# GROUP FUNCTIONS

Sum

Avg

Max

Min

Count

# GROUP FUNCTIONS

**SUM**

This will give the sum of the values of the specified column.

Syntax: sum (column)

Ex:

```
SQL> select sum(sal) from emp;
```

# GROUP FUNCTIONS

**AVG**

This will give the average of the values of the specified column.

Syntax: avg (column)

Ex:

```
SQL> select avg(sal) from emp;
```

# GROUP FUNCTIONS

**MAX**

This will give the maximum of the values of the specified column.

Syntax: max (column)

Ex:

```
SQL> select max(sal) from emp;
```

# GROUP FUNCTIONS

**MIN**

This will give the minimum of the values of the specified column.

Syntax: min (column)

Ex:

```
SQL> select min(sal) from emp;
```

# GROUP FUNCTIONS

## COUNT

This will give the count of the values of the specified column.

Syntax: count (column)

Ex:

```
SQL> select count(sal),count(*) from emp;
```

# SET OPERATORS

Union

Union all

Intersect

Minus

# SET OPERATORS

## UNION

This will combine the records of multiple tables having the same structure.

Ex:

```
SQL> select * from student1 union select * from student2;
```

# SET OPERATORS

**UNION ALL**

This will combine the records of multiple tables having the same structure but including duplicates.

Ex:

SQL> select * from student1 union all select * from student2;

INTERSECT

# SET OPERATORS

## INTERSECT

This will give the common records of multiple tables having the same structure.

Ex:

```
SQL> select * from student1 intersect select * from student2;
```

# SET OPERATORS

## MINUS

This will give the records of a table whose records are not in other tables having the same structure.

Ex:

```
SQL> select * from student1 minus select * from student2;
```

# JOINS

The purpose of a join is to combine the data across tables.

A join is actually performed by the where clause which combines the specified rows of tables.

If a join involves in more than two tables then oracle joins first two tables based on the joins condition and then compares the result with the next table and so on.

# JOINS

## TYPES

- Equi join
- Non-equi join
- Self join
- Natural join
- Cross join
- Outer join
  - Left outer
  - Right outer
  - Full outer
- Inner join

# JOINS

| emp1 | dept |
|---|---|
| ID | ID |
| 1 | 1 |
| 1 | null |
| 2 | 3 |
| null | null |
| null | null |
| 1 | null |
| 3 | 3 |

# JOINS

## EQUI JOIN

A join which contains an ‘=’ operator in the joins condition.

Ex:

```
SQL> select e.id,d.id from emp1 e,dept d where e.id=d.id;
```

## ON CLAUSE

```
SQL> select e.id,d.id from emp1 e,dept d on e.id=d.id;
```

# JOINS

## NON-EQUI JOIN

A join which contains an operator other than ‘=’ in the joins condition.

Ex:

```
select e.id,d.id from emp1 e,dept d where e.id != d.id;
```

# JOINS

## SELF JOIN

Joining the table itself is called self join.

Ex:

```
SQL> select e.id from emp1 e,emp1 e1 where e.id = e1.id;
```

# JOINS

## NATURAL JOIN

Natural join compares all the common columns.

Ex:

```
SQL> select id from emp1 natural join dept
```

# JOINS

**CROSS JOIN**

This will gives the cross product.

Ex:

```
SQL> select empno,ename,job,dname,loc from emp cross join dept;
```

# JOINS

## OUTER JOIN

Outer join gives the non-matching records along with matching records.

## LEFT OUTER JOIN

This will display the all matching records and the records which are in left hand side table those that are not in right hand side table.

Ex:

```
SQL> select e.id from emp1 e left outer join dept d on e.id = d.id;
```

# JOINS

**RIGHT OUTER JOIN**

This will display the all matching records and the records which are in right hand side table those that are not in left hand side table.

Ex:

```
SQL> select e.id from emp1 e right outer join dept d on  e.id = d.id;
```

# JOINS

**FULL OUTER JOIN**

This will display the all matching records and the non-matching records from both tables.

Ex:

```
SQL>select e.id from emp1 e full outer join dept d on  e.id = d.id;
```

# JOINS

## INNER JOIN

This will display all the records that have matched.

Ex:

```
SQL> select e.id from emp1 e inner join dept d on  e.id = d.id;
```