Lab 02: Basic Data Retrieval Operations In SQL\*Plus.

**The SELECT Statement**

The SELECT statement is used in SQL to extract (retrieve) data from a database.

**Capabilities of the SELECT Statement**

1. **Selection**: This allows you to choose specific rows from a table based on certain conditions (criteria).
   * **Example 1**: To select all employees whose salary is between 3500 and 5000 and who were hired after July 31, 1981:

sql

SELECT \*

FROM EMP

WHERE SAL BETWEEN 3500 AND 5000

AND HIREDATE > TO\_DATE('31-JUL-1981', 'DD-MON-YYYY');

* + **Example 2**: To select employees whose job is either "CLERK" or "ANALYST" and who were hired between July 23, 1981, and May 14, 1982:

sql

SELECT \*

FROM EMP

WHERE (JOB = 'CLERK' OR JOB = 'ANALYST')

AND HIREDATE BETWEEN TO\_DATE('23-JUL-1981', 'DD-MON-YYYY')

AND TO\_DATE('14-MAY-1982', 'DD-MON-YYYY');

1. **Projection**: This lets you choose specific columns from a table, so you don't have to retrieve all the columns, just the ones you need.
   * **Example 1**: To select employee numbers, names, and their jobs:

sql

SELECT EMPNO, ENAME, JOB

FROM EMP;

* + **Example 2**: To select employee numbers, names, and salaries for those who do not earn commission:

sql

SELECT EMPNO, ENAME, SAL

FROM EMP

WHERE COMM IS NULL;

1. **Join**: Joins combine data from two or more tables by using a common column (usually a matching key).
   * **Example**: To retrieve the employee’s name, job, and department name by joining two tables (EMP and DEPT) where the DEPTNO (department number) matches in both tables:

sql

SELECT E.ENAME, E.JOB, D.DNAME

FROM EMP E, DEPT D

WHERE E.DEPTNO = D.DEPTNO;

**Comparison Operators**

Comparison operators are used in the WHERE or HAVING clauses to filter data based on conditions.

| **Operator** | **Meaning** |
| --- | --- |
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| <> | Not equal to |

Additional operators include:

| **Operator** | **Meaning** |
| --- | --- |
| BETWEEN ... AND | Between two values (inclusive) |
| IN (list) | Match any value in a list |
| LIKE | Match a character pattern |
| IS NULL | Checks if a value is NULL (empty) |

**Examples of Comparison Operators**

* **Example 1**: To show all employees who are **not managers**:

sql

SELECT \*

FROM EMP

WHERE JOB <> 'MANAGER';

* **Example 2**: To display the employee number, name, salary, and manager’s employee number for employees whose manager's employee number is either 7902, 7566, or 7788:

sql

SELECT EMPNO, ENAME, SAL, MGR

FROM EMP

WHERE MGR IN (7902, 7566, 7788);

* **Example 3**: To display the names of all employees whose names start with the letter "S":

sql

SELECT ENAME

FROM EMP

WHERE ENAME LIKE 'S%';

* + **Note**: % represents any sequence of characters (including none).
* **Example 4**: To display the names of employees whose second letter in their name is "A":

sql

SELECT ENAME

FROM EMP

WHERE ENAME LIKE '\_A%';

* + **Note**: \_ represents any single character, and % represents any sequence of characters.

**Logical Operators in SQL**

Logical operators are used to combine multiple conditions in a query or reverse the result of a condition. There are three main logical operators in SQL:

| **Operator** | **Meaning** |
| --- | --- |
| **AND** | Returns TRUE if **both** conditions are TRUE. |
| **OR** | Returns TRUE if **either** condition is TRUE. |
| **NOT** | Returns TRUE if the condition is FALSE. |

**Examples of Logical Operators**

* **AND Example**: Display records of all clerks who earn more than 1100.

sql

SELECT EMPNO, ENAME, JOB, SAL

FROM EMP

WHERE SAL >= 1100

AND JOB = 'CLERK';

* + Here, both conditions (salary ≥ 1100 and job is "CLERK") must be TRUE for a record to be shown.
* **OR Example**: Display records of all employees who are either clerks or earn more than 1100.

sql

SELECT EMPNO, ENAME, JOB, SAL

FROM EMP

WHERE SAL >= 1100

OR JOB = 'CLERK';

* + Here, if either the salary is ≥ 1100 **or** the job is "CLERK", the record will be shown.
* **NOT Example**: Display the names and job titles of all employees who are **not** clerks, managers, or analysts.

sql

SELECT ENAME, JOB

FROM EMP

WHERE JOB NOT IN ('CLERK', 'MANAGER', 'ANALYST');

* + This excludes employees with the job titles "CLERK", "MANAGER", and "ANALYST" from the results.

**Rules of Precedence (Order of Evaluation)**

When multiple logical operators are used in a query, SQL evaluates them in this order:

1. Comparison operators (=, >, <)
2. NOT
3. AND
4. OR

**Example of Precedence**

Consider this query:

sql

SELECT ENAME, JOB, SAL

FROM EMP

WHERE JOB = 'SALESMAN'

OR JOB = 'PRESIDENT'

AND SAL > 1500;

* This query has two conditions:
  1. Job is "SALESMAN".
  2. Job is "PRESIDENT" **and** salary is greater than 1500.

SQL will first evaluate the AND condition, so it will select employees who are either "SALESMAN" **or** employees who are "PRESIDENT" **and** earn more than 1500.

To change the order and evaluate OR first, you can use parentheses:

sql

SELECT ENAME, JOB, SAL

FROM EMP

WHERE (JOB = 'SALESMAN' OR JOB = 'PRESIDENT')

AND SAL > 1500;

* Now, only employees who are either "SALESMAN" or "PRESIDENT" and also earn more than 1500 will be shown.

**Ordering Data with ORDER BY**

SQL does not guarantee the order of results unless you explicitly specify it using the ORDER BY clause. This allows you to sort the results in **ascending** or **descending** order.

* **Ascending Order** (default):

sql

SELECT ENAME, JOB, DEPTNO, HIREDATE

FROM EMP

ORDER BY HIREDATE;

* **Descending Order**:

sql

SELECT ENAME, JOB, DEPTNO, HIREDATE

FROM EMP

ORDER BY HIREDATE DESC;

**More Examples of Ordering Data**

* **Sort by Column Alias**: You can sort data by a calculated or renamed column (alias).

sql

SELECT EMPNO, ENAME, SAL \* 12 AS ANNSAL

FROM EMP

ORDER BY ANNSAL;

* **Sort by Multiple Columns**: Sort by more than one column, with different sorting orders for each column.

sql

SELECT ENAME, DEPTNO, SAL

FROM EMP

ORDER BY DEPTNO, SAL DESC;

* + This sorts by DEPTNO in ascending order and by SAL in descending order within each department.

**Complex Example Combining Conditions and Sorting**

To display the employee number, name, job, and annual salary of all managers and clerks whose salary is between 3000 and 5500, sorted by annual salary in descending order:

sql

SELECT EMPNO, ENAME, JOB, 12 \* SAL + NVL(COMM, 0) AS ANNUAL\_SALARY

FROM EMP

WHERE JOB = 'MANAGER' OR (JOB = 'CLERK' AND SAL BETWEEN 3000 AND 5500)

ORDER BY ANNUAL\_SALARY DESC;

* This shows managers or clerks with a salary between 3000 and 5500, ordered by their annual salary from highest to lowest.

In summary, logical operators like AND, OR, and NOT allow you to combine or reverse conditions in SQL. The ORDER BY clause is used to sort the results, either in ascending or descending order.