CONDITIONAL SELECTIONS AND OPERATORS

We have two clauses used in this

- Where
- Order by

USING WHERE

select * from <table_name> where <condition>;

Types of operators used in where clause

- Arithmetic operators
- Comparison operators
- Logical operators

Arithmetic operators

Arithmetic operators

+, -, *, /

order_id	unit_price	quantity	discount
1	50	2	5
2	100	1	10
3	20	10	0
4	30	3	2

```
SELECT *
FROM orders
WHERE unit_price * quantity - discount > 50;
```

Comparison operators

- =, !=, >, <, >=, <=, <>
- between, not between
- in, not in
- null, not null
- like

Comparison operators

no	name	marks1	marks2	bonus
1	Alice	70	80	5
2	Bob	60	65	0
3	Charlie	90	85	10
4	Diana	50	45	2
5	Ethan	75	70	3
6	Fiona	40	55	0

Comparison operators

Operator	Meaning
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to (standard)
<>	Not equal to (alternative)

Comparison operators

=, !=, >, <, >=, <=, <>:-

```
select * from student where no = 2;

select * from student where no < 2;

select * from student where no > 2;

select * from student where no <= 2;

select * from student where no >= 2;

select * from student where no != 2;

select * from student where no <> 2;
```

Comparison operators

USING BETWEEN

This will gives the output based on the column and its lower bound, upperbound.

Syntax:

select * from <table_name> where <col> between <lower bound> and <upper bound>;

Ex:

SQL> select * from student where marks between 200 and 400;

Comparison operators

USING NOT BETWEEN

This will gives the output based on the column which values are not in its lower bound, upperbound.

Syntax:

select * from <table_name> where <col> not between <lower bound> and <upper bound>;

Ex:

SQL> select * from student where marks not between 200 and 400;

Comparison operators

USING IN

This will gives the output based on the column and its list of values specified.

Syntax:

select * from <table_name> where <col> in (value1, value2, value3 … valuen);

Ex:

SQL> select * from student where no in (1, 2, 3);

Comparison operators

USING NOT IN

This will gives the output based on the column which values are not in the list of values specified.

Syntax:

select * from <table_name> where <col> not in (value1, value2, value3 … valuen);

Ex:

SQL> select * from student where no not in (1, 2, 3);

Comparison operators

USING NULL

This will gives the output based on the null values in the specified column.

Syntax:

select * from <table_name> where <col> is null;

Ex:

SQL> select * from student where marks is null;

Comparison operators

USING NOT NULL

This will gives the output based on the not null values in the specified column.

Syntax:

select * from <table_name> where <col> is not null;

Comparison operators

USING LIKE

This will be used to search through the rows of database column based on the pattern you specify.

Syntax:

select * from <table_name> where <col> like <pattern>;

Ex:

This will give the rows whose marks are 100.

SQL> select * from student where marks like 100;

Comparison operators

This will give the rows whose name start with ‘S’.

SQL> select * from student where name like 'S%’;

This will give the rows whose name ends with ‘h’.

SQL> select * from student where name like '%h’;

This will give the rows whose name’s second letter start with ‘a’.

SQL> select * from student where name like '_a%';

Comparison operators

This will give the rows whose name’s third letter start with ‘d’.

```
SQL> select * from student where name like '__d%’;
```

This will give the rows whose name’s second letter start with ‘t’ from ending.

```
SQL> select * from student where name like '%_t%’;
```

This will give the rows whose name’s third letter start with ‘e’ from ending.

```
SQL> select * from student where name like '%e__%';
```

Comparison operators

This will give the rows whose name cotains 2 a’s.

```
SQL> select * from student where name like '%a% a %';
```

Logical operators

- And
- Or -- lowest precedence
- not

Comparison operators

USING AND

This will gives the output when all the conditions become true.

select * from <table_name> where <condition1> and <condition2> and .. <conditionn>;

select * from student where no = 2 and marks >= 200;

Comparison operators

USING OR

This will gives the output when either of the conditions become true.

Syntax:

select * from <table_name> where <condition1> and <condition2> or .. <conditionn>;

Ex:

SQL> select * from student where no = 2 or marks >= 200;

USING ORDER BY

This will be used to ordering the columns data (ascending or descending).

Syntax:

Select * from <table_name> order by <col> desc;

By default oracle will use ascending order.

If you want output in descending order you have to use desc keyword after the column.

select * from student order by no;

select * from student order by no desc;

COLUMN ALIASES

Syntax:

Select <orginal_col> <alias_name> from <table_name>;

Ex:

SQL> select no sno from student;

or

SQL> select no “sno” from student;

TABLE ALIASES

If you are using table aliases you can use dot method to the columns.

Syntax:

Select <alias_name>.<col1>, <alias_name>.<col2> … <alias_name>.<coln> from

<table_name> <alias_name>;

Ex:

SQL> select s.no, s.name from student s;

Functions can be categorized as follows

- Single row functions
- Group functions

SINGLE ROW FUNCTIONS

Single row functions can be categorized into five. These will be applied for each row and produces individual output for each row.

- Numeric functions
- String functions
- Date functions
- Miscellaneous functions
- Conversion functions

Group functions

Group functions operate on **sets of rows** to return **a single summary value**.

Function	Description
COUNT()	Counts rows (or non-null values)
SUM()	Total sum of numeric values
AVG()	Average value
MAX()	Maximum value
MIN()	Minimum value