**Tables Used are –**

1. **Employee\_Info:** Contains employee information.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | | **Constraint** | | **Description** |
| Employee\_Id | Number (5) | | Primary key | | Employee Identification No |
| Employee\_Name | Varchar2 (10) | | Not null | | Employee name |
| DoJ | Date | |  | | Employee Date of joining |
| Employee\_Address | Varchar2 (15) | |  | | employee address |
| Phone\_Num | Number (10) | |  | | Employee\_phone No |
| Salary | Number (10) | |  | | Salary earned by employee |
|  | |  | |  | | |

1. **Department\_info:** Contains details of Department.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** | **Description** |
| Depart\_Id | Number (5) | Primary key | Department Identification No |
| Department\_Name | Varchar2 (10) | Not null | Department Name |
| Company\_Address | Varchar2 (20) |  | Company Address |
| Employee\_id | Number (5) | Foreign key | Employee Identification No |

**Questions**

1. Create tables named employee\_info and department\_info with the structure as provided above. Add employee and department details (min 5 rows to be inserted). Write a query to Join both the tables and display information in both the tables. [Employee\_Info, Department\_info] [3 marks]

**Sample output:**

Employee\_id Employee\_Name DoJ Employee\_address Phone\_Num Salary Dept\_id Dept\_name Company\_Address Employee\_id

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

110 Amit 12/12/10 E-7 778888888 45000 D111 HR H-7 110

ANS:

**TABLE CREATION:**

create table Employee\_Info

(

Employee\_Id number(5) constraint emp\_id\_p\_key primary key,

Employee\_Name varchar2(10),

DOJ Date,

Employee\_Address varchar2(15),

Phone\_Num number(10),

Salary number(10)

);

**INSERT FOR THE TABLE:**

Insert into Employee\_Info values(101, 'Amit',TO\_DATE('2010/12/12','yyyy/mm/dd'),'E\_Address\_1', 123456789, 54678);

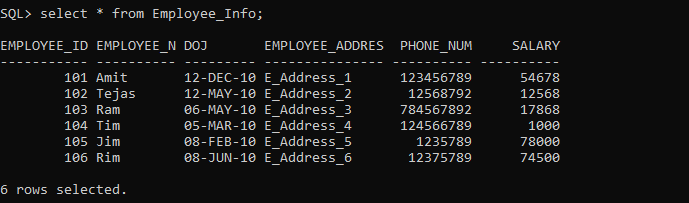
insert into Employee\_Info values(102, 'Tejas',TO\_DATE('2010/05/12','yyyy/mm/dd'),'E\_Address\_2', 12568792, 12568);

insert into Employee\_Info values(103, 'Ram',TO\_DATE('2010/05/06','yyyy/mm/dd'),'E\_Address\_3', 784567892, 17868);

insert into Employee\_Info values(104, 'Tim',TO\_DATE('2010/03/05','yyyy/mm/dd'),'E\_Address\_4', 124566789, 1000);

insert into Employee\_Info values(105, 'Jim',TO\_DATE('2010/02/08','yyyy/mm/dd'),'E\_Address\_5', 1235789, 78000);

insert into Employee\_Info values(106, 'Rim',TO\_DATE('2010/06/08','yyyy/mm/dd'),'E\_Address\_6', 12375789, 74500);



**TABLE CREATION:**

create table Department\_info

(

Depart\_Id number(5) constraint dept\_id\_p\_key primary key,

Department\_Name varchar2(10),

Company\_Address varchar2(20),

Employee\_id number(5)

);

**INSERT FOR THE TABLE:**

insert into Department\_info values(104, 'D101', 'C\_Address\_1',102);

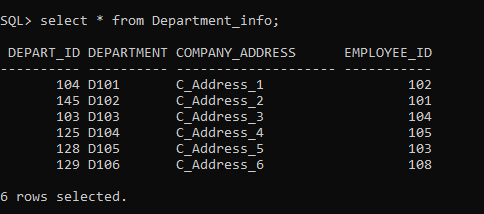
insert into Department\_info values(145, 'D102', 'C\_Address\_2',101);

insert into Department\_info values(103, 'D103', 'C\_Address\_3',104);

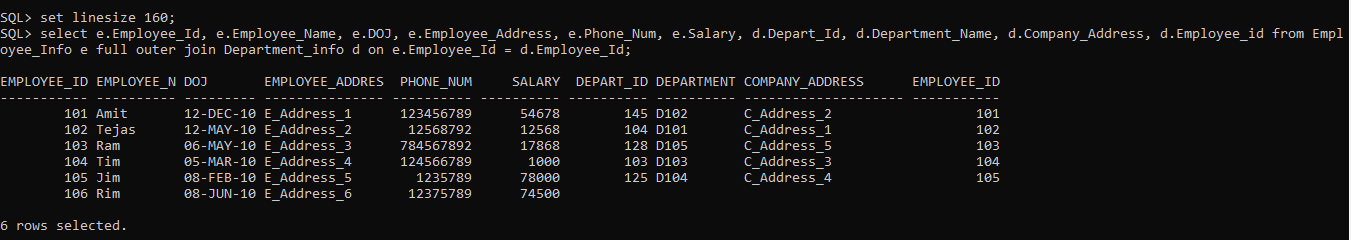
insert into Department\_info values(125, 'D104', 'C\_Address\_4',105);

insert into Department\_info values(128, 'D105', 'C\_Address\_5',103);

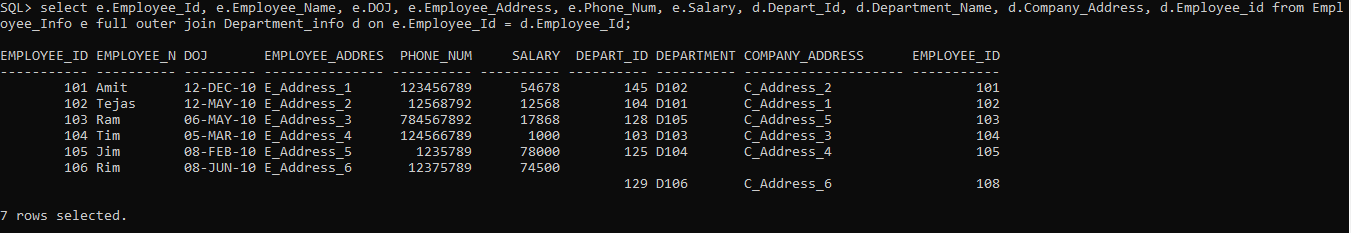
insert into Department\_info values(129, 'D106', 'C\_Address\_6',108);



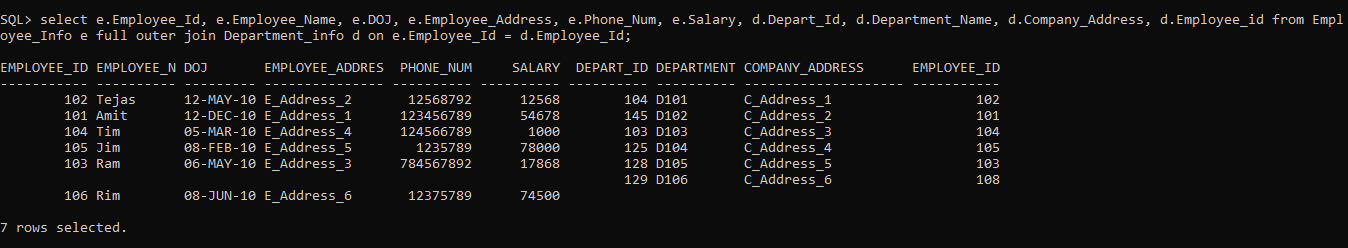
select e.Employee\_Id, e.Employee\_Name, e.DOJ, e.Employee\_Address, e.Phone\_Num, e.Salary, d.Depart\_Id, d.Department\_Name, d.Company\_Address, d.Employee\_id from Employee\_Info e full outer join Department\_info d on e.Employee\_Id = d.Employee\_Id;



Added 1 more column in right table to see the difference between the outer and left join



Both are same output



1. Write a query to display all the records from the left table and from right table only the matched records (Left table: Employee\_info and right table: Department\_info): (4 marks)

**Sample output:**

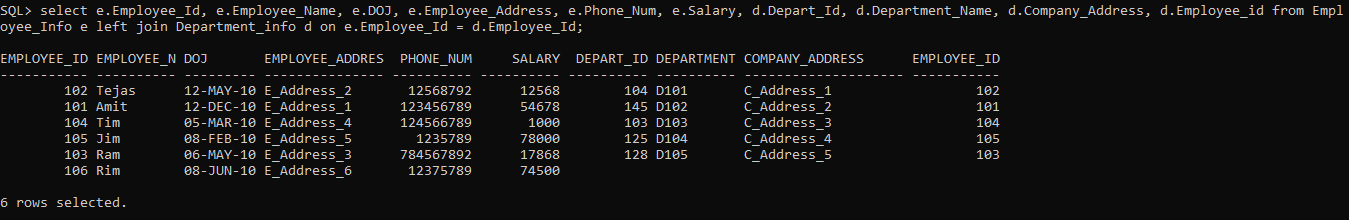
Employee\_id Employee\_Name DoJ Employee\_address Dept\_id Dept\_name

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

112 Raghu 10/12/18 J-7 ….. F101 Finance

ANS:

select e.Employee\_Id, e.Employee\_Name, e.DOJ, e.Employee\_Address, e.Phone\_Num, e.Salary, d.Depart\_Id, d.Department\_Name, d.Company\_Address, d.Employee\_id from Employee\_Info e left join Department\_info d on e.Employee\_Id = d.Employee\_Id;



1. Write a query to return all records/rows from the right table and from left table returns only matched records (Left table: Employee\_info and right table: Department\_info): (4 marks)

**Sample output:**

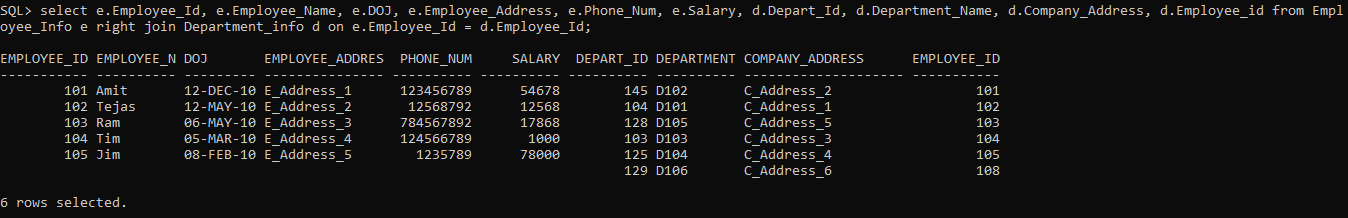
Dept\_id Dept\_name Company\_Address Employee\_id Employee\_name

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

H111 HR H-123,JP nagar 113 Stephen

ANS:

select e.Employee\_Id, e.Employee\_Name, e.DOJ, e.Employee\_Address, e.Phone\_Num, e.Salary, d.Depart\_Id, d.Department\_Name, d.Company\_Address, d.Employee\_id from Employee\_Info e right join Department\_info d on e.Employee\_Id = d.Employee\_Id;



1. Normalization helps to reduce redundancy and complexity. Justify the statement explaining 1NF,2NF,3NF with examples. (10 marks)

ANS:

The Database is the most important part of the project and entire structure has to be handled systematically and the data as well. As the data is perfect or handled properly for any system then it is said that the 70% of the solutions get resolved. For storing the data, databases have to be chosen then the data has to be organized in the database.

So, the entire data process will be organized using the different DB methods. One of them is normalization. Normalization is process through which the data can be organized to avoid or reduce the data redundancy and anomalies of all sorts. It aids in the division of large database tables into smaller ones and the establishment of relationships between them. It has the ability to delete redundant data and is simple to incorporate, manipulate and delete. There are 3 types of anomalies found when we don’t follow the normalization and they are insert, delete, update anomalies.

Six different forms:

* 1NF
* 2NF
* 3NF
* BCNF
* 4NF
* 5NF

**1NF** – First Normal form: According to the first rule, the data in any column must have an atomic value (single value) only, i.e., the column attribute cannot hold multiple values in any table.

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **Knowledge** |
| John | Wick | C++, JAVA |
| Sam | Curran | Python |
| Huge | Philips | JAVA, PYTHON |

The above e.g., is not 1NF as the knowledge column has more than one value. So, we need to convert it to 1NF.

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **Knowledge** |
| John | Wick | C++ |
| John | Wick | JAVA |
| Sam | Curran | Python |
| Huge | Philips | JAVA |
| Huge | Philips | PYTHON |

**2NF** – To perform 2NF the table has to be in 1NF (First Normal form). The non-prime attribute of the table should not be dependent on the subset of any candidate key of the table. The non-prime attribute is defined as the column attribute which are not part of the candidate key.

|  |  |  |
| --- | --- | --- |
| **StudentID** | **StudentName** | **ProjectName** |
| S01 | John | Smart City |
| S10 | Jacob | R&D |
| S03 | Alan | Machine connectivity |
| S08 | Alex | R&D |

ProjectName is a non-prime attribute as it completely dependent on the STUDENTID as the student will have their project assigned to them and they are totally related or dependent. This has to be converted to 2NF. It is already in 1NF as there are no multiple values for any columns.

The below table will be created for the 2NF

|  |  |  |
| --- | --- | --- |
| **StudentID** | **StudentName** | **ProjectID** |
| S01 | John | P05 |
| S10 | Jacob | P04 |
| S03 | Alan | P12 |
| S08 | Alex | P04 |

|  |  |
| --- | --- |
| **ProjectID** | ProjectName |
| P05 | Smart City |
| P04 | R&D |
| P12 | Machine Connectivity |

**3NF** – To implement 3NF the table has to be in 2NF, else we need to convert it to 2NF first and then proceed. It shouldn’t hold the transitive functional dependency. Transitive dependency can be explained like If there are 3 variable a,b,c and a is dependent on b (also called as functional dependency), b is dependent on c so the a will be dependent on c. We should work on to remove these transitive dependencies.

|  |  |  |  |
| --- | --- | --- | --- |
| U\_ID | U\_Name | U\_Ac | Bank |
| 111 | Sejal | 20156 | BOI |
| 256 | Harry | 12560 | SBI |
| 335 | Don | 12480 | SBI |
| 258 | Linn | 12459 | J&K |
| 489 | Sam | 56879 | KMB |

U\_Name, U\_Ac, Bank all are functionally dependent on U\_ID.

Candidate key for the above table – U\_ID, U\_Name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| U\_ID | U\_Name | U\_Ac | Bank\_Code | Bank |
| 111 | Sejal | 20156 | BOI01 | BOI |
| 256 | Harry | 12560 | SBI01 | SBI |
| 335 | Don | 12480 | SBI01 | SBI |
| 258 | Linn | 12459 | J&K01 | J&K |
| 489 | Sam | 56879 | KMB01 | KMB |

The above table is in 1NF and 2NF but there is a transitive dependency of Bank\_Code on Bank. To fix this we will create a separate table for the Bank.

|  |  |  |  |
| --- | --- | --- | --- |
| U\_ID | U\_Name | U\_Ac | Bank\_Code |
| 111 | Sejal | 20156 | BOI01 |
| 256 | Harry | 12560 | SBI01 |
| 335 | Don | 12480 | SBI01 |
| 258 | Linn | 12459 | J&K01 |
| 489 | Sam | 56879 | KMB01 |

|  |  |
| --- | --- |
| Bank\_Code | Bank |
| BOI01 | BOI |
| SBI01 | SBI |
| J&K01 | J&K |
| KMB01 | KMB |

There are 3 more rules for the normalization which are BCNF, 4NF and 5NF. And all these will help to reduce the data redundancy and anomalies like insert, delete and update.

1. Create tables Course\_info,student\_info using the table structure as given below and details in course\_info,student\_info tables (min 5 rows to be inserted) and perform the following operations:

**.**

**Courses\_info:** Contains the details of various courses.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** | **Description** |
| Course\_Id | Number (5) | Primary key | Course Identification No |
| Course\_Name | Varchar2 (20) | Not null | Course Name |
| Course\_details | Varchar2 (30) |  | Course Description |
|  |  |  |  |

Student\_Info

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** | **Description** |
| Rollno | Number (4) |  | Student Rollno |
| Marks | Number | Not null | Marks scored by student |
| Student\_address | Number (5) |  | Student address |
| Course\_Id | Number (5) | Foreign key | Course Identification No |
| Student\_name | Varchar(15) |  | Student’s name |

1. Combine the tables and do not display the duplicates ( 2 marks.)

Sample output:

Course-id Coursename Course-details Roll no Marks Student\_Name Students\_Address

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

M123 Masters MCA 113 85 Anita Rao J-56

ANS: Assumpiton: Combine will help us to merge to tables where both the table should have same schema followed. Combining a table is using UNION concept in DBs. To achieve the solution, we have assumed the Student\_Info as the table 1 with schema as Rollno, Marks, Student\_address, Course\_Id, Student\_name and the next table as Student\_Info1 with the same schema. So that we can combine both the tables.

**QUERY:**

**TABLE CREATION:**

create table Student\_Info

(

Rollno number(4),

Marks number(3),

Student\_address varchar(10),

Course\_Id number(5),

Student\_name varchar(15)

);

**INSERT INTO TABLE:**

insert into Student\_Info values (0001,89, 'ST\_Addr\_1', 11, 'Linn');

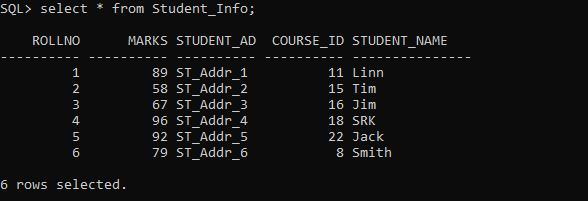
insert into Student\_Info values (0002,58, 'ST\_Addr\_2', 15, 'Tim');

insert into Student\_Info values (0003,67, 'ST\_Addr\_3', 16, 'Jim');

insert into Student\_Info values (0004,96, 'ST\_Addr\_4', 18, 'SRK');

insert into Student\_Info values (0005,92, 'ST\_Addr\_5', 22, 'Jack');

insert into Student\_Info values (0006,79, 'ST\_Addr\_6', 08, 'Smith');



**TABLE CREATION:**

create table Student\_Info1

(

Rollno number(4),

Marks number(3),

Student\_address varchar(10),

Course\_Id number(5),

Student\_name varchar(15)

);

**INSERT INTO TABLE:**

insert into Student\_Info1 values (0002,58, 'ST\_Addr\_2', 15, 'Tim');

insert into Student\_Info1 values (0007,97, 'ST\_Addr\_7', 35, 'Meril');

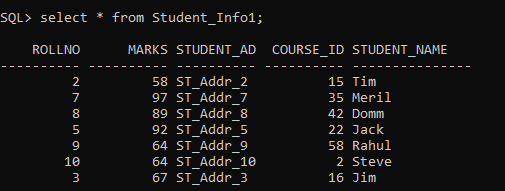
insert into Student\_Info1 values (0008,89, 'ST\_Addr\_8', 42, 'Domm');

insert into Student\_Info1 values (0005,92, 'ST\_Addr\_5', 22, 'Jack');

insert into Student\_Info1 values (0009,64, 'ST\_Addr\_9', 58, 'Rahul');

insert into Student\_Info1 values (0010,64, 'ST\_Addr\_10', 02, 'Steve');

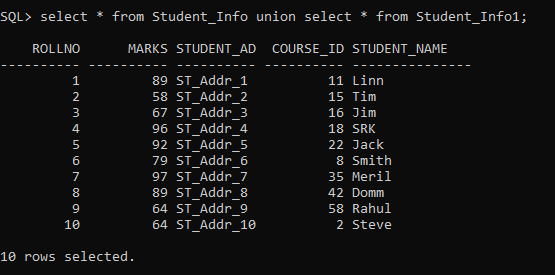
insert into Student\_Info1 values (0003,67, 'ST\_Addr\_3', 16, 'Jim');



**COMBINING QUERY:**

select \* from Student\_Info union select \* from Student\_Info1;

**OUTPUT:**



1. Combine both tables and Display all students enrolled for Course\_details (MBA) and have scored marks above 85 (3 marks.)

Sample output:

Rollno Student\_Name Student\_address Course\_Id Marks Course\_Name Course\_details

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

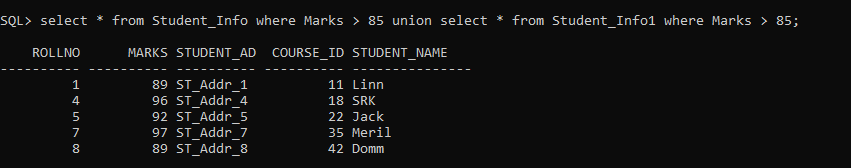
114 Bina Verma B-7, Jayanagar MB1 89 MBA Masters

118 Nimita Singh V-7 Noida MB1 87 MBA Masters

ANS:

**QUERY:**

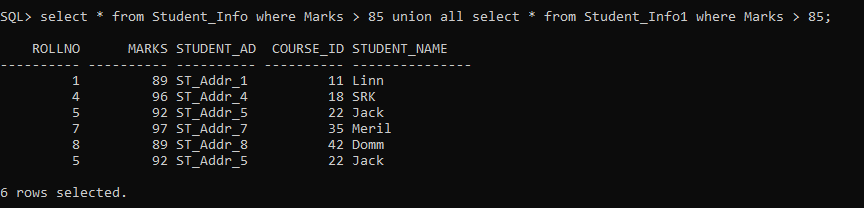
select \* from Student\_Info where Marks > 85 union select \* from Student\_Info1 where Marks > 85;



Above query will return without the duplicate keys and where conditions which is the actual output as per the ques.

But if we want to get the entire data i.e. repetitive data as well then use the below command

select \* from Student\_Info where Marks > 85 union all select \* from Student\_Info1 where Marks > 85;



Also, if we want to have the Course\_Name in the schema and while performing a query then below code and new table can be used. Course\_Name has been taken as MBA as values as respect to the sample output given in the ques considered.

**TABLE CREATION:**

create table Student\_Info2

(

Rollno number(4),

Marks number(3),

Student\_address varchar(10),

Course\_Id number(5),

Course\_details varchar(30),

Course\_Name Varchar2 (20),

Student\_name varchar(15)

);

**INSERT INTO TABLE:**

insert into Student\_Info2 values (0001,89, 'ST\_Addr\_1', 11, 'Eng', 'BTECH', 'Linn');

insert into Student\_Info2 values (0002,58, 'ST\_Addr\_2', 15, 'Arts', 'MBA', 'Meril');

insert into Student\_Info2 values (0003,67, 'ST\_Addr\_3', 16, 'BA Course 1', 'MBA', 'Domm');

insert into Student\_Info2 values (0004,96, 'ST\_Addr\_4', 18, 'BA Course 2', 'MBA', 'Jack');

insert into Student\_Info2 values (0005,92, 'ST\_Addr\_5', 22, 'BA Course 3', 'MBA', 'Steve');

insert into Student\_Info2 values (0006,79, 'ST\_Addr\_6', 08, 'Arts B Course 1', 'BA', 'Smith');

**TABLE CREATION:**

create table Student\_Info3

(

Rollno number(4),

Marks number(3),

Student\_address varchar(10),

Course\_Id number(5),

Course\_details varchar(30),

Course\_Name Varchar2 (20),

Student\_name varchar(15)

);

**INSERT INTO TABLE:**

insert into Student\_Info3 values (0002,58, 'ST\_Addr\_2', 15, 'Arts', 'MBA', 'Meril');

insert into Student\_Info3 values (0007,97, 'ST\_Addr\_7', 38, 'Arts', 'MBA', 'Tim');

insert into Student\_Info3 values (0008,89, 'ST\_Addr\_8', 26, 'Arts', 'MBA', 'Jim');

insert into Student\_Info3 values (0005,92, 'ST\_Addr\_5', 22, 'BA Course 3', 'MBA', 'Steve');

insert into Student\_Info3 values (0009,64, 'ST\_Addr\_9', 19, 'Arts', 'MBA', 'Rahul');

insert into Student\_Info3 values (0010,82, 'ST\_Addr\_10', 68, 'Commerce', 'BCOM', 'Don');

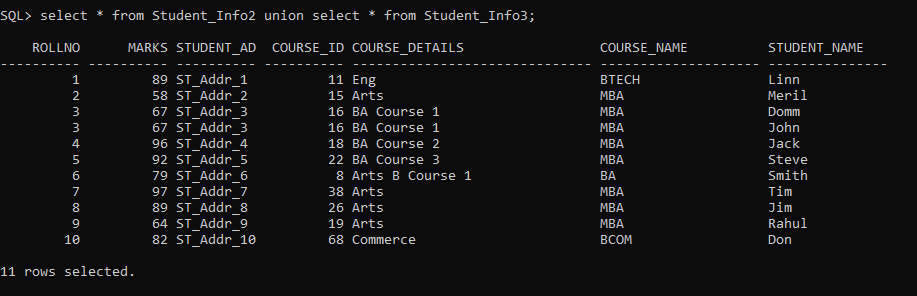
insert into Student\_Info3 values (0003,67, 'ST\_Addr\_3', 16, 'BA Course 1', 'MBA', 'John');

To Display all the data except duplicate:

**COMBINING QUERY:**

select \* from Student\_Info2 union select \* from Student\_Info3;

**OUTPUT:**

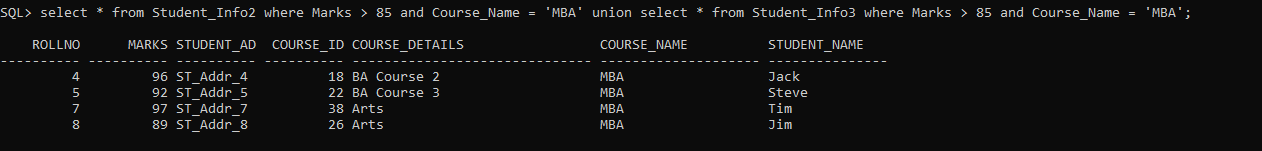


To Display the records based on the condition added i.e. MBA as Course\_Name

**QUERY:**

select \* from Student\_Info2 where Marks > 85 and Course\_Name = 'MBA' union select \* from Student\_Info3 where Marks > 85 and Course\_Name = 'MBA';

**OUTPUT:**



1. Write a procedure to find maximum of 2 numbers. (4 marks)

OR

Write the difference between procedures and functions with two examples

Finding the maximum of the two numbers:

Using the below code to show the output of PROCEDURE else it won’t show:

set serveroutput on;

WRINTING PROCEDURE TO FIND THE MAXIMUM OF THE 2 NUMBERS

CODE:

DECLARE

a1 number;

b1 number;

c1 number;

PROCEDURE findMax(x1 IN number, y1 IN number, z OUT number) IS

BEGIN

IF x1>y1 THEN

z:=x1;

Else

z:=y1;

END IF;

END;

BEGIN

a1 := 102;

b1 := 68;

findMax(a1,b1,c1);

dbms\_output.put\_line('Maximum of the two given number is :' ||c1);

END;

/

**OUTPUT:**

