**Question 1: SQL Query to find second highest salary of Employee**

Answer: There are many ways to find second highest salary of Employee in SQL, you can either use SQL Join or Subquery to solve this problem. Here is SQL query using Subquery:

select MAX(Salary) from Employee WHERE Salary NOT IN (select MAX(Salary) from Employee );

See [How to find second highest salary in SQL](http://javarevisited.blogspot.com/2012/12/how-to-find-second-highest-or-maximum-salary-sql.html) for more ways to solve this problem.

**Question 2: SQL Query to find Max Salary from each department.**

Answer: You can find the maximum salary for each department by grouping all records by DeptId and then using MAX() function to calculate maximum salary in each group or each department.

SELECT DeptID, MAX(Salary) FROM Employee GROUP BY DeptID.

These questions become more interesting if Interviewer will ask you to print department name instead of department id, in that case, you need to join Employee table with Department using foreign key DeptID, make sure you do LEFT or RIGHT OUTER JOIN to include departments without any employee as well.  Here is the query

SELECT DeptName, MAX(Salary) FROM Employee e RIGHT JOIN Department d ON e.DeptId = d.DeptID GROUP BY DeptName;

In this query, we have used RIGHT OUTER JOIN because we need the name of the department from Department table which is on the right side of JOIN clause, even if there is no reference of dept\_id on Employee table.  **Question 3: Write SQL Query to display the current date.**

Answer: SQL has built-in function called GetDate() which returns the current timestamp. This will work in Microsoft SQL Server, other vendors like Oracle and MySQL also has equivalent functions.

SELECT GetDate();

**Question 4: Write an SQL Query to check whether date passed to Query is the date of given format or not**.

Answer: SQL has IsDate() function which is used to check passed value is a date or not of specified format, it returns 1(true) or 0(false) accordingly. Remember ISDATE() is an MSSQL function and it may not work on Oracle, MySQL or any other database but there would be something similar.

SELECT ISDATE('1/08/13') AS "MM/DD/YY";

It will return 0 because passed date is not in correct format.

**Question 5: Write an SQL Query to print the name of the distinct employee whose DOB is between 01/01/1960 to 31/12/1975.**

Answer: This SQL query is tricky, but you can use BETWEEN clause to get all records whose date fall between two dates.

SELECT DISTINCT EmpName FROM Employees WHERE DOB BETWEEN ‘01/01/1960’ AND ‘31/12/1975’;

**Question 6: Write an SQL Query find number of employees according to gender whose DOB is between 01/01/1960 to 31/12/1975.**

Answer : 

SELECT COUNT(\*), sex from Employees WHERE DOB BETWEEN '01/01/1960' AND '31/12/1975' GROUP BY sex;

**Question 7: Write an SQL Query to find an employee whose Salary is equal or greater than 10000**.

Answer : 

SELECT EmpName FROM Employees WHERE Salary>=10000;

**Question 8: Write an SQL Query to find name of employee whose name Start with ‘M’**

Answer : 

SELECT \* FROM Employees WHERE EmpName like 'M%';

**Question 9: find all Employee records containing the word "Joe", regardless of whether it was stored as JOE, Joe, or joe.**

Answer :

SELECT \* from Employees WHERE UPPER(EmpName) like '%JOE%';

**Question 10: Write an SQL Query to find the year from date.**

Answer:  Here is how you can find Year from a Date in SQL Server 2008 

SELECT YEAR(GETDATE()) as "Year";

**Question 11: Write SQL Query to find duplicate rows in a database? and then write SQL query to delete them?**  
Answer: You can use the following query to select distinct records:

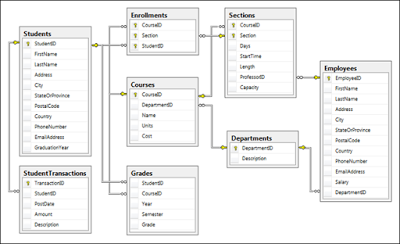
SELECT \* FROM emp a WHERE rowid = (SELECT MAX(rowid) FROM EMP b WHERE a.empno=b.empno)

to Delete:

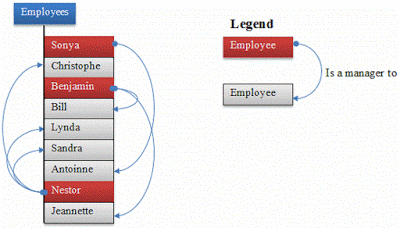
DELETE FROM emp a WHERE rowid != (SELECT MAX(rowid) FROM emp b WHERE a.empno=b.empno);

**Question 12: There is a table which contains two column Student and Marks, you need to find all the students, whose marks are greater than average marks i.e. list of above average students.**  
Answer: This query can be written using subquery as shown below:

SELECT student, marks from table where marks > SELECT AVG(marks) from table)

[](http://1.bp.blogspot.com/-EBP7clmjL1Q/VXrjOw_yTVI/AAAAAAAADAs/n_pQ6j7vkZw/s1600/SQL%2BSchema%2BInterview%2BQuestions.png)

**Question 13: How do you find all employees which are also manager? .**  
You have given a standard employee table with an additional column mgr\_id, which contains employee id of the manager.

[](http://1.bp.blogspot.com/-A8OVkcRMrsM/VXrj9FBPKvI/AAAAAAAADA0/XZvHQHDVaNQ/s1600/Employee%2BManager%2Bquery.gif)

Answer: You need to know about self-join to solve this problem. In Self Join, you can join two instances of the same table to find out additional details as shown below

SELECT e.name, m.name FROM Employee e, Employee m WHERE e.mgr\_id = m.emp\_id;

this will show employee name and manager name in two column e.g.  
  
name  manager\_name  
John   David  
  
One follow-up is to modify this query to include employees which don't have a manager. To solve that, instead of using the inner join, just use left outer join, this will also include employees without managers.  
  
  
  
**Question 14: You have a composite index of three columns, and you only provide the value of two columns in WHERE clause of a select query? Will Index be used for this operation?** For example if Index is on EmpId, EmpFirstName, and EmpSecondName and you write query like

SELECT \* FROM Employee WHERE EmpId=2 and EmpFirstName='Radhe'

If the given two columns are secondary index column then the index will not invoke, but if the given 2 columns contain the primary index(first column while creating index) then the index will invoke. In this case, Index will be used because EmpId and EmpFirstName are primary columns.

Read more: <http://www.java67.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz4J05hmqGL>

**Table Name : Employee**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employee\_id** | **First\_name** | **Last\_name** | **Salary** | **Joining\_date** | **Department** |
| 1 | John | Abraham | 1000000 | 01-JAN-13 12.00.00 AM | Banking |
| 2 | Michael | Clarke | 800000 | 01-JAN-13 12.00.00 AM | Insurance |
| 3 | Roy | Thomas | 700000 | 01-FEB-13 12.00.00 AM | Banking |
| 4 | Tom | Jose | 600000 | 01-FEB-13 12.00.00 AM | Insurance |
| 5 | Jerry | Pinto | 650000 | 01-FEB-13 12.00.00 AM | Insurance |
| 6 | Philip | Mathew | 750000 | 01-JAN-13 12.00.00 AM | Services |
| 7 | TestName1 | 123 | 650000 | 01-JAN-13 12.00.00 AM | Services |
| 8 | TestName2 | Lname% | 600000 | 01-FEB-13 12.00.00 AM | Insurance |

**Table Name : Incentives**

|  |  |  |
| --- | --- | --- |
| **Employee\_ref\_id** | **Incentive\_date** | **Incentive\_amount** |
| 1 | 01-FEB-13 | 5000 |
| 2 | 01-FEB-13 | 3000 |
| 3 | 01-FEB-13 | 4000 |
| 1 | 01-JAN-13 | 4500 |
| 2 | 01-JAN-13 | 3500 |

**SQL Queries Interview Questions and Answers on "SQL Select"**

**1. Get all employee details from the employee table**

Select \* from employee

**2. Get First\_Name,Last\_Name from employee table**

Select first\_name, Last\_Name from employee

**3. Get First\_Name from employee table using alias name “Employee Name”**

Select first\_name Employee Name from employee

**4. Get First\_Name from employee table in upper case**

Select upper(FIRST\_NAME) from EMPLOYEE

**5. Get First\_Name from employee table in lower case**

Select lower(FIRST\_NAME) from EMPLOYEE

**6. Get unique DEPARTMENT from employee table**

select distinct DEPARTMENT from EMPLOYEE

**Don't Miss** - [SQL and Database theory Interview Questions](http://a4academics.com/interview-questions/53-database-and-sql/411-sql-interview-questions-and-answers-database)

**7. Select first 3 characters of FIRST\_NAME from EMPLOYEE**

**Oracle Equivalent of SQL Server SUBSTRING is SUBSTR**, Query : select substr(FIRST\_NAME,0,3) from employee  
 **SQL Server Equivalent of Oracle SUBSTR is SUBSTRING**, Query : select substring(FIRST\_NAME,1,3) from employee  
 **MySQL Server Equivalent of Oracle SUBSTR is SUBSTRING**. In MySQL start position is 1, Query : select substring(FIRST\_NAME,1,3) from employee

**8. Get position of 'o' in name 'John' from employee table**

**Oracle Equivalent of SQL Server CHARINDEX is INSTR**, Query : Select instr(FIRST\_NAME,'o') from employee where first\_name='John'  
 **SQL Server Equivalent of Oracle INSTR is CHARINDEX**, Query: Select CHARINDEX('o',FIRST\_NAME,0) from employee where first\_name='John'  
 **MySQL Server Equivalent of Oracle INSTR is LOCATE**, Query: Select LOCATE('o',FIRST\_NAME) from employee where first\_name='John'

**9. Get FIRST\_NAME from employee table after removing white spaces from right side**

select RTRIM(FIRST\_NAME) from employee

**10. Get FIRST\_NAME from employee table after removing white spaces from left side**

select LTRIM(FIRST\_NAME) from employee

**11. Get length of FIRST\_NAME from employee table**

**Oracle,MYSQL Equivalent of SQL Server Len is Length** , Query :select length(FIRST\_NAME) from employee  
 **SQL Server Equivalent of Oracle,MYSQL Length is Len**, Query :select len(FIRST\_NAME) from employee

**12. Get First\_Name from employee table after replacing 'o' with '$'**

select REPLACE(FIRST\_NAME,'o','$') from employee

**13. Get First\_Name and Last\_Name as single column from employee table separated by a '\_'**

**Oracle Equivalent of MySQL concat is '||'**, Query : Select FIRST\_NAME|| '\_' ||LAST\_NAME from EMPLOYEE  
 **SQL Server Equivalent of MySQL concat is '+'**, Query : Select FIRST\_NAME + '\_' +LAST\_NAME from EMPLOYEE  
 **MySQL Equivalent of Oracle '||' is concat**, Query : Select concat(FIRST\_NAME,'\_',LAST\_NAME) from EMPLOYEE

**14. Get FIRST\_NAME ,Joining year,Joining Month and Joining Date from employee table**

**SQL Queries in Oracle**, Select FIRST\_NAME, to\_char(joining\_date,'YYYY') JoinYear , to\_char(joining\_date,'Mon'), to\_char(joining\_date,'dd') from EMPLOYEE  
 **SQL Queries in SQL Server**, select SUBSTRING (convert(varchar,joining\_date,103),7,4) , SUBSTRING (convert(varchar,joining\_date,100),1,3) , SUBSTRING (convert(varchar,joining\_date,100),5,2) from EMPLOYEE  
 **SQL Queries in MySQL**, select year(joining\_date),month(joining\_date), DAY(joining\_date) from EMPLOYEE

**15. Get all employee details from the employee table order by First\_Name Ascending**

Select \* from employee order by FIRST\_NAME asc

**16. Get all employee details from the employee table order by First\_Name descending**

Select \* from employee order by FIRST\_NAME desc

**17. Get all employee details from the employee table order by First\_Name Ascending and Salary descending**

Select \* from employee order by FIRST\_NAME asc,SALARY desc

**"SQL Where Condition" Interview Questions**

**18. Get employee details from employee table whose employee name is “John”**

Select \* from EMPLOYEE where FIRST\_NAME='John'

**19. Get employee details from employee table whose employee name are “John” and “Roy”**

Select \* from EMPLOYEE where FIRST\_NAME in ('John','Roy')

**20. Get employee details from employee table whose employee name are not “John” and “Roy”**

Select \* from EMPLOYEE where FIRST\_NAME not in ('John','Roy')

**"SQL Wild Card Search" Interview Questions**

**21. Get employee details from employee table whose first name starts with 'J'**

Select \* from EMPLOYEE where FIRST\_NAME like 'J%'

**22. Get employee details from employee table whose first name contains 'o'**

Select \* from EMPLOYEE where FIRST\_NAME like '%o%'

**23. Get employee details from employee table whose first name ends with 'n'**

Select \* from EMPLOYEE where FIRST\_NAME like '%n'

**"SQL Pattern Matching" Interview Questions**

**24. Get employee details from employee table whose first name ends with 'n' and name contains 4 letters**

Select \* from EMPLOYEE where FIRST\_NAME like '\_\_\_n' (Underscores)

**25. Get employee details from employee table whose first name starts with 'J' and name contains 4 letters**

Select \* from EMPLOYEE where FIRST\_NAME like 'J\_\_\_' (Underscores)

**26. Get employee details from employee table whose Salary greater than 600000**

Select \* from EMPLOYEE where Salary >600000

**27. Get employee details from employee table whose Salary less than 800000**

Select \* from EMPLOYEE where Salary <800000

**28. Get employee details from employee table whose Salary between 500000 and 800000**

Select \* from EMPLOYEE where Salary between 500000 and 800000

**29. Get employee details from employee table whose name is 'John' and 'Michael'**

Select \* from EMPLOYEE where FIRST\_NAME in ('John','Michael')

**30. Get employee details from employee table whose joining year is “2013”**

**SQL Queries in Oracle**, Select \* from EMPLOYEE where to\_char(joining\_date,'YYYY')='2013'  
  
**SQL Queries in SQL Server**, Select \* from EMPLOYEE where SUBSTRING(convert(varchar,joining\_date,103),7,4)='2013'  
  
**SQL Queries in MySQL**, Select \* from EMPLOYEE where year(joining\_date)='2013'

**31. Get employee details from employee table whose joining month is “January”**

**SQL Queries in Oracle**, Select \* from EMPLOYEE where to\_char(joining\_date,'MM')='01' or Select \* from EMPLOYEE where to\_char(joining\_date,'Mon')='Jan'  
  
**SQL Queries in SQL Server**, Select \* from EMPLOYEE where SUBSTRING(convert(varchar,joining\_date,100),1,3)='Jan'  
  
**SQL Queries in MySQL**, Select \* from EMPLOYEE where month(joining\_date)='01'

**32. Get employee details from employee table who joined before January 1st 2013**

**SQL Queries in Oracle**, Select \* from EMPLOYEE where JOINING\_DATE <to\_date('01/01/2013','dd/mm/yyyy')  
  
**SQL Queries in SQL Server** (Format - “MM/DD/YYYY”), Select \* from EMPLOYEE where joining\_date <'01/01/2013'  
  
**SQL Queries in MySQL** (Format - “YYYY-DD-MM”), Select \* from EMPLOYEE where joining\_date <'2013-01-01'

**33. Get employee details from employee table who joined after January 31st**

**SQL Queries in Oracle**, Select \* from EMPLOYEE where JOINING\_DATE >to\_date('31/01/2013','dd/mm/yyyy')  
  
**SQL Queries in SQL Server and MySQL** (Format - “MM/DD/YYYY”), Select \* from EMPLOYEE where joining\_date >'01/31/2013'  
  
**SQL Queries in MySQL** (Format - “YYYY-DD-MM”), Select \* from EMPLOYEE where joining\_date >'2013-01-31'

**35. Get Joining Date and Time from employee table**

**SQL Queries in Oracle**, select to\_char(JOINING\_DATE,'dd/mm/yyyy hh:mi:ss') from EMPLOYEE  
  
**SQL Queries in SQL Server**, Select convert(varchar(19),joining\_date,121) from EMPLOYEE  
  
**SQL Queries in MySQL**, Select CONVERT(DATE\_FORMAT(joining\_date,'%Y-%m-%d-%H:%i:00'),DATETIME) from EMPLOYEE

**36. Get Joining Date,Time including milliseconds from employee table**

**SQL Queries in Oracle**, select to\_char(JOINING\_DATE,'dd/mm/yyyy HH:mi:ss.ff') from EMPLOYEE . Column Data Type should be “TimeStamp”  
  
**SQL Queries in SQL Server**, select convert(varchar,joining\_date,121) from EMPLOYEE  
  
**SQL Queries in MySQL**, Select MICROSECOND(joining\_date) from EMPLOYEE

**37. Get difference between JOINING\_DATE and INCENTIVE\_DATE from employee and incentives table**

Select FIRST\_NAME,INCENTIVE\_DATE - JOINING\_DATE from employee a inner join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID

**38. Get database date**

**SQL Queries in Oracle**, select sysdate from dual  
  
**SQL Queries in SQL Server**, select getdate()  
  
**SQL Query in MySQL**, select now()

**39. Get names of employees from employee table who has '%' in Last\_Name. Tip : Escape character for special characters in a query.**

**SQL Queries in Oracle**, Select FIRST\_NAME from employee where Last\_Name like '%?%%'

**SQL Queries in SQL Server**, Select FIRST\_NAME from employee where Last\_Name like '%[%]%'

**SQL Queries in MySQL**, Select FIRST\_NAME from employee where Last\_Name like '%\%%'

**40. Get Last Name from employee table after replacing special character with white space**

**SQL Queries in Oracle**, Select translate(LAST\_NAME,'%',' ') from employee  
  
**SQL Queries in SQL Server and MySQL**, Select REPLACE(LAST\_NAME,'%',' ') from employee

**"SQL Group By Query" Interview Questions and Answers**

**41. Get department,total salary with respect to a department from employee table.**

Select DEPARTMENT,sum(SALARY) Total\_Salary from employee group by department

**42. Get department,total salary with respect to a department from employee table order by total salary descending**

Select DEPARTMENT,sum(SALARY) Total\_Salary from employee group by DEPARTMENT order by Total\_Salary descending

**SQL Queries Interview Questions and Answers on "SQL Mathematical Operations using Group By"**

**43. Get department,no of employees in a department,total salary with respect to a department from employee table order by total salary descending**

Select DEPARTMENT,count(FIRST\_NAME),sum(SALARY) Total\_Salary from employee group by DEPARTMENT order by Total\_Salary descending

**44. Get department wise average salary from employee table order by salary ascending**

select DEPARTMENT,avg(SALARY) AvgSalary from employee group by DEPARTMENT order by AvgSalary asc

**45. Get department wise maximum salary from employee table order by salary ascending**

select DEPARTMENT,max(SALARY) MaxSalary from employee group by DEPARTMENT order by MaxSalary asc

**46. Get department wise minimum salary from employee table order by salary ascending**

select DEPARTMENT,min(SALARY) MinSalary from employee group by DEPARTMENT order by MinSalary asc

**47. Select no of employees joined with respect to year and month from employee table**

**SQL Queries in Oracle**, select to\_char (JOINING\_DATE,'YYYY') Join\_Year,to\_char (JOINING\_DATE,'MM') Join\_Month,count(\*) Total\_Emp from employee group by to\_char (JOINING\_DATE,'YYYY'),to\_char(JOINING\_DATE,'MM')  
  
**SQL Queries in SQL Server**, select datepart (YYYY,JOINING\_DATE) Join\_Year,datepart (MM,JOINING\_DATE) Join\_Month,count(\*) Total\_Emp from employee group by datepart(YYYY,JOINING\_DATE), datepart(MM,JOINING\_DATE)  
  
**SQL Queries in MySQL**, select year (JOINING\_DATE) Join\_Year,month (JOINING\_DATE) Join\_Month,count(\*) Total\_Emp from employee group by year(JOINING\_DATE), month(JOINING\_DATE)

**48. Select department,total salary with respect to a department from employee table where total salary greater than 800000 order by Total\_Salary descending**

Select DEPARTMENT,sum(SALARY) Total\_Salary from employee group by DEPARTMENT having sum(SALARY) >800000 order by Total\_Salary desc

**49. Select employee details from employee table if data exists in incentive table ?**

select \* from EMPLOYEE where exists (select \* from INCENTIVES)

**Explanation** : Here "exists" statement helps us to do the job of If statement. Main query will get executed if the sub query returns at least one row. So we can consider the sub query as "If condition" and the main query as "code block" inside the If condition. We can use any SQL commands (Joins, Group By , having etc) in sub query. This command will be useful in queries which need to detect an event and do some activity.

**50. How to fetch data that are common in two query results ?**

select \* from EMPLOYEE where EMPLOYEE\_ID INTERSECT select \* from EMPLOYEE where EMPLOYEE\_ID < 4

**Explanation** : Here "INTERSECT" command is used to fetch data that are common in 2 queries. In this example, we had taken EMPLOYEE table in both the queries.We can apply INTERSECT command on different tables. The result of the above query will return employee details of "ROY" because, employee id of ROY is 3, and both query results have the information about ROY.

**51. Get Employee ID's of those employees who didn't receive incentives without using sub query ?**

select EMPLOYEE\_ID from EMPLOYEE  
MINUS  
select EMPLOYEE\_REF\_ID from INCENTIVES

**Explanation** : To filter out certain information we use MINUS command. What MINUS Command odes is that, it returns all the results from the first query, that are not part of the second query. In our example, first three employees received the incentives. So query will return employee id's 4 to 8.

**52. Select 20 % of salary from John , 10% of Salary for Roy and for other 15 % of salary from employee table**

SELECT FIRST\_NAME, CASE FIRST\_NAME WHEN 'John' THEN SALARY \* .2 WHEN 'Roy' THEN SALARY \* .10 ELSE SALARY \* .15 END "Deduced\_Amount" FROM EMPLOYEE

**Explanation** : Here, we are using "SQL CASE" statement to achieve the desired results. After case statement, we had to specify the column on which filtering is applied. In our case it is "FIRST\_NAME". And in then condition, specify the name of filter like John, Roy etc. To handle conditions outside our filter, use else block where every one other than John and Roy enters.

**53. Select Banking as 'Bank Dept', Insurance as 'Insurance Dept' and Services as 'Services Dept' from employee table**

SQL Queries in Oracle, SELECT distinct DECODE (DEPARTMENT, 'Banking', 'Bank Dept', 'Insurance', 'Insurance Dept', 'Services', 'Services Dept') FROM EMPLOYEE  
SQL Queries in SQL Server and MySQL, SELECT case DEPARTMENT when 'Banking' then 'Bank Dept' when 'Insurance' then 'Insurance Dept' when 'Services' then 'Services Dept' end FROM EMPLOYEE

**Explanation** : Here "DECODE" keyword is used to specify the alias name. In oracle we had specify, Column Name followed by Actual Name and Alias Name as arguments. In SQL Server and MySQL, we can use the earlier switch case statements for alias names.

**54. Delete employee data from employee table who got incentives in incentive table**

delete from EMPLOYEE where EMPLOYEE\_ID in (select EMPLOYEE\_REF\_ID from INCENTIVES)

**Explanation** : Trick about this question is that we can't delete data from a table based on some condition in another table by joining them. Here to delete multiple entries from EMPLOYEE table, we need to use Subquery. Entries will get deleted based on the result of Subquery.

**55. Insert into employee table Last Name with " ' " (Single Quote - Special Character)**

Tip - Use another single quote before special character  
Insert into employee (LAST\_NAME) values ('Test''')

**56. Select Last Name from employee table which contain only numbers**

Select \* from EMPLOYEE where lower(LAST\_NAME)=upper(LAST\_NAME)

**Explanation** : In order to achieve the desired result, we use "ASCII" property of the database. If we get results for a column using Lower and Upper commands, ASCII of both results will be same for numbers. If there is any alphabets in the column, results will differ.

**57. Write a query to rank employees based on their incentives for a month**

select FIRST\_NAME,INCENTIVE\_AMOUNT,DENSE\_RANK() OVER (PARTITION BY INCENTIVE\_DATE ORDER BY INCENTIVE\_AMOUNT DESC) AS Rank from EMPLOYEE a, INCENTIVES b where a.EMPLOYEE\_ID=b.EMPLOYEE\_REF\_ID

**Explanation** : In order to rank employees based on their rank for a month, "DENSE\_RANK" keyword is used. Here partition by keyword helps us to sort the column with which filtering is done. Rank is provided to the column specified in the order by statement. The above query ranks employees with respect to their incentives for a given month.

**58. Update incentive table where employee name is 'John'**

update INCENTIVES set INCENTIVE\_AMOUNT='9000' where EMPLOYEE\_REF\_ID=(select EMPLOYEE\_ID from EMPLOYEE where FIRST\_NAME='John' )

**Explanation** : We need to join Employee and Incentive Table for updating the incentive amount. But for update statement joining query wont work. We need to use sub query to update the data in the incentive table. SQL Query is as shown below.

**59. Select first\_name, incentive amount from employee and incentives table for those employees who have incentives**

Select FIRST\_NAME,INCENTIVE\_AMOUNT from employee a inner join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID

**60. Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000**

Select FIRST\_NAME,INCENTIVE\_AMOUNT from employee a inner join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID and INCENTIVE\_AMOUNT >3000

**61. Select first\_name, incentive amount from employee and incentives table for all employes even if they didn't get incentives**

Select FIRST\_NAME,INCENTIVE\_AMOUNT from employee a left join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID

**62. Select first\_name, incentive amount from employee and incentives table for all employees even if they didn't get incentives and set incentive amount as 0 for those employees who didn't get incentives.**

**SQL Queries in Oracle**, Select FIRST\_NAME,nvl(INCENTIVE\_AMOUNT,0) from employee a left join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID  
  
**SQL Queries in SQL Server**, Select FIRST\_NAME, ISNULL(INCENTIVE\_AMOUNT,0) from employee a left join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID  
  
**SQL Queries in MySQL**, Select FIRST\_NAME, IFNULL(INCENTIVE\_AMOUNT,0) from employee a left join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID

**63. Select first\_name, incentive amount from employee and incentives table for all employees who got incentives using left join**

**SQL Queries in Oracle**, Select FIRST\_NAME,nvl(INCENTIVE\_AMOUNT,0) from employee a right join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID  
  
**SQL Queries in SQL Server**, Select FIRST\_NAME, isnull(INCENTIVE\_AMOUNT,0) from employee a right join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID  
  
**SQL Queries in MySQL**, Select FIRST\_NAME, IFNULL(INCENTIVE\_AMOUNT,0) from employee a right join incentives B on A.EMPLOYEE\_ID=B.EMPLOYEE\_REF\_ID

**64. Select max incentive with respect to employee from employee and incentives table using sub query**

**SQL Queries in Oracle**, select DEPARTMENT,(select nvl(max(INCENTIVE\_AMOUNT),0) from INCENTIVES where EMPLOYEE\_REF\_ID=EMPLOYEE\_ID) Max\_incentive from EMPLOYEE  
  
**SQL Queries in SQL Server**, select DEPARTMENT,(select ISNULL(max(INCENTIVE\_AMOUNT),0) from INCENTIVES where EMPLOYEE\_REF\_ID=EMPLOYEE\_ID) Max\_incentive from EMPLOYEE  
  
**SQL Queries in SQL Server**, select DEPARTMENT,(select IFNULL (max(INCENTIVE\_AMOUNT),0) from INCENTIVES where EMPLOYEE\_REF\_ID=EMPLOYEE\_ID) Max\_incentive from EMPLOYEE

**"Top N Salary" SQL Interview Questions and Answers**

**65. Select TOP 2 salary from employee table**

**SQL Queries in Oracle**, select \* from (select \* from employee order by SALARY desc) where rownum <3  
  
**SQL Queries in SQL Server**, select top 2 \* from employee order by salary desc  
  
**SQL Queries in MySQL**, select \* from employee order by salary desc limit 2

**66. Select TOP N salary from employee table**

**SQL Queries in Oracle**, select \* from (select \* from employee order by SALARY desc) where rownum <N + 1  
  
**SQL Queries in SQL Server**, select top N \* from employee  
  
**SQL Queries in MySQL**, select \* from employee order by salary desc limit N

**67. Select 2nd Highest salary from employee table**

**SQL Queries in Oracle**, select min(salary) from (select \* from (select \* from employee order by SALARY desc) where rownum <3)  
  
**SQL Queries in SQL Server**, select min(SALARY) from (select top 2 \* from employee) a  
  
**SQL Queries in MySQL**, select min(SALARY) from (select \* from employee order by salary desc limit 2) a

**68. Select Nth Highest salary from employee table**

**SQL Queries in Oracle**, select min(salary) from (select \* from (select \* from employee order by SALARY desc) where rownum <N + 1)  
  
**SQL Queries in SQL Server**, select min(SALARY) from (select top N \* from employee) a  
  
**SQL Queries in MySQL**, select min(SALARY) from (select \* from employee order by salary desc limit N) a

**"SQL Union" Query Interview Questions**

**69. Select First\_Name,LAST\_NAME from employee table as separate rows**

select FIRST\_NAME from EMPLOYEE union select LAST\_NAME from EMPLOYEE

**70. What is the difference between UNION and UNION ALL ?**

Both UNION and UNION ALL is used to select information from structurally similar tables. That means corresponding columns specified in the union should have same data type. For example, in the above query, if FIRST\_NAME is DOUBLE and LAST\_NAME is STRING above query wont work. Since the data type of both the columns are VARCHAR, union is made possible. Difference between UNION and UNION ALL is that , UNION query return only distinct values.

**71. Write create table syntax for employee table**

Oracle -CREATE TABLE EMPLOYEE (  
EMPLOYEE\_ID NUMBER,  
FIRST\_NAME VARCHAR2(20 BYTE),  
LAST\_NAME VARCHAR2(20 BYTE),  
SALARY FLOAT(126),  
JOINING\_DATE TIMESTAMP (6) DEFAULT sysdate,  
DEPARTMENT VARCHAR2(30 BYTE) )  
SQL Server -CREATE TABLE EMPLOYEE(  
EMPLOYEE\_ID int NOT NULL,  
FIRST\_NAME varchar(50) NULL,  
LAST\_NAME varchar(50) NULL,  
SALARY decimal(18, 0) NULL,  
JOINING\_DATE datetime2(7) default getdate(),  
DEPARTMENT varchar(50) NULL)

**72. Write syntax to delete table employee**

DROP table employee;

**73. Write syntax to set EMPLOYEE\_ID as primary key in employee table**

ALTER TABLE EMPLOYEE add CONSTRAINT EMPLOYEE\_PK PRIMARY KEY(EMPLOYEE\_ID)

**74. Write syntax to set 2 fields(EMPLOYEE\_ID,FIRST\_NAME) as primary key in employee table**

ALTER TABLE EMPLOYEE add CONSTRAINT EMPLOYEE\_PK PRIMARY KEY(EMPLOYEE\_ID,FIRST\_NAME)

**75. Write syntax to drop primary key on employee table**

Alter TABLE EMPLOYEE drop CONSTRAINT EMPLOYEE\_PK;

**76. Write Sql Syntax to create EMPLOYEE\_REF\_ID in INCENTIVES table as foreign key with respect to EMPLOYEE\_ID in employee table**

ALTER TABLE INCENTIVES ADD CONSTRAINT INCENTIVES\_FK FOREIGN KEY (EMPLOYEE\_REF\_ID) REFERENCES EMPLOYEE(EMPLOYEE\_ID)

**77. Write SQL to drop foreign key on employee table**

ALTER TABLE INCENTIVES drop CONSTRAINT INCENTIVES\_FK;

**78. Write SQL to create Orcale Sequence**

CREATE SEQUENCE EMPLOYEE\_ID\_SEQ START WITH 0 NOMAXVALUE MINVALUE 0 NOCYCLE NOCACHE NOORDER;

**79. Write Sql syntax to create Oracle Trigger before insert of each row in employee table**

CREATE OR REPLACE TRIGGER EMPLOYEE\_ROW\_ID\_TRIGGER  
BEFORE INSERT ON EMPLOYEE FOR EACH ROW  
DECLARE  
seq\_no number(12);  
BEGIN  
select EMPLOYEE\_ID\_SEQ.nextval into seq\_no from dual ;  
:new EMPLOYEE\_ID :=seq\_no;  
END;  
SHOW ERRORS;

**80. Oracle Procedure81. Oracle View**

An example oracle view script is given below  
create view Employee\_Incentive as select FIRST\_NAME,max(INCENTIVE\_AMOUNT) INCENTIVE\_AMOUNT from EMPLOYEE a, INCENTIVES b where a.EMPLOYEE\_ID=b.EMPLOYEE\_REF\_ID group by FIRST\_NAME

**82. Oracle materialized view - Daily Auto Refresh**

CREATE MATERIALIZED VIEW Employee\_Incentive  
REFRESH COMPLETE  
START WITH SYSDATE  
NEXT SYSDATE + 1 AS  
select FIRST\_NAME,INCENTIVE\_DATE,INCENTIVE\_AMOUNT from EMPLOYEE a, INCENTIVES b   
where a.EMPLOYEE\_ID=b.EMPLOYEE\_REF\_ID

**83. Oracle materialized view - Fast Refresh on Commit**

Create materialized view log for fast refresh. Following materialized view script wont get executed if materialized view log doesn't exists  
  
CREATE MATERIALIZED VIEW MAT\_Employee\_Incentive\_Refresh  
BUILD IMMEDIATE  
REFRESH FAST ON COMMIT AS  
select FIRST\_NAME,max(INCENTIVE\_AMOUNT) from EMPLOYEE a, INCENTIVES b  
where a.EMPLOYEE\_ID=b.EMPLOYEE\_REF\_ID group by FIRST\_NAME

**84. What is SQL Injection ?**

SQL Injection is one of the the techniques uses by hackers to hack a website by injecting SQL commands in data fields.

# 24 Essential SQL Interview Questions[\*](https://www.toptal.com/sql/interview-questions#note-1)

What does UNION do? What is the difference between UNION and UNION ALL?

Hide answer

UNION merges the contents of two structurally-compatible tables into a single combined table. The difference between UNION and UNION ALL is that UNION will omit duplicate records whereas UNION ALL will include duplicate records.

It is important to note that the performance of UNION ALL will typically be better than UNION, since UNION requires the server to do the additional work of removing any duplicates. So, in cases where is is certain that there will not be any duplicates, or where having duplicates is not a problem, use of UNION ALL would be recommended for performance reasons.

[Comment](https://www.toptal.com/sql/interview-questions)

List and explain the different types of JOIN clauses supported in ANSI-standard SQL.

View the answer →

Consider the following two query results:

SELECT count(\*) AS total FROM orders;

+-------+

| total |

+-------+

| 100 |

+-------+

SELECT count(\*) AS cust\_123\_total FROM orders WHERE customer\_id = '123';

+----------------+

| cust\_123\_total |

+----------------+

| 15 |

+----------------+

Given the above query results, what will be the result of the query below?

SELECT count(\*) AS cust\_not\_123\_total FROM orders WHERE customer\_id <> '123'

Hide answer

The obvious answer is 85 (i.e, 100 - 15). However, that is not necessarily correct. Specifically, any records with a customer\_id of NULL will not be included in either count (i.e., they won’t be included in cust\_123\_total, nor will they be included in cust\_not\_123\_total). For example, if exactly one of the 100 customers has a NULL customer\_id, the result of the last query will be:

+--------- ----------+

| cust\_not\_123\_total |

+--------------------+

| 84 |

+--------------------+

[Comment](https://www.toptal.com/sql/interview-questions)

What will be the result of the query below? Explain your answer and provide a version that behaves correctly.

select case when null = null then 'Yup' else 'Nope' end as Result;

Hide answer

This query will actually yield “Nope”, seeming to imply that null is not equal to itself! The reason for this is that the proper way to compare a value to null in SQL is with the is operator, not with =.

Accordingly, the correct version of the above query that yields the expected result (i.e., “Yup”) would be as follows:

select case when null is null then 'Yup' else 'Nope' end as Result;

[Comment](https://www.toptal.com/sql/interview-questions)

Given the following tables:

sql> SELECT \* FROM runners;

+----+--------------+

| id | name |

+----+--------------+

| 1 | John Doe |

| 2 | Jane Doe |

| 3 | Alice Jones |

| 4 | Bobby Louis |

| 5 | Lisa Romero |

+----+--------------+

sql> SELECT \* FROM races;

+----+----------------+-----------+

| id | event | winner\_id |

+----+----------------+-----------+

| 1 | 100 meter dash | 2 |

| 2 | 500 meter dash | 3 |

| 3 | cross-country | 2 |

| 4 | triathalon | NULL |

+----+----------------+-----------+

What will be the result of the query below?

SELECT \* FROM runners WHERE id NOT IN (SELECT winner\_id FROM races)

Explain your answer and also provide an alternative version of this query that will avoid the issue that it exposes.

Hide answer

Surprisingly, given the sample data provided, the result of this query will be an empty set. The reason for this is as follows: If the set being evaluated by the SQL NOT IN condition contains any values that are null, then the outer query here will return an empty set, even if there are many runner ids that match winner\_ids in the races table.

Knowing this, a query that avoids this issue would be as follows:

SELECT \* FROM runners WHERE id NOT IN (SELECT winner\_id FROM races WHERE winner\_id IS NOT null)

[Comment](https://www.toptal.com/sql/interview-questions)

Given two tables created and populated as follows:

CREATE TABLE dbo.envelope(id int, user\_id int);

CREATE TABLE dbo.docs(idnum int, pageseq int, doctext varchar(100));

INSERT INTO dbo.envelope VALUES

(1,1),

(2,2),

(3,3);

INSERT INTO dbo.docs(idnum,pageseq) VALUES

(1,5),

(2,6),

(null,0);

What will the result be from the following query:

UPDATE docs SET doctext=pageseq FROM docs INNER JOIN envelope ON envelope.id=docs.idnum

WHERE EXISTS (

SELECT 1 FROM dbo.docs

WHERE id=envelope.id

);

Explain your answer.

Hide answer

The result of the query will be as follows:

idnum pageseq doctext

1 5 5

2 6 6

NULL 0 NULL

The EXISTS clause in the above query is a red herring. It will always be true since ID is not a member of dbo.docs. As such, it will refer to the envelope table comparing itself to itself!

The idnum value of NULL will not be set since the join of NULL will not return a result when attempting a match with any value of envelope.

[Comment](https://www.toptal.com/sql/interview-questions)

What is wrong with this SQL query? Correct it so it executes properly.

SELECT Id, YEAR(BillingDate) AS BillingYear

FROM Invoices

WHERE BillingYear >= 2010;

Hide answer

The expression BillingYear in the WHERE clause is invalid. Even though it is defined as an alias in the SELECT phrase, which appears before the WHERE phrase, the logical processing order of the phrases of the statement is different from the written order. Most programmers are accustomed to code statements being processed generally top-to-bottom or left-to-right, but T-SQL processes phrases in a different order.

The correct query should be:

SELECT Id, YEAR(BillingDate) AS BillingYear

FROM Invoices

WHERE YEAR(BillingDate) >= 2010;

[Comment](https://www.toptal.com/sql/interview-questions)

Given these contents of the Customers table:

Id Name ReferredBy

1 John Doe NULL

2 Jane Smith NULL

3 Anne Jenkins 2

4 Eric Branford NULL

5 Pat Richards 1

6 Alice Barnes 2

Here is a query written to return the list of customers not referred by Jane Smith:

SELECT Name FROM Customers WHERE ReferredBy <> 2;

What will be the result of the query? Why? What would be a better way to write it?

Hide answer

Although there are 4 customers not referred by Jane Smith (including Jane Smith herself), the query will only return one: Pat Richards. All the customers who were referred by nobody at all (and therefore have NULL in their ReferredBy column) don’t show up. But certainly those customers weren’t referred by Jane Smith, and certainly NULL is not equal to 2, so why didn’t they show up?

SQL Server uses three-valued logic, which can be troublesome for programmers accustomed to the more satisfying two-valued logic (TRUE or FALSE) most programming languages use. In most languages, if you were presented with two predicates: ReferredBy = 2 and ReferredBy <> 2, you would expect one of them to be true and one of them to be false, given the same value of ReferredBy. In SQL Server, however, if ReferredBy is NULL, neither of them are true and neither of them are false. Anything compared to NULL evaluates to the third value in three-valued logic: UNKNOWN.

The query should be written:

SELECT Name FROM Customers WHERE ReferredBy IS NULL OR ReferredBy <> 2

Watch out for the following, though!

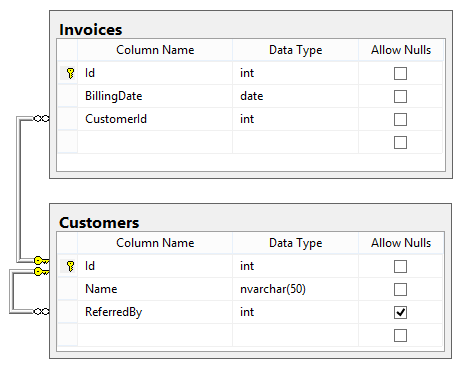
SELECT Name FROM Customers WHERE ReferredBy = NULL OR ReferredBy <> 2

This will return the same faulty set as the original. Why? We already covered that: Anything compared to NULL evaluates to the third value in the three-valued logic: UNKNOWN. That “anything” includes NULL itself! That’s why SQL Server provides the IS NULL and IS NOT NULL operators to specifically check for NULL. Those particular operators will always evaluate to true or false.

Even if a candidate doesn’t have a great amount of experience with SQL Server, diving into the intricacies of three-valued logic in general can give a good indication of whether they have the ability learn it quickly or whether they will struggle with it.

[Comment](https://www.toptal.com/sql/interview-questions)

Considering the database schema displayed in the SQLServer-style diagram below, write a SQL query to return a list of all the invoices. For each invoice, show the Invoice ID, the billing date, the customer’s name, and the name of the customer who referred that customer (if any). The list should be ordered by billing date.



Hide answer

SELECT i.Id, i.BillingDate, c.Name, r.Name AS ReferredByName

FROM Invoices i

JOIN Customers c ON i.CustomerId = c.Id

LEFT JOIN Customers r ON c.ReferredBy = r.Id

ORDER BY i.BillingDate;

This question simply tests the candidate’s ability take a plain-English requirement and write a corresponding SQL query. There is nothing tricky in this one, it just covers the basics:

* Did the candidate remember to use a LEFT JOIN instead of an inner JOIN when joining the customer table for the referring customer name? If not, any invoices by customers not referred by somebody will be left out altogether.
* Did the candidate alias the tables in the JOIN? Most experienced T-SQL programmers always do this, because repeating the full table name each time it needs to be referenced gets tedious quickly. In this case, the query would actually break if at least the Customer table wasn’t aliased, because it is referenced twice in different contexts (once as the table which contains the name of the invoiced customer, and once as the table which contains the name of the referring customer).
* Did the candidate disambiguate the Id and Name columns in the SELECT? Again, this is something most experienced programmers do automatically, whether or not there would be a conflict. And again, in this case there would be a conflict, so the query would break if the candidate neglected to do so.

Note that this query will not return Invoices that do not have an associated Customer. This may be the correct behavior for most cases (e.g., it is guaranteed that every Invoice is associated with a Customer, or unmatched Invoices are not of interest). However, in order to guarantee that all Invoices are returned no matter what, the Invoices table should be joined with Customers using LEFT JOIN:

SELECT i.Id, i.BillingDate, c.Name, r.Name AS ReferredByName

FROM Invoices i

LEFT JOIN Customers c ON i.CustomerId = c.Id

LEFT JOIN Customers r ON c.ReferredBy = r.Id

ORDER BY i.BillingDate;

[Comment](https://www.toptal.com/sql/interview-questions)

Assume a schema of Emp ( Id, Name, DeptId ) , Dept ( Id, Name).

If there are 10 records in the Emp table and 5 records in the Dept table, how many rows will be displayed in the result of the following SQL query:

Select \* From Emp, Dept

Explain your answer.

Hide answer

The query will result in 50 rows as a “cartesian product” or “cross join”, which is the default whenever the ‘where’ clause is omitted.

[Comment](https://www.toptal.com/sql/interview-questions)

Given a table SALARIES, such as the one below, that has m = male and f = female values. Swap all f and m values (i.e., change all f values to m and vice versa) with a single update query and no intermediate temp table.

Id Name Sex Salary

1 A m 2500

2 B f 1500

3 C m 5500

4 D f 500

Hide answer

UPDATE SALARIES SET sex = CASE sex WHEN 'm' THEN 'f' ELSE 'm' END

[Comment](https://www.toptal.com/sql/interview-questions)

Given two tables created as follows

create table test\_a(id numeric);

create table test\_b(id numeric);

insert into test\_a(id) values

(10),

(20),

(30),

(40),

(50);

insert into test\_b(id) values

(10),

(30),

(50);

Write a query to fetch values in table test\_a that are and not in test\_b **without** using the NOT keyword.

Hide answer

In SQL Server, PostgreSQL, and SQLite, this can be done using the [except](https://en.wikipedia.org/wiki/Set_operations_%28SQL%29#EXCEPT_operator) keyword as follows:

select \* from test\_a

except

select \* from test\_b;

In Oracle, the [minus](http://www.techonthenet.com/sql/minus.php) keyword is used instead.

MySQL does not support the except function, so it is necessary to use not in.

[Comment](https://www.toptal.com/sql/interview-questions)

Given a table TBL with a field Nmbr that has rows with the following values:

1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1

Write a query to add 2 where Nmbr is 0 and add 3 where Nmbr is 1.

Hide answer

This can be done as follows:

update TBL set Nmbr = case when Nmbr > 0 then Nmbr+3 else Nmbr+2 end;

[Comment](https://www.toptal.com/sql/interview-questions)

Write a SQL query to find the 10th highest employee salary from an Employee table. Explain your answer.

(Note: You may assume that there are at least 10 records in the Employee table.)

Hide answer

This can be done as follows:

SELECT TOP (1) Salary FROM

(

SELECT DISTINCT TOP (10) Salary FROM Employee ORDER BY Salary DESC

) AS Emp ORDER BY Salary

This works as follows:

First, the SELECT DISTINCT TOP (10) Salary FROM Employee ORDER BY Salary DESC query will select the top 10 salaried employees in the table. However, those salaries will be listed in descending order. That was necessary for the first query to work, but now picking the top 1 from that list will give you the highest salary not the the 10th highest salary.

Therefore, the second query reorders the 10 records in ascending order (which the default sort order) and then selects the top record (which will now be the lowest of those 10 salaries).

Not all databases support the TOP keyword. For example, MySQL and PostreSQL use the LIMIT keyword, as follows:

SELECT Salary FROM

(

SELECT DISTINCT Salary FROM Employee ORDER BY Salary DESC LIMIT 10

) AS Emp ORDER BY Salary LIMIT 1;

[Comment](https://www.toptal.com/sql/interview-questions)

Write a SQL query using UNION ALL (**not** UNION) that uses the WHERE clause to eliminate duplicates. Why might you want to do this?

Hide answer

You can avoid duplicates using UNION ALL and still run much faster than UNION DISTINCT (which is actually same as UNION) by running a query like this:

SELECT \* FROM mytable WHERE a=X UNION ALL SELECT \* FROM mytable WHERE b=Y AND a!=X

The key is the AND a!=X part. This gives you the benefits of the UNION (a.k.a., UNION DISTINCT) command, while avoiding much of its performance hit.

[Comment](https://www.toptal.com/sql/interview-questions)

Given the following tables:

SELECT \* FROM users;

user\_id username

1 John Doe

2 Jane Don

3 Alice Jones

4 Lisa Romero

SELECT \* FROM training\_details;

user\_training\_id user\_id training\_id training\_date

1 1 1 "2015-08-02"

2 2 1 "2015-08-03"

3 3 2 "2015-08-02"

4 4 2 "2015-08-04"

5 2 2 "2015-08-03"

6 1 1 "2015-08-02"

7 3 2 "2015-08-04"

8 4 3 "2015-08-03"

9 1 4 "2015-08-03"

10 3 1 "2015-08-02"

11 4 2 "2015-08-04"

12 3 2 "2015-08-02"

13 1 1 "2015-08-02"

14 4 3 "2015-08-03"

Write a query to to get the list of users who took the a training lesson more than once in the same day, grouped by user and training lesson, each ordered from the most recent lesson date to oldest date.

Hide answer

SELECT

u.user\_id,

username,

training\_id,

training\_date,

count( user\_training\_id ) AS count

FROM users u JOIN training\_details t ON t.user\_id = u.user\_id

GROUP BY user\_id,

training\_id,

training\_date

HAVING count( user\_training\_id ) > 1

ORDER BY training\_date DESC;

user\_id username training\_id training\_date count

4 Lisa Romero 2 August, 04 2015 00:00:00 2

4 Lisa Romero 3 August, 03 2015 00:00:00 2

1 John Doe 1 August, 02 2015 00:00:00 3

3 Alice Jones 2 August, 02 2015 00:00:00 2

[Comment](https://www.toptal.com/sql/interview-questions)

What is an execution plan? When would you use it? How would you view the execution plan?

Hide answer

An execution plan is basically a road map that graphically or textually shows the data retrieval methods chosen by the SQL server’s query optimizer for a stored procedure or ad hoc query. Execution plans are very useful for helping a developer understand and analyze the performance characteristics of a query or stored procedure, since the plan is used to execute the query or stored procedure.

In many SQL systems, a textual execution plan can be obtained using a keyword such as EXPLAIN, and visual representations can often be obtained as well. In Microsoft SQL Server, the Query Analyzer has an option called “Show Execution Plan” (located on the Query drop down menu). If this option is turned on, it will display query execution plans in a separate window when a query is run.

[Comment](https://www.toptal.com/sql/interview-questions)

List and explain each of the ACID properties that collectively guarantee that database transactions are processed reliably.

Hide answer

**ACID (Atomicity, Consistency, Isolation, Durability)** is a set of properties that guarantee that database transactions are processed reliably. They are defined as follows:

* **Atomicity.** Atomicity requires that each transaction be “all or nothing”: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes.
* **Consistency.** The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
* **Isolation.** The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method (i.e. if it uses strict - as opposed to relaxed - serializability), the effects of an incomplete transaction might not even be visible to another transaction.
* **Durability.** Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

[Comment](https://www.toptal.com/sql/interview-questions)

What is a key difference between Truncate and Delete?

Hide answer

Truncate is used to delete table content and the action can **not** be rolled back, whereas Delete is used to delete one or more rows in the table and **can** be rolled back.

[Comment](https://www.toptal.com/sql/interview-questions)

Given a table dbo.users where the column user\_id is a unique identifier, how can you efficiently select the first 100 odd user\_id values from the table?

(Assume the table contains well over 100 records with odd user\_id values.)

Hide answer

SELECT TOP 100 user\_id FROM dbo.users WHERE user\_id % 2 = 1 ORDER BY user\_id

[Comment](https://www.toptal.com/sql/interview-questions)

How can you select all the even number records from a table? All the odd number records?

Hide answer

To select all the **even** number records from a table:

Select \* from table where id % 2 = 0

To select all the **odd** number records from a table:

Select \* from table where id % 2 != 0

[Comment](https://www.toptal.com/sql/interview-questions)

What are the NVL and the NVL2 functions in SQL? How do they differ?

Hide answer

Both the NVL(exp1, exp2) and NVL2(exp1, exp2, exp3) functions check the value exp1 to see if it is null.

With the NVL(exp1, exp2) function, if exp1 is not null, then the value of exp1 is returned; otherwise, the value of exp2 is returned, but case to the same data type as that of exp1.

With the NVL2(exp1, exp2, exp3) function, if exp1 is not null, then exp2 is returned; otherwise, the value of exp3 is returned.

[Comment](https://www.toptal.com/sql/interview-questions)

What is the difference between the RANK() and DENSE\_RANK() functions? Provide an example.

Hide answer

The only difference between the RANK() and DENSE\_RANK() functions is in cases where there is a “tie”; i.e., in cases where multiple values in a set have the same ranking. In such cases, RANK() will assign non-consecutive “ranks” to the values in the set (resulting in gaps between the integer ranking values when there is a tie), whereas DENSE\_RANK() will assign consecutive ranks to the values in the set (so there will be no gaps between the integer ranking values in the case of a tie).

For example, consider the set {25, 25, 50, 75, 75, 100}. For such a set, RANK() will return {1, 1, 3, 4, 4, 6} (note that the values 2 and 5 are skipped), whereas DENSE\_RANK() will return {1,1,2,3,3,4}.

[Comment](https://www.toptal.com/sql/interview-questions)

What is the difference between the WHERE and HAVING clauses?

Hide answer

When GROUP BY is not used, the WHERE and HAVING clauses are essentially equivalent.

However, when GROUP BY **is** used:

* The WHERE clause is used to filter records from a result. The filtering occurs before any groupings are made.
* The HAVING clause is used to filter values from a group (i.e., to check conditions after aggregation into groups has been performed).

# 50 Popular SQL Interview Questions for Testers

In this tutorial, we have listed the **50 most popularly asked SQL interview questions for testers.**

Below are the most common and useful SQL interview questions for testers so that you can prepare for interview while learning SQL.

### **SQL Interview Questions & Answers**

**Also read =>** [**All about Database Testing**](http://www.softwaretestinghelp.com/database-testing-process/)

**Q#1. What does SQL stand for?**  
**Ans.** SQL stands for [Structured Query Language](http://en.wikipedia.org/wiki/SQL).

**Q#2. How to select all records from the table?**   
**Ans.** To select all the records from the table we need to use following syntax:

Select \* from table\_name;

**Q#3. Define join and name different type of joins?**  
**Ans.** Join keyword is used to fetch data from related two or more tables. It returns rows where there is at least one match in both the tables included in join. [Read more here](http://www.w3schools.com/sql/sql_join.asp).  
Type of joins are-

1. Right Join
2. Outer Join
3. Full Join
4. Cross Join
5. Self Join.

**Q#4. What is the syntax to add record to a table?**  
**Ans.** To add record in a table INSERT syntax is used.

Ex: INSERT into table\_name VALUES (value1, value2..);

**Q#5. How do you add a column to a table?**  
**Ans.** To add another column in the table following command has been used.

ALTER TABLE table\_name ADD (column\_name);

**Q#6. Define SQL Delete statement.**  
**Ans.** Delete is used to delete a row or rows from a table based on the specified condition.  
Basic syntax is as follows:

DELETE FROM table\_name

WHERE <Condition>

**Q#7. Define COMMIT ?**  
**Ans.** COMMIT saves all changes made by DML statements.

**Q#8. What is a primary key?**  
**Ans.** A Primary key is column whose values uniquely identify every row in a table. Primary key values can never be reused.

**Q#9. What are foreign keys?**  
**Ans.** When a one table’s primary key field is added to related tables in order to create the common field which relates the two tables, it called a foreign key in other tables.  
Foreign Key constraints enforce referential integrity.

**Q#10. What is CHECK Constraint?**  
**Ans.** A CHECK constraint is used to limit the values or type of data that can be stored in a column. They are used to enforce domain integrity.

**Q#11. Is it possible for a table to have more than one foreign key?**  
**Ans.** Yes, a table can have many foreign keys and only one primary key.

**Q#12. What are the possible values for BOOLEAN data field.**  
**Ans.** For a BOOLEAN data field two values are possible: -1(true) and 0(false).

**Q#13. What is a stored procedure?**  
**Ans.** A stored procedure is a set of SQL queries which can take input and send back output.

**Q#14. What is identity in SQL?**  
**Ans.** An identity column in the SQL automatically generates numeric values. We can defined a start and increment value of identity column.

**Q#15. What is Normalization?**  
**Ans.** The process of table design to minimize the data redundancy is called normalization. We need to divide a database into two or more table and define relationships between them.

**Q#16. What is Trigger?**  
**Ans.** Trigger allows us to execute a batch of SQL code when a table event occurs (Insert, update or delete command executed against a specific table)

**Q#17. How to select random rows from a table?**  
**Ans.** Using SAMPLE clause we can select random rows.

Example:  
SELECT \* FROM table\_name SAMPLE(10);

**Q#18. Which TCP/IP port does SQL Server run?**  
**Ans.** By default SQL Server runs on port 1433.

**Q#19. Write a SQL SELECT query that only returns each name only once from a table?**  
**Ans.** To get the each name only once, we need to use the DISTINCT keyword.

SELECT DISTINCT name FROM table\_name;

**Q#20. Explain DML and DDL?**  
**Ans.** DML stands for Data Manipulation Language. INSERT, UPDATE and DELETE  are DML statements.

DDL stands for Data Definition Language. CREATE ,ALTER, DROP, RENAME are DDL statements.

**Q#21. Can we rename a column in the output of SQL query?**  
**Ans.** Yes using the following syntax we can do this.

SELECT column\_name AS new\_name FROM table\_name;

**Q#22. Give the order of SQL SELECT ?**  
**Ans.** Order of SQL SELECT clauses is: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY. Only the SELECT and FROM clause are mandatory.

**Q#23. Suppose a Student column has two columns, Name and Marks. How to get name and marks of top three students.**  
**Ans.** SELECT Name, Marks FROM Student s1 where 3 <= (SELECT COUNT(\*) FROM Students s2 WHERE s1.marks = s2.marks)

**Q#24. What is SQL comments?**  
**Ans.** SQL comments can be put by two consecutive hyphens (–).

**Q#25. Difference between TRUNCATE, DELETE and DROP commands?**  
**Ans.** DELETE removes some or all rows from a table based on the condition. It can be rolled back.

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

TRUNCATE removes ALL rows from a table by de-allocating the memory pages. The operation cannot be rolled back

DROP command removes a table from the database completely.

**Q#26. What are the properties of a transaction?**  
**Ans.** Generally these properties are referred as ACID properties. They are:

1. Atomicity
2. Consistency
3. Isolation
4. Durability.

**Q#27. What do you mean by ROWID ?**   
**Ans.** It’s a 18 character long pseudo column attached with each row of a table.

**Q#28. Define UNION, MINUS, UNION ALL, INTERSECT ?**  
**Ans.** MINUS – returns all distinct rows selected by the first query but not by the second.

UNION – returns all distinct rows selected by either query

UNION ALL – returns all rows selected by either query, including all duplicates.

INTERSECT – returns all distinct rows selected by both queries.

**Q#29. What is a transaction?**  
**Ans.** A transaction is a sequence of code that runs against a database. It takes database from one consistent state to another.

**Q#30. What is difference between UNIQUE and PRIMARY KEY constraints?**   
**Ans.** A table can have only one PRIMARY KEY whereas there can be any number of UNIQUE keys.

Primary key cannot contain Null values whereas Unique key can contain Null values.

**Q#31. What is a composite primary key?**  
**Ans.** Primary key created on more than one column is called composite primary key.

**Q#32. What is an Index ?**  
**Ans.** An Index is an special structure associated with a table speed up the performance of queries. Index can be created on one or more columns of a table.

**Q#33. What is the Subquery ?**   
**Ans.** A Subquery is sub set of select statements whose return values are used in filtering conditions of the main query.

**Q#34. What do you mean by query optimization?**   
**Ans.** Query optimization is a process in which database system compares different query strategies and select the query with the least cost.

**Q#35. What is Collation?**   
**Ans.** Set of rules that defines how data is stored, how case sensitivity and Kana character can be treated etc.

**Q#36. What is Referential Integrity?**  
**Ans.** Set of rules that restrict the values of one or more columns of the tables based on the values of primary key or unique key of the referenced table.

**Q#37. What is Case Function?**   
**Ans.** Case facilitates if-then-else type of logic in SQL. It evaluates a list of conditions and returns one of multiple possible result expressions.

**Q#38. Define a temp table?**  
**Ans.** A temp table is a temporary storage structure to store the data temporarily.

**Q#39. How we can avoid duplicating records in a query?**   
**Ans.** By using DISTINCT keyword duplicating records in a query can be avoided.

**Q#40. Explain the difference between Rename and Alias?**  
**Ans.** Rename is a permanent name given to a table or column whereas Alias is a temporary name given to a table or column.

**Q#41. What is a View?**  
**Ans.** A view is a virtual table which contains data from one or more tables. Views restrict data access of table by selecting only required values and make complex queries easy.

**Q#42. What are the advantages of Views?**  
**Ans.** Advantages of Views:

1. Views restrict access to the data because the view can display selective columns from the table.
2. Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing.

**Q#43. List the various privileges that a user can grant to another user?**  
**Ans.**   SELECT, CONNECT, RESOURCES.

**Q#44. What is schema?**   
**Ans.** A schema is collection of database objects of a User.

**Q#45. What is Table ?**   
**Ans.** A table is the basic unit of data storage in the database management system. Table data is stored in rows and columns.

**Q#46. Do View contain Data?**   
**Ans.** No, Views are virtual structure.

**Q#47. Can a View based on another View?**   
**Ans.** Yes, A View is based on another View.

**Q#48. What is difference between Having clause and Where clause?**   
**Ans.** Both specify a search condition but Having clause is used only with the SELECT statement and typically used with GROUP BY clause.  
If GROUP BY clause is not used then Having behaves like WHERE clause only.

**Q#49. What is difference between Local and Global temporary table?**   
**Ans.** If defined in inside a compound statement a local temporary table exists only for the duration of that statement but a global temporary table exists permanently in the db but its rows disappears when the connection is closed.

**Q#50. What is CTE?**  
**Ans.** A CTE or common table expression is an expression which contains temporary result set which is defined in a SQL statement.

### (i) Finding the nth highest salary of an employee.

Create a table named Employee\_Test and insert some test data as:-

Hide   Copy Code

CREATE TABLE Employee\_Test

(

Emp\_ID INT Identity,

Emp\_name Varchar(100),

Emp\_Sal Decimal (10,2)

)

INSERT INTO Employee\_Test VALUES ('Anees',1000);

INSERT INTO Employee\_Test VALUES ('Rick',1200);

INSERT INTO Employee\_Test VALUES ('John',1100);

INSERT INTO Employee\_Test VALUES ('Stephen',1300);

INSERT INTO Employee\_Test VALUES ('Maria',1400);

It is very easy to find the highest salary as:-

Hide   Copy Code

--Highest Salary

select max(Emp\_Sal) from Employee\_Test

Now, if you are asked to find the 3rd highest salary, then the query is as:-

Hide   Copy Code

--3rd Highest Salary

select min(Emp\_Sal) from Employee\_Test where Emp\_Sal in

(select distinct top 3 Emp\_Sal from Employee\_Test order by Emp\_Sal desc)

The result is as :- 1200   
To find the nth highest salary, replace the top 3 with top n (n being an integer 1,2,3 etc.)

Hide   Copy Code

--nth Highest Salary

select min(Emp\_Sal) from Employee\_Test where Emp\_Sal in

(select distinct top n Emp\_Sal from Employee\_Test order by Emp\_Sal desc)

### (ii) Finding TOP X records from each group

Create a table named photo\_test and insert some test data as :-

Hide   Shrink http://www.codeproject.com/images/arrow-up-16.png  Copy Code

create table photo\_test

(

pgm\_main\_Category\_id int,

pgm\_sub\_category\_id int,

file\_path varchar(MAX)

)

insert into photo\_test values

(17,15,'photo/bb1.jpg');

insert into photo\_test values(17,16,'photo/cricket1.jpg');

insert into photo\_test values(17,17,'photo/base1.jpg');

insert into photo\_test values(18,18,'photo/forest1.jpg');

insert into photo\_test values(18,19,'photo/tree1.jpg');

insert into photo\_test values(18,20,'photo/flower1.jpg');

insert into photo\_test values(19,21,'photo/laptop1.jpg');

insert into photo\_test values(19,22,'photo/camer1.jpg');

insert into photo\_test values(19,23,'photo/cybermbl1.jpg');

insert into photo\_test values

(17,24,'photo/F1.jpg');

There are three groups of pgm\_main\_category\_id each with a value of 17 (group 17 has four records),18 (group 18 has three records) and 19 (group 19 has three records).   
Now, if you want to select top 2 records from each group, the query is as follows:-

Hide   Copy Code

select pgm\_main\_category\_id,pgm\_sub\_category\_id,file\_path from

(

select pgm\_main\_category\_id,pgm\_sub\_category\_id,file\_path,

rank() over (partition by pgm\_main\_category\_id order by pgm\_sub\_category\_id asc) as rankid

from photo\_test

) photo\_test

where rankid < 3 -- replace 3 by any number 2,3 etc for top2 or top3.

order by pgm\_main\_category\_id,pgm\_sub\_category\_id

The result is as:-

Hide   Copy Code

pgm\_main\_category\_id pgm\_sub\_category\_id file\_path

17 15 photo/bb1.jpg

17 16 photo/cricket1.jpg

18 18 photo/forest1.jpg

18 19 photo/tree1.jpg

19 21 photo/laptop1.jpg

19 22 photocamer1.jpg

### (iii) Deleting duplicate rows from a table

A table with a primary key doesn’t contain duplicates. But if due to some reason, the keys have to be disabled or when importing data from other sources, duplicates come up in the table data, it is often needed to get rid of such duplicates.   
This can be achieved in tow ways :-   
(a) Using a temporary table.   
(b) Without using a temporary table.

### (a) Using a temporary or staging table

Let the table employee\_test1 contain some duplicate data like:-

Hide   Copy Code

CREATE TABLE Employee\_Test1

(

Emp\_ID INT,

Emp\_name Varchar(100),

Emp\_Sal Decimal (10,2)

)

INSERT INTO Employee\_Test1 VALUES (1,'Anees',1000);

INSERT INTO Employee\_Test1 VALUES (2,'Rick',1200);

INSERT INTO Employee\_Test1 VALUES (3,'John',1100);

INSERT INTO Employee\_Test1 VALUES (4,'Stephen',1300);

INSERT INTO Employee\_Test1 VALUES (5,'Maria',1400);

INSERT INTO Employee\_Test1 VALUES (6,'Tim',1150);

INSERT INTO Employee\_Test1 VALUES (6,'Tim',1150);

**Step 1:** Create a temporary table from the main table as:-

Hide   Copy Code

select top 0\* into employee\_test1\_temp from employee\_test1

**Step2 :** Insert the result of the GROUP BY query into the temporary table as:-

Hide   Copy Code

insert into employee\_test1\_temp

select Emp\_ID,Emp\_name,Emp\_Sal

from employee\_test1

group by Emp\_ID,Emp\_name,Emp\_Sal

**Step3:** Truncate the original table as:-

Hide   Copy Code

truncate table employee\_test1

**Step4:** Fill the original table with the rows of the temporary table as:-

Hide   Copy Code

insert into employee\_test1

select \* from employee\_test1\_temp

Now, the duplicate rows from the main table have been removed.

Hide   Copy Code

select \* from employee\_test1

gives the result as:-

Hide   Copy Code

Emp\_ID Emp\_name Emp\_Sal

1 Anees 1000

2 Rick 1200

3 John 1100

4 Stephen 1300

5 Maria 1400

6 Tim 1150

### (b) Without using a temporary table

Hide   Copy Code

;with T as

(

select \* , row\_number() over (partition by Emp\_ID order by Emp\_ID) as rank

from employee\_test1

)

delete

from T

where rank > 1

The result is as:-

Hide   Copy Code

Emp\_ID Emp\_name Emp\_Sal

1 Anees 1000

2 Rick 1200

3 John 1100

4 Stephen 1300

5 Maria 1400

6 Tim 1150

**1. What is DBMS?**

A [Database](http://career.guru99.com/category/database/) Management System (DBMS) is a program that controls creation, maintenance and use of a database. DBMS can be termed as File Manager that manages data in a database rather than saving it in file systems.

**2. What is RDBMS?**

RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.

**Example: SQL** [**Server**](http://career.guru99.com/category/server/)**.**

**3. What is SQL?**

SQL stands for Structured Query Language , and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

Standard SQL Commands are Select.

**4. What is a Database?**

Database is nothing but an organized form of data for easy access, storing, retrieval and managing of data. This is also known as structured form of data which can be accessed in many ways.

Example: School Management Database, Bank Management Database.

**5. What are tables and Fields?**

A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has specified number of column called fields but can have any number of rows which is called record.

Example:.

Table: Employee.

Field: Emp ID, Emp Name, Date of Birth.

Data: 201456, David, 11/15/1960.

**6. What is a primary key?**

A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

**7. What is a unique key?**

A Unique key constraint uniquely identified each record in the database. This provides uniqueness for the column or set of columns.

A Primary key constraint has automatic unique constraint defined on it. But not, in the case of Unique Key.

There can be many unique constraint defined per table, but only one Primary key constraint defined per table.

**8. What is a foreign key?**

A foreign key is one table which can be related to the primary key of another table. Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.

**9. What is a join?**

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

**10. What are the types of join and explain each?**

There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

**Inner join.**

Inner join return rows when there is at least one match of rows between the tables.

**Right Join.**

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table.

**Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

**Full Join.**

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

**11. What is normalization?**

Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

**12. What is Denormalization.**

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

**13. What are all the different normalizations?**

The normal forms can be divided into 5 forms, and they are explained below -.

**First Normal Form (1NF):.**

This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.

**Second Normal Form (2NF):.**

Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.

**Third Normal Form (3NF):.**

This should meet all requirements of 2NF. Removing the columns which are not dependent on primary key constraints.

**Fourth Normal Form (3NF):.**

Meeting all the requirements of third normal form and it should not have multi- valued dependencies.

**14. What is a View?**

A view is a virtual table which consists of a subset of data contained in a table. Views are not virtually present, and it takes less space to store. View can have data of one or more tables combined, and it is depending on the relationship.

**15. What is an Index?**

An index is performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.

**16. What are all the different types of indexes?**

There are three types of indexes -.

**Unique Index.**

This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.

**Clustered Index.**

This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.

**NonClustered Index.**

NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 nonclustered indexes.

**17. What is a Cursor?**

A database Cursor is a control which enables traversal over the rows or records in the table. This can be viewed as a pointer to one row in a set of rows. Cursor is very much useful for traversing such as retrieval, addition and removal of database records.

**18. What is a relationship and what are they?**

Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:.

* One to One Relationship.
* One to Many Relationship.
* Many to One Relationship.
* Self-Referencing Relationship.

**19. What is a query?**

A DB query is a code written in order to get the information back from the database. Query can be designed in such a way that it matched with our expectation of the result set. Simply, a question to the Database.

**20. What is subquery?**

A subquery is a query within another query. The outer query is called as main query, and inner query is called subquery. SubQuery is always executed first, and the result of subquery is passed on to the main query.

**21. What are the types of subquery?**

There are two types of subquery – Correlated and Non-Correlated.

A correlated subquery cannot be considered as independent query, but it can refer the column in a table listed in the FROM the list of the main query.

A Non-Correlated sub query can be considered as independent query and the output of subquery are substituted in the main query.

**22. What is a stored procedure?**

Stored Procedure is a function consists of many SQL statement to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required.

**23. What is a trigger?**

A DB trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

**24. What is the difference between DELETE and TRUNCATE commands?**

DELETE command is used to remove rows from the table, and WHERE clause can be used for conditional set of parameters. Commit and Rollback can be performed after delete statement.

TRUNCATE removes all rows from the table. Truncate operation cannot be rolled back.

**25. What are local and global variables and their differences?**

Local variables are the variables which can be used or exist inside the function. They are not known to the other functions and those variables cannot be referred or used. Variables can be created whenever that function is called.

Global variables are the variables which can be used or exist throughout the program. Same variable declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

**26. What is a constraint?**

Constraint can be used to specify the limit on the data type of table. Constraint can be specified while creating or altering the table statement. Sample of constraint are.

* NOT NULL.
* CHECK.
* DEFAULT.
* UNIQUE.
* PRIMARY KEY.
* FOREIGN KEY.

**27. What is data Integrity?**

Data Integrity defines the accuracy and consistency of data stored in a database. It can also define integrity constraints to enforce business rules on the data when it is entered into the application or database.

**28. What is Auto Increment?**

Auto increment keyword allows the user to create a unique number to be generated when a new record is inserted into the table. AUTO INCREMENT keyword can be used in Oracle and IDENTITY keyword can be used in SQL SERVER.

Mostly this keyword can be used whenever PRIMARY KEY is used.

**29. What is the difference between Cluster and Non-Cluster Index?**

Clustered index is used for easy retrieval of data from the database by altering the way that the records are stored. Database sorts out rows by the column which is set to be clustered index.

A nonclustered index does not alter the way it was stored but creates a complete separate object within the table. It point back to the original table rows after searching.

**30. What is Datawarehouse?**

Datawarehouse is a central repository of data from multiple sources of information. Those data are consolidated, transformed and made available for the mining and online processing. Warehouse data have a subset of data called Data Marts.

**31. What is Self-Join?**

Self-join is set to be query used to compare to itself. This is used to compare values in a column with other values in the same column in the same table. ALIAS ES can be used for the same table comparison.

**32. What is Cross-Join?**

Cross join defines as Cartesian product where number of rows in the first table multiplied by number of rows in the second table. If suppose, WHERE clause is used in cross join then the query will work like an INNER JOIN.

**33. What is user defined functions?**

User defined functions are the functions written to use that logic whenever required. It is not necessary to write the same logic several times. Instead, function can be called or executed whenever needed.

**34. What are all types of user defined functions?**

Three types of user defined functions are.

* Scalar Functions.
* Inline Table valued functions.
* Multi statement valued functions.

Scalar returns unit, variant defined the return clause. Other two types return table as a return.

**35. What is collation?**

Collation is defined as set of rules that determine how character data can be sorted and compared. This can be used to compare A and, other language characters and also depends on the width of the characters.

ASCII value can be used to compare these character data.

**36. What are all different types of collation sensitivity?**

Following are different types of collation sensitivity -.

* Case Sensitivity – A and a and B and b.
* Accent Sensitivity.
* Kana Sensitivity – Japanese Kana characters.
* Width Sensitivity – Single byte character and double byte character.

**37. Advantages and Disadvantages of Stored Procedure?**

Stored procedure can be used as a modular [programming](http://career.guru99.com/category/programming-2/) – means create once, store and call for several times whenever required. This supports faster execution instead of executing multiple queries. This reduces network traffic and provides better security to the data.

Disadvantage is that it can be executed only in the Database and utilizes more memory in the database server.

**38. What is Online Transaction Processing (OLTP)?**

Online Transaction Processing or OLTP manages transaction based applications which can be used for data entry and easy retrieval processing of data. This processing makes like easier on simplicity and efficiency. It is faster, more accurate results and expenses with respect to OTLP.

Example – Bank Transactions on a daily basis.

**39. What is CLAUSE?**

SQL clause is defined to limit the result set by providing condition to the query. This usually filters some rows from the whole set of records.

Example – Query that has WHERE condition

Query that has HAVING condition.

**40. What is recursive stored procedure?**

A stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps programmers to use the same set of code any number of times.

**41. What is Union, minus and Interact commands?**

UNION operator is used to combine the results of two tables, and it eliminates duplicate rows from the tables.

MINUS operator is used to return rows from the first query but not from the second query. Matching records of first and second query and other rows from the first query will be displayed as a result set.

INTERSECT operator is used to return rows returned by both the queries.

**42. What is an ALIAS command?**

ALIAS name can be given to a table or column. This alias name can be referred in WHERE clause to identify the table or column.

Example-.

Select st.StudentID, Ex.Result from student st, Exam as Ex where st.studentID = Ex. StudentID

|  |  |
| --- | --- |
| 1 | Select st.StudentID, Ex.Result from student st, Exam as Ex where st.studentID = Ex. StudentID |

Here, st refers to alias name for student table and Ex refers to alias name for exam table.

**43. What is the difference between TRUNCATE and DROP statements?**

TRUNCATE removes all the rows from the table, and it cannot be rolled back. DROP command removes a table from the database and operation cannot be rolled back.

**44. What are aggregate and scalar functions?**

Aggregate functions are used to evaluate mathematical calculation and return single values. This can be calculated from the columns in a table. Scalar functions return a single value based on the input value.

Example -.

Aggregate – max(), count – Calculated with respect to numeric.

Scalar – UCASE(), NOW() – Calculated with respect to strings.

**45. How can you create an empty table from an existing table?**

Example will be -.

Select \* into studentcopy from student where 1=2

|  |  |
| --- | --- |
| 1 | Select \* into studentcopy from student where 1=2 |

Here, we are copying student table to another table with the same structure with no rows copied.

**46. How to fetch common records from two tables?**

Common records result set can be achieved by -.

Select studentID from student. <strong>INTERSECT </strong> Select StudentID from Exam

|  |  |
| --- | --- |
| 1 | Select studentID from student. <strong>INTERSECT </strong> Select StudentID from Exam |

**47. How to fetch alternate records from a table?**

Records can be fetched for both Odd and Even row numbers -.

To display even numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0

|  |  |
| --- | --- |
| 1 | Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0 |

To display odd numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1

|  |  |
| --- | --- |
| 1 | Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1 |

from (Select rowno, studentId from student) where mod(rowno,2)=1.[/sql]

**48. How to select unique records from a table?**

Select unique records from a table by using DISTINCT keyword.

Select DISTINCT StudentID, StudentName from Student.

|  |  |
| --- | --- |
| 1 | Select DISTINCT StudentID, StudentName from Student. |

**49. What is the command used to fetch first 5 characters of the string?**

There are many ways to fetch first 5 characters of the string -.

Select SUBSTRING(StudentName,1,5) as studentname from student

|  |  |
| --- | --- |
| 1 | Select SUBSTRING(StudentName,1,5) as studentname from student |

Select RIGHT(Studentname,5) as studentname from student

|  |  |
| --- | --- |
| 1 | Select RIGHT(Studentname,5) as studentname from student |

**50. Which operator is used in query for pattern matching?**

LIKE operator is used for pattern matching, and it can be used as -.

1. % – Matches zero or more characters.
2. \_(Underscore) – Matching exactly one character.

**Example -.**

Select \* from Student where studentname like ‘a%’

|  |  |
| --- | --- |
| 1 | Select \* from Student where studentname like ‘a%’ |

Select \* from Student where studentname like ‘ami\_’

|  |  |
| --- | --- |
| 1 | Select \* from Student where studentname like ‘ami\_’ |

# [Complex Queries in SQL ( Oracle )](http://www.bullraider.com/database/sql-tutorial/7-complex-queries-in-sql)

[ddThis Social Bookmark Button](http://www.addthis.com/bookmark.php)

These questions are the most frequently asked in interviews.

1. **To fetch ALTERNATE records from a table. (EVEN NUMBERED)**select \* from emp where rowid in (select decode(mod(rownum,2),0,rowid, null) from emp);
2. **To select ALTERNATE records from a table. (ODD NUMBERED)**select \* from emp where rowid in (select decode(mod(rownum,2),0,null ,rowid) from emp);
3. **Find the 3rd MAX salary in the emp table.**select distinct sal from emp e1 where 3 = (select count(distinct sal) from emp e2 where e1.sal <= e2.sal);
4. **Find the 3rd MIN salary in the emp table.**select distinct sal from emp e1 where 3 = (select count(distinct sal) from emp e2where e1.sal >= e2.sal);
5. **Select FIRST n records from a table.**select \* from emp where rownum <= &n;
6. **Select LAST n records from a table**select \* from emp minus select \* from emp where rownum <= (select count(\*) - &n from emp);
7. **List dept no., Dept name for all the departments in which there are no employees in the department.**select \* from dept where deptno not in (select deptno from emp);    
   alternate solution:  select \* from dept a where not exists (select \* from emp b where a.deptno = b.deptno);  
   altertnate solution:  select empno,ename,b.deptno,dname from emp a, dept b where a.deptno(+) = b.deptno and empno is null;
8. **How to get 3 Max salaries ?**select distinct sal from emp a where 3 >= (select count(distinct sal) from emp b where a.sal <= b.sal) order by a.sal desc;
9. **How to get 3 Min salaries ?**select distinct sal from emp a  where 3 >= (select count(distinct sal) from emp b  where a.sal >= b.sal);
10. **How to get nth max salaries ?**  
    select distinct hiredate from emp a where &n =  (select count(distinct sal) from emp b where a.sal >= b.sal);
11. **Select DISTINCT RECORDS from emp table.**select \* from emp a where  rowid = (select max(rowid) from emp b where  a.empno=b.empno);
12. **How to delete duplicate rows in a table?**delete from emp a where rowid != (select max(rowid) from emp b where  a.empno=b.empno);
13. **Count of number of employees in  department  wise.**select count(EMPNO), b.deptno, dname from emp a, dept b  where a.deptno(+)=b.deptno  group by b.deptno,dname;
14. **Suppose there is annual salary information provided by emp table. How to fetch monthly salary of each and every employee?**

select ename,sal/12 as monthlysal from emp;

1. **Select all record from emp table where deptno =10 or 40.**

select \* from emp where deptno=30 or deptno=10;

1. **Select all record from emp table where deptno=30 and sal>1500.**

select \* from emp where deptno=30 and sal>1500;

1. **Select  all record  from emp where job not in SALESMAN  or CLERK.**

select \* from emp where job not in ('SALESMAN','CLERK');

1. **Select all record from emp where ename in 'BLAKE','SCOTT','KING'and'FORD'.**

select \* from emp where ename in('JONES','BLAKE','SCOTT','KING','FORD');

1. **Select all records where ename starts with ‘S’ and its lenth is 6 char.**

select \* from emp where ename like'S\_\_\_\_';

1. **Select all records where ename may be any no of  character but it should end with ‘R’.**

select \* from emp where ename like'%R';

1. **Count  MGR and their salary in emp table.**

select count(MGR),count(sal) from emp;

1. **In emp table add comm+sal as total sal  .**

select ename,(sal+nvl(comm,0)) as totalsal from emp;

1. **Select  any salary <3000 from emp table.**

select \* from emp  where sal> any(select sal from emp where sal<3000);

1. **Select  all salary <3000 from emp table.**

select \* from emp  where sal> all(select sal from emp where sal<3000);

1. **Select all the employee  group by deptno and sal in descending order.**

select ename,deptno,sal from emp order by deptno,sal desc;

1. **How can I create an empty table emp1 with same structure as emp?**

Create table emp1 as select \* from emp where 1=2;

1. **How to retrive record where sal between 1000 to 2000?**  
   Select \* from emp where sal>=1000 And  sal<2000
2. **Select all records where dept no of both emp and dept table matches.**  
   select \* from emp where exists(select \* from dept where emp.deptno=dept.deptno)
3. **If there are two tables emp1 and emp2, and both have common record. How can I fetch all the recods but common records only once?**  
   (Select \* from emp) Union (Select \* from emp1)
4. **How to fetch only common records from two tables emp and emp1?**  
   (Select \* from emp) Intersect (Select \* from emp1)
5. **How can I retrive all records of emp1 those should not present in emp2?**  
   (Select \* from emp) Minus (Select \* from emp1)
6. **Count the totalsa  deptno wise where more than 2 employees exist.**  
   SELECT  deptno, sum(sal) As totalsal  
   FROM emp  
   GROUP BY deptno  
   HAVING COUNT(empno) > 2