of parent->child (boss->employee or thing->parts).*

***recurse-condition****can make use of the keyword****prior****:*

***connect by***

***prior foo = bar***

***select*** [***lpad***](http://www.adp-gmbh.ch/ora/sql/rpad.html)***(' ',2\*(***[***level***](http://www.adp-gmbh.ch/ora/sql/level.html)***-1)) ||*** [***to\_char***](http://www.adp-gmbh.ch/ora/sql/to_char.html)***(child) s***

***from test\_connect\_by***

***start with parent is*** [***null***](http://www.adp-gmbh.ch/ora/misc/null.html)

***connect by prior child = parent;***

### sys\_connect\_by\_path

With [sys\_connect\_by\_path](http://www.adp-gmbh.ch/ora/sql/sys_connect_by_path.html) it is possible to show the entire path from the top level down to the 'actual' child:

## Using hierarchical result sets

With this technique, it is possible to show all kind of hierarchical data relations. Here [is an example](http://www.adp-gmbh.ch/ora/misc/recursively_list_privilege.html) that lists [privileges](http://www.adp-gmbh.ch/ora/misc/users_roles_privs.html#privileges), [roles](http://www.adp-gmbh.ch/ora/misc/users_roles_privs.html#roles) and[users](http://www.adp-gmbh.ch/ora/misc/users_roles_privs.html#users) in their hierarchical relation.

See also [flat hiearchy](http://www.adp-gmbh.ch/ora/sql/flat_hierarchy.html).

## connect\_by\_root

**connect\_by\_root** is a new operator that comes with [Oracle 10g](http://www.adp-gmbh.ch/ora/misc/10g.html)and enhances the ability to perform hierarchical queries.

I have yet to dig into this subject and will write about it when things become clearer.

## connect\_by\_is\_leaf

**connect\_by\_isleaf** is a new operator that comes with [Oracle 10g](http://www.adp-gmbh.ch/ora/misc/10g.html)and enhances the ability to perform hierarchical queries.

I have yet to dig into this subject and will write about it when things become clearer.

## connect\_by\_iscycle

**connect\_by\_is\_cycle** is a new operator that comes with [Oracle 10g](http://www.adp-gmbh.ch/ora/misc/10g.html) and enhances the ability to perform hierarchical queries.

I have yet to dig into this subject and will write about it when things become clearer.

### SELF JOIN

A self-join can be an inner join or an outer join. A table is joined to itself based upon a field or combination of fields that have duplicate data in different records. The data-type of the inter-related columns must be of the same type or needs to cast them in same type.

|  |  |  |
| --- | --- | --- |
| **emp\_id** | **emp\_name** | **emp\_supv** |
| 4081 | Nancy Brown | NULL |
| 4083 | Peter Parker | 4081 |
| 4055 | Jacob Miller | 4083 |
| 4058 | Mary Ray | 4083 |
| 4060 | Jane Smith | 4081 |
| 4061 | Bob Hunter | 4081 |

***SELECT a.emp\_id AS “Emp\_ID”,a.emp\_name AS “Employee Name”,***

***b.emp\_id AS “Supervisor ID”,b.emp\_name AS “Supervisor Name”***

***FROM employees AS a LEFT OUTER JOIN employees AS b***

***ON a.emp\_supv = b.emp\_id***

The corresponding output table will look like:

Emp\_ID               Employee Name               Supervisor ID                  Supervisor Name

4081                    Nancy Brown                     NULL                                   NULL

4083                    Peter Parker                       4081                                   Nancy Brown

4055                     Jacob Miller                       4083                                  Peter Parker

4058                     Mary Ray                            4083                                  Peter Parker

4060                   Jane Smith                           4081                                  Nancy Brown

4061                    Bob Hunter                         4081                                 Nancy Brown

***It’s important to select the correct join type while writing a SELF JOIN statement. In this case, we used a LEFT OUTER JOIN to ensure we had output records corresponding to each employee. If we used an INNER JOIN, then Nancy Brown, the CEO would have been omitted as she does not have a supervisor.***

**With Clause**

***WITH dept\_count AS (***

***SELECT /\*+ MATERIALIZE \*/ deptno, COUNT(\*) AS dept\_count***

***FROM emp***

***GROUP BY deptno)***

***SELECT ...***

***WITH dept\_count AS (***

***SELECT /\*+ INLINE \*/ deptno, COUNT(\*) AS dept\_count***

***FROM emp***

***GROUP BY deptno)***

***SELECT ...***

The MATERIALIZE and INLINE optimizer hints can be used to influence the decision. The undocumented MATERIALIZE hint tells the optimizer to resolve the subquery as a global temporary table, while the INLINE hint tells it to process the query inline

## https://www.toptal.com/sql/interview-questions

## <http://www.techbeamers.com/sql-interview-questions-answers-experienced/>

.how to write sql query for the below scenario  
I/p:ORACLE

O/p:  
O  
R  
A  
C  
L  
E  
i.e, splitting into multiple columns a string using sql.

**Answer:**

Select Substr(‘ORACLE’,Level,1) From Dual  
Connect By Level<= Length(‘ORACLE’);

## Pivot in SQL:

In previous article i have explained about [**Oracle 11 G features**](http://www.complexsql.com/oracle-11-g-new-features/);One of the function named Pivot is new Oracle 11 G feature which is used specifically to transpose or convert rows in to columns or columns in to rows (Unpivot) to display the result in crosstab format. The simple meaning of Pivot in English is ‘Center point on  which mechanism turns or oscillates’.Just like that Pivot in SQL is used to convert the column values in to attributes(transpose rows in to columns).

***“Pivot in SQL helps to convert column values into attributes or transpose rows into columns.”***



[**CLICK HERE TO GET 20 MOST IMPORTANT SQL QUERIES FOR INTERVIEW**](http://www.complexsql.com/complex-sql-queries-examples-with-answers/)

Following are the simple steps to perform the Pivoting:

**1. Seperate the Rows:**

The first step is seperate all rows.

**2.Aggregate Required Data:**

We need to aggregate the required data using the [**aggregate functions**](http://www.complexsql.com/sql-functions-list/) like Sum,Avg,Min,Max,Count function.

**3.Convert aggregated data into columns:**

The last step is to transpose aggregated data into column.

**Syntax:**

*SELECT Column\_name1,Column\_name2…..FROM (SubQuery)  
PIVOT [XML]  
( pivot\_clause\_Aggregate\_Function  
pivot\_for\_clause  
pivot\_in\_clause )  
WHERE  Condition;*

Following are different parameters and arguments used in Pivot/Unpivot:

**1.Pivot Clause with Aggregate Function:**

There must be the aggregate function to Pivot the table. The aggregate functions like Sum,Avg,Min,Max and Count needs to be used for Pivoting table.

**2.Pivot For Clause:**

The Column name which needs to be converted from rows to column.

**3.Pivot IN Clause:**

These are nothing but the list of values to column 2 to pivot it in to headings in to cross table result.

[**4.Subquery:**](http://www.complexsql.com/subqueries-correlated-subquery/)

We need to use the Subquery for fetching the records instead of list of values.In this case result of subquery would be used to determine the values from column to pivot in to headings.

**Real Life Example:**

Consider Following table.We need count of Employees department wise where Department ID is column.

Name of Table: Department

|  |  |
| --- | --- |
| Department ID | Employee Name |
| 100 | Amit |
| 100 | Rohan |
| 101 | Rohit |
| 102 | Pradnya |

We need to convert the Department ID column in to Rows and then We need to display the count of  employees .

*Select \* from*

*(Select DepartmentId from Department)*

*PIVOT*

*(Count(Employee\_name)*

*For DepartmentId IN (100,101,102)*

*);*

Using above Pivot statement the DepartmentId is pivoted and the table is been transposed and we are using Count() as aggregate function

**Output:**

|  |  |  |
| --- | --- | --- |
| 100 | 101 | 103 |
| 2 | 1 | 1 |

**Using With Clause:**

*WITH Pivot\_Department AS (*

*Select DepartmentId   from Department*

*)*

*SELECT \**

*FROM   Pivot\_Department*

*PIVOT (*

*Count(Employee\_name)        — pivot\_clause*

*FOR DepartmentId                      — pivot\_for\_clause*

*IN  (100,101,102)                           — pivot\_in\_clause*

*);*

**Output:**

|  |  |  |
| --- | --- | --- |
| 100 | 101 | 103 |
| 2 | 1 | 1 |

### Unpivot:

We have checked the Pivot in SQL which is used to convert the rows in to columns.Unpivot simply means opposite of pivot which is used in opposite of Pivot table but without dis-aggregating the data.One row of data for every column is unpivoted.

*The Unpivot operator converts column based  data in to individual rows.*

**Syntax:**

*SELECT Column\_name1,Column\_name2…*

*FROM*

*UNPIVOT [INCLUDE|EXCLUDENULLS]*

*( unpivot\_clause*

*unpivot\_for\_clause*

*unpivot\_in\_clause )*

*WHERE  Condition;*

The Syntax of Unpivot is quite similar to Pivot but there are some differences:

* **unpivot\_clause:** this clause specifies a name for a column to represent the unpivoted measure values. In our previous pivot examples, the measure column was the Count employee department grouping.
* **unpivot\_for\_clause:** the unpivot\_for\_clause specifies the name for the column that will result from our unpivot query. The data in this column describes the measure values in the unpivot\_clause column; and
* **unpivot\_in\_clause:** this contains the list of pivoted columns (not values) to be unpivoted.

We need to convert new object for Pivoted data.So Consider to convert the view for the pivoted data.

*Create view V\_Pivot\_Data*

*as*

*Select \* from*

*(Select DepartmentId,Employee\_name from Department)*

*PIVOT*

*(Count(Employee\_name)*

*For DepartmentId IN (100,101,102)*

*);*

*Select \* from V\_Pivot\_Data;*

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| Employee Name | 100 | 101 | 102 |
| Amit | 1 | 0 | 0 |
| Rohan | 1 | 0 | 0 |
| Rohit | 0 | 1 | 0 |
| Pradnya | 0 | 0 | 1 |

We will now unpivot the data:

select \* from V\_Pivot\_Data

Unpivot

(

Employee\_name,             –unpivot call

For DepartmentId IN (100,101,102)

);

The output of the Query will be:

|  |  |
| --- | --- |
| Department ID | Employee Name |
| 100 | Amit |
| 100 | Rohan |
| 101 | Rohit |
| 102 | Pradnya |

## Occupations

[Pivot](https://www.hackerrank.com/external_redirect?to=https://en.wikipedia.org/wiki/Pivot_table) the Occupation column in **OCCUPATIONS** so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively.

**Note:** Print **NULL** when there are no more names corresponding to an occupation.

**Input Format**

The **OCCUPATIONS** table is described as follows:Occupation will only contain one of the following values: **Doctor**, **Professor**, **Singer** or **Actor**.

**Sample Input**



**Sample Output**

Jenny Ashley Meera Jane

Samantha Christeen Priya Julia

NULL Ketty NULL Maria

**Explanation**

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.

## Binary Tree Nodes

You are given a table, *BST*, containing two columns: *N*and *P,* where *N* represents the value of a node in *Binary Tree*, and *P* is the parent of *N*.



Write a query to find the node type of *Binary Tree* 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.

**Sample Input**



**Sample Output**

1 Leaf

2 Inner

3 Leaf

5 Root

6 Leaf

8 Inner

9 Leaf

**Explanation**

The *Binary Tree* below illustrates the sample:



Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:

Given the table schemas below, write a query to print the *company\_code*, *founder* name, total number of *lead*managers, total number of *senior* managers, total number of *managers*, and total number of *employees*. Order your output by ascending *company\_code*.

**Note:**

* The tables may contain duplicate records.
* The *company\_code* is string, so the sorting should not be **numeric**. For example, if the *company\_codes* are *C\_1*, *C\_2*, and *C\_10*, then the ascending *company\_codes* will be *C\_1*, *C\_10*, and *C\_2*.

**Input Format**

The following tables contain company data:

* *Company:* The *company\_code* is the code of the company and *founder* is the founder of the company.
* 
* *Lead\_Manager:* The *lead\_manager\_code* is the code of the lead manager, and the *company\_code* is the code of the working company.
* *Senior\_Manager:* The *senior\_manager\_code* is the code of the senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.
* 
* *Manager:* The *manager\_code* is the code of the manager, the *senior\_manager\_code* is the code of its senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.
* *Employee:* The *employee\_code* is the code of the employee, the *manager\_code* is the code of its manager, the *senior\_manager\_code* is the code of its senior manager, the *lead\_manager\_code* is the code of its lead manager, and the *company\_code* is the code of the working company.
* 

**Sample Input**

*Company* Table:



*Lead\_Manager* Table:



*Senior\_Manager* Table:

*Manager* Table:*Employee* Table:

**Sample Output**

C1 Monika 1 2 1 2

C2 Samantha 1 1 2 2

**Explanation**

In company *C1*, the only lead manager is *LM1*. There are two senior managers, *SM1* and *SM2*, under *LM1*. There is one manager, *M1*, under senior manager *SM1*. There are two employees, *E1* and *E2*, under manager *M1*.

In company *C2*, the only lead manager is *LM2*. There is one senior manager, *SM3*, under *LM2*. There are two managers, *M2* and *M3*, under senior manager *SM3*. There is one employee, *E3*, under manager *M2*, and another employee, *E4*, under manager, *M3*.

select c.company\_code,c.founder,count(distinct(LM.lead\_manager\_code)),count(distinct(SM.senior\_manager\_code)),count(distinct(M.manager\_code)),count(distinct(employee\_code)) from company c

left outer join

(select distinct \* from Lead\_Manager) LM

on c.company\_code=LM.company\_code

left outer join

(select distinct \* from Senior\_Manager )SM

on LM.lead\_manager\_code=SM.lead\_manager\_code

and SM.company\_code=c.company\_code

left outer join

(select distinct \* from Manager) M

on SM.senior\_manager\_code=M.senior\_manager\_code

and LM.lead\_manager\_code=M.lead\_manager\_code

and M.company\_code=c.company\_code

left outer join

(select distinct \* from Employee) E

on M.Manager\_code=E.manager\_code

and M.senior\_manager\_code=SM.senior\_manager\_code

and M.Lead\_manager\_code=LM.lead\_manager\_code

group by c.company\_code,c.founder

order by c.company\_code,c.founder

;