What is the difference between inner and outer join? Explain with example.
Inner Join

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).

Inner join returns rows when there is at least one match in both tables

If none of the record matches between two tables, then INNER JOIN will return a NULL set. Below is an example of INNER JOIN and the resulting set.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE
FROM DEPT dept, EMPLOYEE emp
WHERE emp.dept_id = dept.id
Department Employee
HR Inno
HR Privy
Engineering Robo
Engineering Hash
Engineering Anno
Engineering Darl
Marketing Pete
Marketing Meme
Sales Tomiti
Sales Bhuti
Outer Join

Outer Join can be full outer or single outer

Outer Join, on the other hand, will return matching rows from both tables as well as any unmatched rows from one or both the tables (based on whether it is single outer or full outer join respectively).

Notice in our record set that there is no employee in the department 5 (Logistics). Because of this if we perform inner join, then Department 5 does not appear in the above result. However in the below query we perform an outer join (dept left outer join emp), and we can see this department.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE
FROM DEPT dept, EMPLOYEE emp
WHERE dept.id = emp.dept_id (+)
Department Employee
HR Inno
HR Privy
Engineering Robo
Engineering Hash
Engineering Anno

Engineering Darl
Marketing Pete
Marketing Meme
Sales Tomiti
Sales Bhuti
Logistics
The (+) sign on the emp side of the predicate indicates that emp is the outer table here. The above SQL can be alternatively written as below (will yield the same result as above):

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE
FROM DEPT dept LEFT OUTER JOIN EMPLOYEE emp
ON dept.id = emp.dept_id
What is the difference between JOIN and UNION?
SQL JOIN allows us to "lookup" records on other table based on the given conditions between two tables. For example, if we have the department ID of each employee, then we can use this department ID of the employee table to join with the department ID of department table to lookup department names.

UNION operation allows us to add 2 similar data sets to create resulting data set that contains all the data from the source data sets. Union does not require any condition for joining. For example, if you have 2 employee tables with same structure, you can UNION them to create one result set that will contain all the employees from both of the tables.

SELECT * FROM EMP1
UNION
SELECT * FROM EMP2;
What is the difference between UNION and UNION ALL?
UNION and UNION ALL both unify for add two structurally similar data sets, but UNION operation returns only the unique records from the resulting data set whereas UNION ALL will return all the rows, even if one or more rows are duplicated to each other.

In the following example, I am choosing exactly the same employee from the emp table and performing UNION and UNION ALL. Check the difference in the result.

SELECT * FROM EMPLOYEE WHERE ID = 5
UNION ALL
SELECT * FROM EMPLOYEE WHERE ID = 5
ID MGR_ID DEPT_ID NAME SAL DOJ
5.0 2.0 2.0 Anno 80.0 01-Feb-2012
5.0 2.0 2.0 Anno 80.0 01-Feb-2012
SELECT * FROM EMPLOYEE WHERE ID = 5
UNION
SELECT * FROM EMPLOYEE WHERE ID = 5
ID MGR_ID DEPT_ID NAME SAL DOJ
5.0 2.0 2.0 Anno 80.0 01-Feb-2012
What is the difference between WHERE clause and HAVING clause?

WHERE and HAVING both filters out records based on one or more conditions. The difference is, WHERE clause can only be applied on a static non-aggregated column whereas we will need to use HAVING for aggregated columns.

To understand this, consider this example.
Suppose we want to see only those departments where department ID is greater than 3. There is no aggregation operation and the condition needs to be applied on a static field. We will use WHERE clause here:

```
SELECT * FROM DEPT WHERE ID > 3
ID      NAME
4       Sales
5       Logistics
```

Next, suppose we want to see only those Departments where Average salary is greater than 80. Here the condition is associated with a non-static aggregated information which is “average of salary”. We will need to use HAVING clause here:

```
SELECT dept.name DEPARTMENT, avg(emp.sal) AVG_SAL
FROM DEPT dept, EMPLOYEE emp
WHERE dept.id = emp.dept_id (+)
GROUP BY dept.name
HAVING AVG(emp.sal) > 80
DEPARTMENT      AVG_SAL
Engineering     90
```

As you see above, there is only one department (Engineering) where average salary of employees is greater than 80.

What is the difference among UNION, MINUS and INTERSECT?
UNION combines the results from 2 tables and eliminates duplicate records from the result set.

MINUS operator when used between 2 tables, gives us all the rows from the first table except the rows which are present in the second table.

INTERSECT operator returns us only the matching or common rows between 2 result sets.

To understand these operators, let’s see some examples. We will use two different queries to extract data from our emp table and then we will perform UNION, MINUS and INTERSECT operations on these two sets of data.

UNION

```
SELECT * FROM EMPLOYEE WHERE ID = 5
UNION
SELECT * FROM EMPLOYEE WHERE ID = 6
ID      MGR_ID  DEPT_ID NAME    SAL     DOJ
5       2       2.0     Anno    80.0    01-Feb-2012
```

6 2 2.0 Darl 80.0 11-Feb-2012
MINUS

SELECT * FROM EMPLOYEE
MINUS
SELECT * FROM EMPLOYEE WHERE ID > 2
ID MGR_ID DEPT_ID NAME SAL DOJ
1 2 Hash 100.0 01-Jan-2012
2 1 2 Robo 100.0 01-Jan-2012
INTERSECT

SELECT * FROM EMPLOYEE WHERE ID IN (2, 3, 5)
INTERSECT
SELECT * FROM EMPLOYEE WHERE ID IN (1, 2, 4, 5)
ID MGR_ID DEPT_ID NAME SAL DOJ
5 2 2 Anno 80.0 01-Feb-2012
2 1 2 Robo 100.0 01-Jan-2012
What is Self Join and why is it required?
Self Join is the act of joining one table with itself.

Self Join is often very useful to convert a hierarchical structure into a flat structure

In our employee table example above, we have kept the manager ID of each employee in the same row as that of the employee. This is an example of how a hierarchy (in this case employee-manager hierarchy) is stored in the RDBMS table. Now, suppose if we need to print out the names of the manager of each employee right beside the employee, we can use self join. See the example below:

SELECT e.name EMPLOYEE, m.name MANAGER
FROM EMPLOYEE e, EMPLOYEE m
WHERE e.mgr_id = m.id (+)
EMPLOYEE MANAGER
Pete Hash
Darl Hash
Inno Hash
Robo Hash
Tomiti Robo
Anno Robo
Privy Robo
Meme Pete
Bhuti Tomiti
Hash
The only reason we have performed a left outer join here (instead of INNER JOIN) is we have one employee in this table without a manager (employee ID = 1). If we perform inner join, this employee will not show-up.

How can we transpose a table using SQL (changing rows to column or vice-versa) ?

The usual way to do it in SQL is to use CASE statement or DECODE statement.

How to generate row number in SQL Without ROWNUM
Generating a row number – that is a running sequence of numbers for each row is not easy using plain SQL. In fact, the method I am going to show below is not very generic either. This method only works if there is at least one unique column in the table. This method will also work if there is no single unique column, but collection of columns that is unique. Anyway, here is the query:

```
SELECT name, sal, (SELECT COUNT(*)  FROM EMPLOYEE i WHERE o.name >= i.name) row_num
FROM EMPLOYEE o
order by row_num
```

NAME	SAL	ROW_NUM
Anno	80	1
Bhuti	60	2
Darl	80	3
Hash	100	4
Inno	50	5
Meme	60	6
Pete	70	7
Privy	50	8
Robo	100	9
Tomiti	70	10

The column that is used in the row number generation logic is called “sort key”. Here sort key is “name” column. For this technique to work, the sort key needs to be unique. We have chosen the column “name” because this column happened to be unique in our Employee table. If it was not unique but some other collection of columns was, then we could have used those columns as our sort key (by concatenating those columns to form a single sort key).

Also notice how the rows are sorted in the result set. We have done an explicit sorting on the row_num column, which gives us all the row numbers in the sorted order. But notice that name column is also sorted (which is probably the reason why this column is referred as sort-key). If you want to change the order of the sorting from ascending to descending, you will need to change “>=” sign to “<=” in the query.

As I said before, this method is not very generic. This is why many databases already implement other methods to achieve this. For example, in Oracle database, every SQL result set contains a hidden column called ROWNUM. We can just explicitly select ROWNUM to get sequence numbers.

How to select first 5 records from a table?
This question, often asked in many interviews, does not make any sense to me. The problem here is how do you define which record is first and which is second. Which record is retrieved first from the database is not deterministic. It depends on many uncontrollable factors such as how database works at that moment of execution etc. So the question should really be – “how to select any 5 records from the table?” But whatever it is, here is the solution:
In Oracle,

```
SELECT *
FROM EMP
WHERE ROWNUM <= 5;
```
In SQL Server,

```
SELECT TOP 5 * FROM EMP;
```
Generic solution,

I believe a generic solution can be devised for this problem if and only if there exists at least one distinct column in the table. For example, in our EMP table ID is distinct. We can use that distinct column in the below way to come up with a generic solution of this question that does not require database specific functions such as ROWNUM, TOP etc.

```
SELECT  name
FROM EMPLOYEE o
WHERE (SELECT count(*) FROM EMPLOYEE i WHERE i.name < o.name) < 5
```
name
Inno
Anno
Darl
Meme
Bhuti

I have taken “name” column in the above example since “name” is happened to be unique in this table. I could very well take ID column as well.

In this example, if the chosen column was not distinct, we would have got more than 5 records returned in our output.

Do you have a better solution to this problem? If yes, post your solution in the comment.

What is the difference between ROWNUM pseudo column and ROW_NUMBER() function?
ROWNUM is a pseudo column present in Oracle database returned result set prior to ORDER BY being evaluated. So ORDER BY ROWNUM does not work.

ROW_NUMBER() is an analytical function which is used in conjunction to OVER() clause wherein we can specify ORDER BY and also PARTITION BY columns.

Suppose if you want to generate the row numbers in the order of ascending employee salaries for example, ROWNUM will not work. But you may use ROW_NUMBER() OVER() like shown below:

```
SELECT name, sal, row_number() over(order by sal desc) rownum_by_sal
FROM EMPLOYEE o
```
name Sal ROWNUM_BY_SAL
Hash 100 1
Robo 100 2
Anno 80 3

Darl 80 4
Tomiti 70 5
Pete 70 6
Bhuti 60 7
Meme 60 8
Inno 50 9
Privy 50 10

What are the differences among ROWNUM, RANK and DENSE_RANK?

ROW_NUMBER assigns contiguous, unique numbers from 1.. N to a result set.

RANK does not assign unique numbers—nor does it assign contiguous numbers. If two records tie for second place, no record will be assigned the 3rd rank as no one came in third, according to RANK. See below:

SELECT name, sal, rank() over(order by sal desc) rank_by_sal
FROM EMPLOYEE o

name Sal RANK_BY_SAL
Hash 100 1
Robo 100 1
Anno 80 3
Darl 80 3
Tomiti 70 5
Pete 70 5
Bhuti 60 7
Meme 60 7
Inno 50 9
Privy 50 9

DENSE_RANK, like RANK, does not assign unique numbers, but it does assign contiguous numbers. Even though two records tied for second place, there is a third-place record. See below:

SELECT name, sal, dense_rank() over(order by sal desc) dense_rank_by_sal
FROM EMPLOYEE o

name Sal DENSE_RANK_BY_SAL
Hash 100 1
Robo 100 1
Anno 80 2
Darl 80 2
Tomiti 70 3
Pete 70 3
Bhuti 60 4
Meme 60 4
Inno 50 5
Privy 50 5

This article is to be extended ...