SQL> desc employee;

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NOT NULL NUMBER(3)

NAME VARCHAR2(20)

SALARY NUMBER(12,2)

DEPARTMENT VARCHAR2(10)

SQL> select \* from employee;

ID NAME SALARY DEPARTMENT

---------- -------------------- ---------- ----------

3 Kavya 30000 Support

4 Gopal 100000 HR

6 Amit 7000 Support

1 Dilip 50000 Admin

2 Kiran 75000 Admin

7 Kinjal 8000 Support

6 rows selected.

SQL> select department, sum(salary) from employee group by department;

DEPARTMENT SUM(SALARY)

---------- -----------

Support 45000

HR 100000

Admin 125000

SQL> select department, sum(salary), avg(salary) from employee group by department;

DEPARTMENT SUM(SALARY) AVG(SALARY)

---------- ----------- -----------

Support 45000 15000

HR 100000 100000

Admin 125000 62500

SQL> select department, sum(salary), avg(salary), min(salary), max(salary) from employee group by department;

DEPARTMENT SUM(SALARY) AVG(SALARY) MIN(SALARY) MAX(SALARY)

---------- ----------- ----------- ----------- -----------

Support 45000 15000 7000 30000

HR 100000 100000 100000 100000

Admin 125000 62500 50000 75000

SQL> select department, count(id), sum(salary), avg(salary), min(salary), max(salary) from employee group by department;

DEPARTMENT COUNT(ID) SUM(SALARY) AVG(SALARY) MIN(SALARY) MAX(SALARY)

---------- ---------- ----------- ----------- ----------- -----------

Support 3 45000 15000 7000 30000

HR 1 100000 100000 100000 100000

Admin 2 125000 62500 50000 75000

SQL> select \* from employee;

ID NAME SALARY DEPARTMENT

---------- -------------------- ---------- ----------

3 Kavya 30000 Support

4 Gopal 100000 HR

6 Amit 7000 Support

1 Dilip 50000 Admin

2 Kiran 75000 Admin

7 Kinjal 8000 Support

6 rows selected.

SQL> select id, UPPER(name), salary from employee;

ID UPPER(NAME) SALARY

---------- -------------------- ----------

3 KAVYA 30000

4 GOPAL 100000

6 AMIT 7000

1 DILIP 50000

2 KIRAN 75000

7 KINJAL 8000

6 rows selected.

SQL> select id, UPPER(name), lower(department), salary from employee;

ID UPPER(NAME) LOWER(DEPA SALARY

---------- -------------------- ---------- ----------

3 KAVYA support 30000

4 GOPAL hr 100000

6 AMIT support 7000

1 DILIP admin 50000

2 KIRAN admin 75000

7 KINJAL support 8000

6 rows selected.

SQL> CREATE TABLE class (id NUMBER(2) PRIMARY KEY, name VARCHAR2(10));

Table created.

SQL> CREATE TABLE class\_info (id NUMBER(2) PRIMARY KEY, address VARCHAR2(10));

Table created.

SQL> INSERT INTO class VALUES(1, 'abhi');

1 row created.

SQL> INSERT INTO class VALUES(2, 'adam');

1 row created.

SQL> INSERT INTO class VALUES(4, 'alex');

1 row created.

SQL> INSERT INTO class\_info VALUES(1, 'DELHI');

1 row created.

SQL> INSERT INTO class\_info VALUES(2, 'MUMBAI');

1 row created.

SQL> INSERT INTO class\_info VALUES(3, 'CHENNAI');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> select \* from class;

ID NAME

---------- ----------

1 abhi

2 adam

4 alex

SQL> select \* from class\_info;

ID ADDRESS

---------- ----------

1 DELHI

2 MUMBAI

3 CHENNAI

CROSS JOIN: Cartesian Product: This type of JOIN returns product of rows from the tables. It will return a table which consists of records which combines each row from the first table with each row of the second table

SQL> SELECT \* FROM class CROSS JOIN class\_info;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

1 abhi 2 MUMBAI

1 abhi 3 CHENNAI

2 adam 1 DELHI

2 adam 2 MUMBAI

2 adam 3 CHENNAI

4 alex 1 DELHI

4 alex 2 MUMBAI

4 alex 3 CHENNAI

9 rows selected.

INNER JOIN: EQUI JOIN: This is a Simple Join in which the result is based on matched data as per the equality condition specified in SQL Query

SQL> SELECT \* FROM class INNER JOIN class\_info ON class.id = class\_info.id;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

2 adam 2 MUMBAI

NATURAL JOIN: This is a type of INNER JOIN which is based on Column having same name and same datatype present in both the tables

SQL> SELECT \* FROM class INNER JOIN class\_info ON class.id = class\_info.id;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

2 adam 2 MUMBAI

SQL> SELECT \* FROM class NATURAL JOIN class\_info;

ID NAME ADDRESS

---------- ---------- ----------

1 abhi DELHI

2 adam MUMBAI

OUTER JOIN: Outer Join is based on both matched & unmatched data

* LEFT OUTER JOIN
* RIGHT OUTER JOIN
* FULL OUTER JOIN

LEFT OUTER JOIN: The Left Outer Join returns Table with the matched data from the two tables and then the remaining rows of the LEFT Table and the null values from the RIGHT Table

SQL> SELECT \* FROM class LEFT OUTER JOIN class\_info ON class.id = class\_info.id;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

2 adam 2 MUMBAI

4 alex

RIGHT OUTER JOIN: The Right Outer Join returns Table with the matched data from the two tables and then the remaining rows of the RIGHT Table and the null values from the LEFT Table

SQL> SELECT \* FROM class RIGHT OUTER JOIN class\_info ON class.id = class\_info.id;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

2 adam 2 MUMBAI

3 CHENNAI

FULL OUTER JOIN: The Full Outer Join returns Table with the matched data of two table then remaining rows of both First the LEFT Table and then the RIGHT Table

SQL> SELECT \* FROM class FULL OUTER JOIN class\_info ON class.id = class\_info.id;

ID NAME ID ADDRESS

---------- ---------- ---------- ----------

1 abhi 1 DELHI

2 adam 2 MUMBAI

3 CHENNAI

4 alex

SQL> CREATE TABLE First (id NUMBER(3) PRIMARY KEY, name VARCHAR2(15));

Table created.

SQL> CREATE TABLE Second (id NUMBER(3) PRIMARY KEY, name VARCHAR2(15));

Table created.

SQL> INSERT INTO First VALUES (1, 'abhi');

1 row created.

SQL> INSERT INTO First VALUES (2, 'adam');

1 row created.

SQL> INSERT INTO SECOND VALUES (2, 'adam');

1 row created.

SQL> INSERT INTO SECOND VALUES (3, 'Chester');

1 row created.

SQL> select \* from FIRST;

ID NAME

---------- ---------------

1 abhi

2 adam

SQL> select \* from SECOND;

ID NAME

---------- ---------------

2 adam

3 Chester

SET:

* UNION
* UNION ALL
* INTERSECT
* MINUS

UNION: Union is used to combine the results of 2 or more SELECT Statements

SQL> SELECT \* FROM first UNION SELECT \* FROM second;

ID NAME

---------- ---------------

1 abhi

2 adam

3 Chester

UNION ALL: Similar to UNION but also shows Duplicate Rows

SQL> SELECT \* FROM first UNION ALL SELECT \* FROM second;

ID NAME

---------- ---------------

1 abhi

2 adam

2 adam

3 Chester

INTERSECT: Intersect is used to combine the results of 2 SELECT Statements, but it only returns the records which are Common among both Tables

SQL> SELECT \* FROM first INTERSECT SELECT \* FROM second;

ID NAME

---------- ---------------

2 adam

MINUS: The Minus Operation combines results of 2 SELECT Statements and return only those in the final result, which belongs to the first set of the result

SQL> SELECT \* FROM first MINUS SELECT \* FROM second;

ID NAME

---------- ---------------

1 abhi