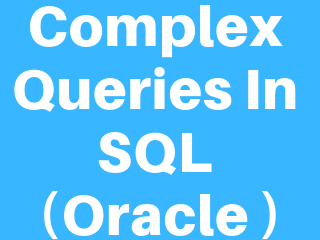
**[Complex Queries in SQL ( Oracle )](https://www.pavantestingtools.com/2012/03/complex-queries-in-sql-oracle.html)**

[](https://2.bp.blogspot.com/-Xh9tESBHFBs/XFVHoDl8V8I/AAAAAAAAP6I/FoAK38yVk7QZDvuw-jcZI9H4oLW_1IX0QCLcBGAs/s1600/Programs%2Bfor%2BSelenium%25286%2529.png)

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;  
  
**15.Select all record from emp table where deptno =10 or 40.**  
  
select \* from emp where deptno=30 or deptno=10;  
  
**16.Select all record from emp table where deptno=30 and sal>1500.**  
  
select \* from emp where deptno=30 and sal>1500;  
  
**17.Select all record from emp where job not in SALESMAN or CLERK.**  
  
select \* from emp where job not in ('SALESMAN','CLERK');  
  
**18.Select all record from emp where ename in 'BLAKE','SCOTT','KING'and'FORD'.**  
  
select \* from emp where ename in('JONES','BLAKE','SCOTT','KING','FORD');  
  
**19.Select all records where ename starts with ‘S’ and its lenth is 6 char.**  
  
select \* from emp where ename like'S\_\_\_\_';  
  
**20.Select all records where ename may be any no of character but it should end with ‘R’.**  
  
select \* from emp where ename like'%R';  
  
**21.Count MGR and their salary in emp table.**  
  
select count(MGR),count(sal) from emp;  
  
**22.In emp table add comm+sal as total sal .**  
select ename,(sal+nvl(comm,0)) as totalsal from emp;  
  
**23.Select any salary <3000 from emp table.**  
  
select \* from emp where sal> any(select sal from emp where sal<3000);  
  
**24.Select all salary <3000 from emp table.**  
  
select \* from emp where sal> all(select sal from emp where sal<3000);  
  
**25.Select all the employee group by deptno and sal in descending order.**  
  
select ename,deptno,sal from emp order by deptno,sal desc;  
  
**26.How can I create an empty table emp1 with same structure as emp?**  
  
Create table emp1 as select \* from emp where 1=2;  
  
**27.How to retrive record where sal between 1000 to 2000?**  
  
Select \* from emp where sal>=1000 And sal<2000  
  
**28.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)  
  
**29.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)  
  
**30.How to fetch only common records from two tables emp and emp1?**  
  
(Select \* from emp) Intersect (Select \* from emp1)  
**31. How can I retrive all records of emp1 those should not present in emp2?**  
  
(Select \* from emp) Minus (Select \* from emp1)  
  
**32.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

A **subquery**is a type of SQL query, where a query is embedded within another query. Sub-queries are very powerful. To help you understand a subquery consider the following SELECT statement to retrieve the details of employees who belong to department 30.

SELECT \* FROM EMPLOYEES

WHERE DEPARTMENT\_ID=30;

In the above query, the department ID value has been provided, and is used on the right hand side of the WHERE condition. However, such constant values might not also be provided or known. For example, consider the query re-phrased as - retrieve the details of employees who belong to the same department as 'Alexander Khoo'. Here the department number has not been provided. Instead the name of an employee is given. Using this name, you would need to first find out - to which department does Alexander Khoo belong. Let say this is some value 'X'. You would have to proceed further to find out all the other employees who belong to the department X.

If you notice this is a 2-step process involving:

1) Which department does Alexander Khoo belong to.

2) Who are the others who belong to the department number returned by the first step.

Subquery Syntax:

SELECT select\_list

FROM table\_name

WHERE column\_name operator (SELECT select\_list

FROM table\_name

…)

In the syntax, observe a second SELECT statement written in the WHERE clause, on the right hand side of the WHERE condition. This SELECT statement is enclosed in parantheses. This subquery is called the inner query and is executed once to return a value that is used by the main (outer) query. Subqueries can be different in different places in a SELECT statement, such as the WHERE clause, HAVING clause, FROM clause, SELECT column list etc.

The query to retrieve the details of employees who belong to the same department as Alexander Khoo is :

SELECT \*

FROM EMPLOYEES

WHERE DEPARTMENT\_ID = (SELECT DEPARTMENT\_ID

FROM EMPLOYEES

WHERE FIRST\_NAME='Alexander' AND LAST\_NAME='Khoo')

**Some guidelines related to subqueries are:**

• Enclose subqueries in parentheses.

• Place subqueries on the right side of the comparison condition.

• Use single-row operators with single-row subqueries.

• Use multiple-row operators with multiple-row subqueries.

**A single-row** subquery is one where the subquery returns only one value. In such a subquery you must use a single-row operator such as:

|  |  |
| --- | --- |
| Operator | Description |
| = | Equal To |
| <> | Not Equal To |
| > | Greater Than |
| >= | Greater Than Equal To |
| < | Less Than |
| <= | Less Than Equal To |

The single-row operators are used to write single-row subqueries. The table below demonstrates the use of the single-row operators in writing single-row subqueries.

|  |  |  |
| --- | --- | --- |
| Operator | Query | Example |
| = | Retreive the details of employees who get the same salary as the employee whose ID is 101. | SELECT \* FROM EMPLOYEES  WHERE SALARY=(SELECT SALARY FROM EMPLOYEES  WHERE EMPLOYEE\_ID=101); |
| <> | Retreive the details of departments that are not located in the same location ID as department 10. | SELECT \*  FROM DEPARTMENTS  WHERE LOCATION\_ID <>(SELECT           LOCATION\_ID         FROM DEPARTMENTS         WHERE DEPARTMENT\_ID=10); |
| > | Retrieve the details of employees whose salary is greater than the minimum salary. | SELECT \*  FROM EMPLOYEES  WHERE SALARY > (SELECT          MIN(SALARY)         FROM EMPLOYEES); |
| >= | Retrieve the details of employees who were hired on or after the same date that employee 201 was hired. | SELECT \* FROM EMPLOYEES  WHERE HIRE\_DATE >=(SELECT         HIRE\_DATE         FROM EMPLOYEES        WHERE EMPLOYEE\_ID=201); |
| < | Retrieve the details of employees whose salary is less than the maximum salary of employees in department 20. | SELECT \* FROM EMPLOYEES  WHERE SALARY < (SELECT      MAX(SALARY)      FROM EMPLOYEES      WHERE DEPARTMENT\_ID=20); |
| <= | Retrieve the details of employees who were hired on or before the same date that employee 201 was hired. | SELECT \* FROM EMPLOYEES  WHERE HIRE\_DATE <=(SELECT        HIRE\_DATE        FROM EMPLOYEES        WHERE EMPLOYEE\_ID=201); |

**A multiple row** subquery is one where the subquery may return more than one value. In such type of subquery, it is necessary to use a multiple-row operator. If not you might get the ORA-01427 error: single-row subquery returns more than requested number of rows.

The table below describes the multiple-row operators that can be used when writing multiple-row subqueries:

|  |  |
| --- | --- |
| Operator | Meaning |
| IN | Equal to any value returned by the subquery |
| ANY | Compare value to each value returned by the subquery |
| ALL | Compare value to every value returned by the subquery |

The multiple-row operators are used to write multiple-row subqueries. The table below demonstrates the use of the multiple-row operators in writing multiple-row subqueries.

|  |  |  |
| --- | --- | --- |
| Operator | Query | Example |
| IN | Retreive the department ID, department name and location ID of departments that are located in the same location ID as a location in the UK. | ﻿﻿SELECT DEPARTMENT\_ID, DEPARTMENT\_NAME, LOCATION\_ID  FROM DEPARTMENTS  WHERE LOCATION\_ID IN (SELECT LOCATION\_ID FROM LOCATIONS WHERE COUNTRY\_ID='UK') |
| >ALL  (Greater than the maximum returned by the subquery) | Retrieve the first name of employees whose salary is greater than the all the salaries of employees belonging to department 20. | SELECT FIRST\_NAME  FROM EMPLOYEES  WHERE SALARY > ALL  (SELECT SALARY  FROM EMPLOYEES  WHERE DEPARTMENT\_ID=20) |
| <ALL  (Less than the least value returned by the subquery) | Retrieve the first name of employees whose salary is less than all the salaries of employees belonging to department 20. | SELECT FIRST\_NAME  FROM EMPLOYEES  WHERE SALARY < ALL  (SELECT SALARY  FROM EMPLOYEES  WHERE DEPARTMENT\_ID=20) |
| >ANY  (Greater than the minimum value returned by the subquery) | Retrieve the first name of employees whose salary is greater than the minimum salary of employees in department 60. | SELECT FIRST\_NAME  FROM EMPLOYEES  WHERE SALARY > ANY (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=60) |
| <ANY  (Less than the maximum value returned by the subquery) | Retrieve the first name of employees whose salary is less than the maximum salary of employees in department 60. | SELECT FIRST\_NAME  FROM EMPLOYEES  WHERE SALARY < ANY  (SELECT SALARY  FROM EMPLOYEES WHERE DEPARTMENT\_ID=10) |

A sub query is a type of SQL query, where a query is embedded within another query. Sub-queries are very powerful. To help you understand subqueries consider the following SELECT statement to retrieve the details of employees who belong to department 30.  
  
  
**SELECT \* FROM EMPLOYEES**  
  
  
**WHERE DEPARTMENT\_ID=30;**  
  
  
In the above query, the department ID value has been provided, and is used on the right hand side of the WHERE condition. However, such constant values might not also be provided or known. For example, consider the query re-phrased as - retrieve the details of employees who belong to the same department as 'Alexander Khoo'. Here the department number has not been provided. Instead the name of an employee is given. Using this name, you would need to first find out - to which department does Alexander Khoo belong. Let say this is some value 'X'. You would have to proceed further to find out all the other employees who belong to the department X.  
  
  
If you notice this is a 2-step process involving:  
  
  
1) Which department does Alexander Khoo belong to.  
  
  
2) Who are the others who belong to the department number returned by the first step.  
  
  
Subquery Syntax:  
  
  
**SELECT select\_list**  
  
  
**FROM table\_name**  
  
  
**WHERE column\_name operator (SELECT select\_list**  
  
  
**FROM table\_name**  
  
  
**…)**  
  
  
  
In the syntax, observe a second SELECT statement written in the WHERE clause, on the right hand side of the WHERE condition. This SELECT statement is enclosed in parantheses. This subquery is called the inner query and is executed once to return a value that is used by the main (outer) query. Subqueries can be different in different places in a SELECT statement, such as the WHERE clause, HAVING clause, FROM clause, SELECT column list etc.  
  
  
The query to retrieve the details of employees who belong to the same department as Alexander Khoo is :  
  
  
**SELECT \***  
  
  
**FROM EMPLOYEES**  
  
  
**WHERE DEPARTMENT\_ID = (SELECT DEPARTMENT\_ID**  
  
  
**FROM EMPLOYEES**  
  
  
**WHERE FIRST\_NAME='Alexander' AND LAST\_NAME='Khoo')**  
  
  
Some guidelines related to subqueries are:  
  
  
• Enclose subqueries in parentheses.  
  
  
• Place subqueries on the right side of the comparison condition.  
  
  
• Use single-row operators with single-row subqueries.  
  
  
• Use multiple-row operators with multiple-row subqueries.  
  
  
  
  
**A single-row subquery** is one where the subquery returns only one value. In such a subquery you must use a single-row operator such as:  
  
  
The single-row operators are used to write single-row subqueries. The table below demonstrates the use of the single-row operators in writing single-row subqueries.  
  
  
  
  
=    **Retreive the details of employees who get the same salary as the employee whose ID is 101.**  
  
SELECT \* FROM EMPLOYEES  
  
  
WHERE SALARY=(SELECT SALARY FROM EMPLOYEES  
  
  
WHERE EMPLOYEE\_ID=101);  
  
  
<> **Retreive the details of departments that are not located in the same location ID as department 10.**  
  
  
SELECT \*  
  
  
FROM DEPARTMENTS  
  
  
WHERE LOCATION\_ID <>(SELECT  
  
  
LOCATION\_ID  
  
  
FROM DEPARTMENTS  
  
  
WHERE DEPARTMENT\_ID=10);  
  
  
>   **Retrieve the details of employees whose salary is greater than the minimum salary.**  
  
  
SELECT \*  
  
  
FROM EMPLOYEES  
  
  
WHERE SALARY > (SELECT  
  
  
MIN(SALARY)  
  
  
FROM EMPLOYEES);  
  
  
>=    **Retrieve the details of employees who were hired on or after the same date that employee 201 was hired.**  
  
SELECT \* FROM EMPLOYEES  
  
  
WHERE HIRE\_DATE >=(SELECT  
  
  
HIRE\_DATE  
  
  
FROM EMPLOYEES WHERE EMPLOYEE\_ID=201);  
  
  
<   **Retrieve the details of employees whose salary is less than the maximum salary of employees in department 20.**  
  
  
SELECT \* FROM EMPLOYEES  
  
  
WHERE SALARY < (SELECT  
  
  
MAX(SALARY)  
  
  
FROM EMPLOYEES  
  
  
WHERE DEPARTMENT\_ID=20);  
  
  
<=    **Retrieve the details of employees who were hired on or before the same date that employee 201 was hired.**  
  
  
SELECT \* FROM EMPLOYEES  
  
  
WHERE HIRE\_DATE <=(SELECT  
  
  
HIRE\_DATE  
  
  
FROM EMPLOYEES  
  
  
WHERE EMPLOYEE\_ID=201);  
  
  
**A multiple row subquery** is one where the subquery may return more than one value. In such type of subquery, it is necessary to use a multiple-row operator. If not you might get the ORA-01427 error: single-row subquery returns more than requested number of rows.  
  
  
The table below describes the multiple-row operators that can be used when writing multiple-row subqueries:  
  
  
  
  
**IN** Equal to any value returned by the subquery  
  
  
**ANY**  Compare value to each value returned by the subquery  
  
  
**ALL** Compare value to every value returned by the subquery  
  
  
The multiple-row operators are used to write multiple-row subqueries. The table below demonstrates the use of the multiple-row operators in writing multiple-row subqueries.  
  
  
  
  
IN  -    **Retreive the department ID, department name and location ID of departments that are located in the same location ID as a location in the UK.**  
  
  
SELECT DEPARTMENT\_ID, DEPARTMENT\_NAME, LOCATION\_ID  
  
  
FROM DEPARTMENTS  
  
  
WHERE LOCATION\_ID IN (SELECT LOCATION\_ID FROM LOCATIONS WHERE COUNTRY\_ID='UK')  
  
  
>  **ALL** (Greater than the maximum returned by the subquery)  
  
**Retrieve the first name of employees whose salary is greater than the all the salaries of employees belonging to department 20.**  
  
  
SELECT FIRST\_NAME  
  
  
FROM EMPLOYEES  
  
  
WHERE SALARY > ALL  
  
  
(SELECT SALARY  
  
  
FROM EMPLOYEES  
  
  
WHERE DEPARTMENT\_ID=20)  
  
  
  
  
  
(Less than the least value returned by the subquery)  
  
**Retrieve the first name of employees whose salary is less than all the salaries of employees belonging to department 20.**  
  
  
SELECT FIRST\_NAME  
  
  
FROM EMPLOYEES  
  
  
WHERE SALARY < ALL  
  
  
(SELECT SALARY  
  
  
FROM EMPLOYEES  
  
  
WHERE DEPARTMENT\_ID=20)  
  
  
**>ANY -**(Greater than the minimum value returned by the subquery)  
  
**Retrieve the first name of employees whose salary is greater than the minimum salary of employees in department 60.**  
  
  
SELECT FIRST\_NAME  
  
  
FROM EMPLOYEES  
  
  
WHERE SALARY > ANY (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=60)  
  
  
  
**(Less than the maximum value returned by the subquery)**  
  
Retrieve the first name of employees whose salary is less than the maximum salary of employees in department 60.  
  
  
SELECT FIRST\_NAME  
  
  
FROM EMPLOYEES  
  
  
WHERE SALARY < ANY  
  
  
(SELECT SALARY  
  
  
FROM EMPLOYEES WHERE DEPARTMENT\_ID=10)