SELECT

Querying Database Tables

The SELECT statement instructs the database to retrieve information from the tables.

Syntax:

```
SELECT  * |[DISTINCT <Column_list>]
FROM <table_name>
[WHERE <condition>]
[GROUP BY <column_name(s)>]
[HAVING <condition>
[ORDER BY <expression>]
```

```
set PAGESIZE 60;
set LINESIZE 132;
```

Ex:

To select all the columns and all the rows from a table

```
> SELECT * FROM EMP;
```

Ex:

To select few columns and all rows of a table

```
> SELECT empno, ename, job, sal FROM emp;

> SELECT deptno, loc FROM dept;
```

Conditional Retrieval of Rows:

The WHERE clause is used along with the SELECT statement to specify the condition, based on the condition, rows will be extracted from a table with SELECT.

Operators used to specify the conditions:

Relational Operators	Logical Operators
= Equal to	AND Logical AND
> Greater than	OR Logical OR
< Less then	NOT Logical NOT
>= Greater then or Equal to	
<= Less then or Equal to	
<> or != Not equal to	

Ex. List the employees belonging to the department 20

```
SELECT  * FROM EMP WHERE DEPTNO = 20;
```

Ex. List details of all the managers

```
SELECT * FROM EMP
WHERE UPPER(JOB) = 'MANAGER'
```

Ex. List the names of the clerks working in the department 20

```
SELECT ENAME FROM EMP
WHERE UPPER(JOB) = 'CLERK' AND DEPTNO = 20;
```

Ex. List the names of analysts and salesmen

```
SELECT ENAME FROM EMP
WHERE UPPER(JOB) = 'ANALYST' OR  UPPER(JOB)  = ' SALESMEN' ;
```

Ex. List details of employees who are not managers

```
SELECT * FROM EMP
WHERE UPPER(JOB) <> 'MANAGER' ;
```

Ex. List the details of employees who have joined before the end of September 81

```
SELECT * FROM EMP
WHERE  HIREDATE <= '30-SEP-81';
```

IN	**Checking a value in a set**
BETWEEN	**Checking a value with in a range**
LIKE	**Matching a pattern from a column**

Ex. List the details of the employees who are working in the departments 10 , 20 or 30

```
SELECT * FROM EMP
WHERE DEPTNO = 10 OR DEPTNO = 20 OR DEPTNO = 30;
```

or

```
SELECT * FROM EMP
WHERE DEPTNO IN (10,20,30);
```

Ex. List the employee details not belonging to the department 10,30 and 40

```
SELECT * FROM EMP
WHERE DEPTNO NOT IN (10,30,40);
```

Ex. List the employee names and salary whose salary is between 10000 and 25000

```
SELECT ENAME , SAL FROM EMP
WHERE  SAL BETWEEN 10000 AND 25000;
```

NOTE: In the BETWEEN operator, both lower limit and upper limit are inclusive.

Ex. List the employees who have joined before 30th June '81 and after December '81

```
SELECT ENAME FROM EMP
WHERE HIREDATE
NOT BETWEEN  '30-JUN-81' AND '31-DEC-81';
```

DISTINCT Clause:

The DISTINCT clause is used with select to suppress duplicate values if any in a column.

Ex. List the different jobs(designations) available in the emp table.

```
SELECT DISTINCT JOB FROM EMP;
```

NULL Values:

- NULL value is not 0 or a Blank.
- It represents an unknown or inapplicable value.
- It cannot be compared using the relational and/or logical operators.
- The special operator ‘IS’ is used with keyword ‘NULL’ to locate NULL values.

Ex. List the employee names, salary who are not eligible for commission

```
SELECT ENAME, SAL FROM EMP
WHERE COMM IS NULL;
```

Ex. List the name of the employee and designation (job) of the employee, who does not report to anyone.

```
SELECT ENAME, JOB FROM EMP
WHERE MGR IS NULL;
```

Ex. List the employees who are eligible for commission.

```
SELECT * FROM EMP
WHERE COMM IS NOT NULL;
```

Ex. List the employees whose salary is greater than 20000 and commission is NULL.

```
SELECT * FROM EMP
WHERE SAL > 20000 AND COMM IS NULL;
```

Matching a pattern with a column from table

- The LIKE operator is used to match pattern.
- '%' represents a sequence of zero or more characters.
- '_' (Underscore) represents for any single character.
- Both '%' and '_' are wild card characters and are used along with like operator to specify a pattern

Ex:

List the employees whose names start with an 'S'

```
SELECT EMPNO,ENAME,JOB FROM EMP
WHERE ENAME LIKE 'S%';
```

Ex. List the employee names ending with an 'S'

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '%S';
```

Ex. List the names of employees whose names have exactly 5 characters.

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '_____';
```

Ex. List the employee names having 'm' as the second character.

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '_m%';
```

Ex. List the employees who joined in the year '81'

```
SELECT * FROM EMP
WHERE HIREDATE LIKE '%81';
```

To search the pattern containing the wild card characters ('%','_') use the ESCAPE clause with the LIKE operator

Ex. List the employee names contains underscore ('_')

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '%\_%' ESCAPE '\';
```

Using expressions with columns

You can use arithmetic expressions on the number columns

EX: List the employee name, job, salary and annual pay of the all the employees

```
SELECT ENAME,JOB,SAL ,12 * (SAL + NVL(COMM,0)) AS "ANNUAL PAY "
FROM EMP;
```

Ex. List the employee names and experience of all the employees

```
SELECT ENAME, ROUND ((SYSDATE-HIREDATE)/365) AS "EXP"
FROM EMP;
```

Concatenation Operator

```
>     select ename ||  '  earns '  ||     sal   from emp ;

> select ' Empno : ' || empno || ' is ' || job || ' and earns  Rs. ' ||  sal
  from emp ;
```

PSEUDOCOLUMNS

ROWNUM , NEXTVAL , CURRVAL, ROWID & LEVEL

```
Select   rownum , ename from emp where deptno = 10;

Select  * from emp where rownum <=10;
```

Ordering the Results of Query

- **SQL uses the ORDER BY clause to impose an order on the result of a query**
- **ORDER BY clause is used with select statement**
- **One or more columns and/or expressions can be specified in ORDER BY clause.**

Ex: ***List the empno, ename , sal in ascending order of sal***

```
SELECT EMPNO,ENAME, SAL FROM EMP
ORDER BY SAL;
```

Ex. List employee names and hiredates in descending order of hiredate

```
SELECT ENAME,HIREDATE AS DATE_OF_JOINING
FROM EMP
ORDER BY DATE_OF_JOINING;
```

Ex. List the employee name, salary, job and department number in ascending order of department number and then on descending order of salary.

```
SELECT DEPTNO, JOB, ENAME, SAL FROM EMP
ORDER BY DEPTNO, SAL DESC
```

Note: In place of column names we can use numbers to indicate the fields being used to order the output.
These numbers will refer not to the order of the columns in the table, but to their orders in the output.

Ex. List the employee details in ascending order of salary

```
SELECT EMPNO, ENAME, SAL FROM EMP
ORDER BY 3;
```

Aggregate (Group) Functions:

Aggregate functions produce a single value for a group of table data.

COUNT : returns number of rows of non NULL column values.
SUM : returns sum of all selected column values.
MAX : returns maximum value of the selected column.
MIN : returns minimum value of the selected column.
AVG : returns average value of the selected column.

NOTE:

All the above functions ignore NULL values.
MIN & MAX functions can be applied on character & date type values also.

Ex. List no. of SALESMAN who doesn't get commission

```
SELECT COUNT(COMM) FROM EMP
WHERE UPPER(JOB) = 'SALESMAN';
```

Ex. List total no. of employees.

```
SELECT COUNT(*) FROM EMP ;
```

Ex. List minimum and maximum salary of the employees.

```
SELECT MIN (SAL) , MAX(SAL)
FROM EMP;
```

List the minimum & maximum salary of a salesman.

```
SELECT MIN (SAL) , MAX(SAL)
FROM EMP WHERE UPPER(JOB) = 'SALESMAN' ;
```

Ex. List the average salary and number of employees working in department 20

```
SELECT AVG (SAL), COUNT (*) FROM EMP
WHERE DEPTNO = 20;
```

Ex. List total pay of all the employees working in department 20

```
SELECT sum(sal) + nvl(sum(comm),0) as payroll
FROM EMP
WHERE DEPTNO = 20;
```

Grouping the result of a Query

- The GROUP BY clause is used with SELECT to combine a group of rows based on the values of a particular column.
- Aggregate functions are used to return summary information for each group.
- The aggregate functions are applied to the individual group.
- The GROUP BY clause is used to divide the rows in a table in to smaller groups.
- The GROUP BY clause is used with SELECT statement.
- SQL groups the results after it retrieves the rows from table.
- Conditional retrieval of rows from a grouped result is possible with the HAVING clause.
- ORDER BY clause can be used to order the final result.

Ex. List the department numbers and number of employees in each department.

```
SELECT DEPTNO, COUNT (*)
FROM EMP
GROUP BY DEPTNO;
```

Ex. List the department number and payroll of each department.

```
SELECT DEPTNO, SUM(SAL) + nvl(sum(comm),0) as payroll
FROM EMP
GROUP BY DEPTNO;
```

Ex. *List the average salary from each job excluding managers.*

```
SELECT JOB, AVG(SAL) FROM EMP
WHERE UPPER(JOB) != 'MANAGER'
GROUP BY JOB;
```

Ex. List the jobs and the number of employees in each job. The result should be in descending order of the number of employees.

```
SELECT JOB, COUNT (*) as "No. of employees"
FROM EMP
GROUP BY JOB  ORDER BY 2;
```

HAVING Clause

Having clause is used to specify which groups are to be displayed, that is restrict the groups that you return on the basis of aggregate functions.

Ex. List the average salary for all departments employing more then five people.

```
SELECT DEPTNO, AVG (SAL) FROM EMP
GROUP DEPTNO
HAVING COUNT(*) > 5;
```

Ex. List total , maximum , minimum and average salary of all the employees job wise for department 20 and display only those rows having average salary greater then 10000.

```
SELECT JOB, SUM(SAL), MAX(SAL), MIN(SAL), AVG(SAL)
FROM EMP
WHERE DEPTNO = 20
GROUP BY JOB
HAVING AVG(SAL) > 10000
ORDER BY JOB
```

Note: Columns specified in the select list should also appear in the group by clause.

Querying the data from multiple tables

JOINS:

There are four types of joins

1. **Cartesian Join**
2. **Equi-Join (Natural Join)**
3. **Outer Join**
4. **Self Join**

DEPT Table

Deptno	Dname	Loc
10	EDP	Hyderabad
20	STORES	Mumbai
30	ACCOUNTS	Bangalore

EMP Table

Empno	Ename	Deptno
1001	Ravi	10
1002	Laxmi	20
1003	Venu	10
1004	Madhu	40

Cartesian Join

When no join condition clause is specified in where clause than each row of one table matches every row of the other table.
This results in a Cartesian product.

Ex.

```
Select   empno, ename, e.deptno, d.deptno,dname,loc
 from emp e , dept d ;
```

EMPNO	ENAME	E.DEPTNO	D.DEPTNO	DNAME	LOC
1001	Ravi	10	10	EDP	Hyderabad
1001	Ravi	10	20	STORES	Mumbai
1001	Ravi	10	30	ACCOUNTS	Bangalore
1002	Laxmi	20	10	EDP	Hyderabad
1002	Laxmi	20	20	STORES	Mumbai
1002	Laxmi	20	30	ACCOUNTS	Bangalore
1003	Venu	10	10	EDP	Hyderabad
1003	Venu	10	20	STORES	Mumbai
1003	Venu	10	30	ACCOUNTS	Bangalore
1004	Madhu	40	10	EDP	Hyderabad
1004	Madhu	40	20	STORES	Mumbai
1004	Madhu	40	30	ACCOUNTS	Bangalore

EQUI-JOIN OR NATURAL JOIN

When two tables are joined together using equality values in one or more columns, they make an equi-join.

Table aliases are used to prevent the ambiguity and the WHERE clause specifies the columns being joined.

Equi-Join or Natural Join

```
> select   EMPNO,ENAME,E.DEPTNO,D.DEPTNO,DNAME,LOC
  from   emp  e, dept d
   where  e.deptno = d.deptno;
                  OR
> SELECT EMPNO,ENAME,DEPTNO,DEPTNO,DNAME,LOC
  FROM EMP NATURAL JOIN DEPT;
```

EMPNO	ENAME	E .DEPTNO	D.DEPTNO	DNAME	LOC
1001	Ravi	10	10	EDP	Hyderabad
1002	Laxmi	20	20	STORES	Mumbai
1003	Venu	10	10	EDP	Hyderabad

OUTER JOINS

If there are any values in one table that do not have corresponding value(s) in the other table, such rows can be forcefully selected by using outer join symbol (+).

The corresponding columns for that row will have NULLs.

There are two-types of outer-joins:

1. **one-way outer join**
2. **two-way outer join (Full outer join)**

ONE-WAY OUTER JOIN

Ex. List the employee details with department names and the details of departments with no employees

```
SELECT EMPNO,ENAME, E.DEPTNO, D.DEPTNO, DNAME FROM EMP E, DEPT D
WHERE E.DEPTNO (+) = D.DEPTNO;
                        OR
SELECT ENAME, JOB, SAL,DEPTNO, DNAME
FROM EMP LEFT OUTER JOIN DEPT USING (DEPTNO);
```

EMPNO	ENAME	E.DEPTNO	D.DEPTNO	DNAME	LOC
1001	Ravi	10	10	EDP	Hyderabad
1002	Laxmi	20	20	STORES	Mumbai
1003	Venu	10	10	EDP	Hyderabad
			30	ACCOUNTS	Bangalore

Ex. List the employee details with department names and also the details of the employees with no department.

```
SELECT EMPNO,ENAME, E.DEPTNO, D.DEPTNO, DNAME FROM
EMP E, DEPT D
WHERE E.DEPTNO  = D.DEPTNO(+);
                    OR
SELECT ENAME, JOB, SAL, DEPTNO, DNAME
FROM EMP RIGHT OUTER JOIN DEPT  USING (DEPTNO);
```

Empno	EName	E.Deptno	D.Deptno	Dname	Loc
1001	Ravi	10	10	Edp	Hyd
1002	Laxmi	20	20	Stores	Mumbai
1003	Venu	10	10	Edp	Hyd
1004	Madhu	40			

FULL OUTER JOIN

```
SELECT * FROM EMP E FULL OUTER JOIN DEPT D
ON (E.DEPTNO = D.DEPTNO);
                    OR
SELECT * FROM EMP FULL OUTER JOIN  DEPT
USING (DEPTNO);
```

```
Select * from emp e left outer join dept d
        on( e.deptno = d.deptno);
```

```
Select * from emp e right outer join dept d
        on( e.deptno = d.deptno);
```

SELF JOIN

A table joining itself is called self-join.

The self join can be viewed as a join of two copies of the same table.

The table is not actually copied, but the SQL performs the command as though it were 2 different tables.

Ex. List out distinct pairs of empnos'

```
SELECT E.EMPNO, M.EMPNO
 FROM EMP E, EMP M
WHERE E.EMPNO <>  M.EMPNO AND
                          E.EMPNO < M.EMPNO;
```

SET Operators

SET operators are used to combine information of similar type from one or more than one table. Datatype of the corresponding column must be the same.

SET operators supported in Oracle :

1. UNION : Rows of first query plus rows of second query, less duplicate rows.

2. UNION ALL: Rows of first query plus rows of second query, with duplicate rows.

3. INTERSECT : Common rows from all the queries.

4. MINUS : Rows unique to the first query.

UNION:

The UNION clause merges the output of two or more queries into a single set of rows and columns.

```
SELECT <statement>
UNION
SELECT <statement> ;
```

Ex. Display the different designations in department 20 and 30

```
SELECT JOB FROM EMP WHERE DEPTNO = 20
UNION
SELECT JOB FROM EMP WHERE DEPTNO = 30;
```

NOTE: The number of columns retrieved by the first select must be equal to number of columns retrieved by the second select.
The data types of columns retrieved by all the select statements should be same.

INTERSECT:

INTERSECT operator returns the rows that are common between two sets of rows.

Ex. List the jobs common to departments 20 and 30

```
SELECT JOB FROM EMP WHERE DEPTNO =20
INTERSECT
SELECT JOB FROM EMP WHERE DEPTNO = 30;
```

MINUS:

Minus operator returns the rows unique to the first query.

Ex. List the jobs in dept. 10 but not in department 20

```
SELECT JOB FROM EMP WHERE DEPTNO  = 10
MINUS
SELECT JOB FROM EMP WHERE DEPTNO  = 20;
```

SUB QUERIES

A query within a query is called a sub query.

There are 2 types of sub queries:

1. **Non-correlated sub queries**

2. **Correlated sub queries**

Non-Correlated Sub query

- A sub query is a SELECT statement that is embedded in the where clause or having clause of outer SELECT statement.
- A non-correlated sub query executes first and its output is used to complete the query condition for the main or outer query.
- A sub query must be enclosed in parentheses

Types of non-correlated sub queries

Single-row non-correlated sub queries:
Queries that return only one row from the inner SELECT statement

Multiple-row non-correlated sub queries:
Queries that return more than one row from the inner SELECT

Single-Row Non-Correlated Sub queries

- Returns only one row value
- Use single-row comparison operators

Ex. List employee names and their jobs of the employees whose job is same as job of empno: 7369

Ex.

```
SELECT ename, job
FROM   emp
WHERE  job = (SELECT job  FROM   emp
                    WHERE  empno = 7369);
```

Single-Row Non-Correlated Sub queries

Ex. List the details of the employee who gets highest salary

```
SELECT *
FROM   emp
WHERE sal  = (SELECT max(sal)  from emp)
```

Ex. List the job with highest average salary

```
select job
from emp
group by job
having avg(sal) = (select max(avg_sal)
                   from (select avg(sal) AS avg_sal
                         from emp
                         group by job) as x );
```

Single-Row Non-Correlated Sub queries

HAVING Clause with Subqueries. You can use subqueries not only in the WHERE clause, but also in the HAVING clause. The Oracle server executes the subquery, and the results are returned into the HAVING clause of the main query .

Ex. List department whose minimum salary is greater than least salary of dept. 20

```
SELECT   deptno, MIN(sal)
FROM     emp
GROUP BY deptno
HAVING   MIN(sal) > (SELECT MIN(SAL) FROM EMP
                     WHERE DEPTNO = 20 ) ;
```

Multi-Row Non-Correlated Sub queries

- Returns more than one row
- Use multi-row comparison operators (IN, ANY, ALL)

Ex. List the names of the employees who earn lowest salary in each department.

```
SELECT ENAME, SAL, DEPTNO FROM EMP
WHERE (DEPTNO,SAL) IN (    SELECT DEPTNO,MIN(SAL)
                           FROM EMP
                           GROUP BY DEPTNO );
```

Ex. List employees who earn highest salary in each job type

```
SELECT JOB,ENAME,SAL FROM EMP
WHERE (SAL,JOB)
IN (    SELECT MAX(SAL),JOB
        FROM EMP
        GROUP BY JOB) ;
```

Multi-Row Non-Correlated Sub queries

Ex. List employees who are not clerks and whose salary is greater than that of any clerk.

```
Select * from emp
Where sal > ANY ( select distinct sal
                  from emp
                  where upper(job) = 'CLERK')
AND  upper(job) <> 'CLERK';
```

Ex. List employees who are not clerks and whose salary is greater than salary of all the clerks.

```
Select * from emp
Where sal > ALL ( select distinct sal
                  from emp
                  where upper(job) = 'CLERK' )
AND  upper(job) <> 'CLERK' ;
```

Correlated Sub-Queries

The inner query of a correlated sub-query is executed for each row of the table specified in the outer query.

Ex. List the employee details who earn salary greater than the average salary of their departments.

```
SELECT EMPNO, ENAME, SAL, DEPTNO FROM EMP E
WHERE SAL > ( SELECT AVG( SAL) FROM EMP D
                WHERE D.DEPTNO = E.DEPTNO );
```

Correlated Sub-Queries

Ex. List top 2 top earners in the company.

```
SELECT *
FROM EMP E
WHERE 2 > ( SELECT COUNT(*)
            FROM EMP D  WHERE E.SAL <  D.SAL);
```

OR

```
select * from ( select * from emp order by sal desc)
where rownum <= 2;
```

Ex. List two most recently joined employees

```
SELECT HIREDATE
FROM EMP E
WHERE 2 > ( SELECT COUNT(*) FROM EMP D
            WHERE E.HIREDATE <  D.HIREDATE);
```

END