

## BHARATI VIDYAPEETH’S

**INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT**

(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)

**Database Management**

**Systems (MCA-165)**

**Practical File**

## Submitted To: Submitted By:

Dr. Vaishali Joshi Sachin Rana (02511604421) (Assistant Professor) MCA 1st Sem, Sec 1

**INDEX**

|  |  |  |
| --- | --- | --- |
|  | **LAB ASSIGNMENT** | **PAGE NO.** |
| **I** | LAB ASSIGNMENT I | **3** |
| **II** | LAB ASSIGNMENT II | **10** |
| **III** | LAB ASSIGNMENT III (JOINS) | **17** |
| LAB ASSIGNMENT III (JOINS) (Part2) | **24** |
| **IV** | LAB ASSIGNMENT IV | **29** |
| **V** | LAB ASSIGNMENT V (PL/SQL) | **36** |
| LAB ASSIGNMENT V (Use of Cursors) | **42** |
| LAB ASSIGNMENT V (Cursor and Error Handling) | **46** |
| LAB ASSIGNMENT V (Procedures and Functions) | **48** |
| LAB ASSIGNMENT V (Trigger) | **52** |

# LAB ASSIGNMENT – I

#### Table : Book

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data type** | **Size** | **Constraint** |
| Book\_Code | Varchar2 | 5 | Primary Key |
| ISBN No. | Varchar2 | 8 | Not Null/ |
| Book Name | Varchar2 | 14 | Unique |
| Publisher | Varchar2 | 10 |  |
| Price | Number | 5,2 | >=100 |
| Author Name | Varchar2 | 14 |  |
| Date\_of\_Launch | Date |  |  |

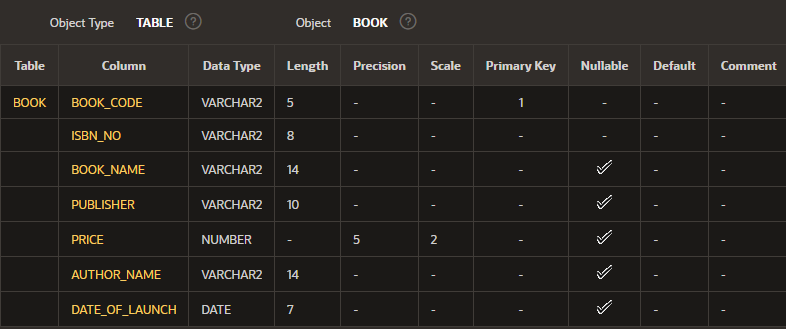
***Table Creation Query-***

CREATE TABLE Book(

Book\_Code VARCHAR2(5) PRIMARY KEY, ISBN\_No VARCHAR(8) NOT NULL, Book\_Name VARCHAR2(14) UNIQUE,

Publisher VARCHAR2(10),

Price NUMBER(5,2) CHECK (Price>=100),

Author\_Name VARCHAR2(14), Date\_of\_Launch DATE);

**Write queries to perform the following:**

#### 1. Insert a few records in it.

***Sql Query-***

INSERT INTO Book VALUES ('M138', 'IB0013', 'Intro to C#', 'TMH', 500.00, 'Yashwant K', '04/25/2003');

INSERT INTO Book VALUES ('M272', 'IB1045', 'Graphics with C', 'PHI', 650.00, 'F Campbell', '09/20/2006');

INSERT INTO Book VALUES ('M106', 'IB0024', 'JAVA Basics', 'TMH', 920.50, 'Brie Avant', '01/15/1999');

INSERT INTO Book VALUES ('M210', 'IB1048', 'Kotlin Guide', 'PHI', 200.00, 'Mason G Tam', '06/30/2015');

INSERT INTO Book VALUES ('M167', 'IB0061', 'CPP Coding', 'Pearson', 750.25, 'Jean Arrot', '07/13/1998');

INSERT INTO Book VALUES ('M236', 'IB1031', 'JAVA with VS', 'BPB', 800.55, 'Brie Avant', '03/15/2008');

INSERT INTO Book VALUES ('M123', 'IB0032', 'Graphics in C#', 'Pearson', 750.25, 'K Dibiasky', '02/14/2005');

INSERT INTO Book VALUES ('M242', 'IB1074', 'Python', 'BPB', 980.55, 'Juon Orlean', '09/27/2007');

INSERT INTO Book VALUES ('M119', 'IB0038', 'SQL Basics', 'TMH', 800.00, 'Yule Drask', '06/26/2003');

INSERT INTO Book VALUES ('M289', 'IB1027', 'Database Guide', 'Pearson', 475.25, 'Yule Drask', '05/21/2010');

***Results-***

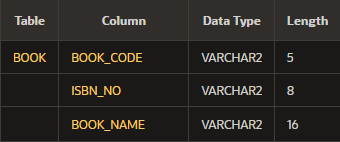
#### —----------------------------------

**Q2. Increase the size of the Book\_Name field to 16 characters.**

***Sql Query-***

ALTER TABLE Book

MODIFY Book\_Name VARCHAR2(16);

***Results-***

#### —----------------------------------

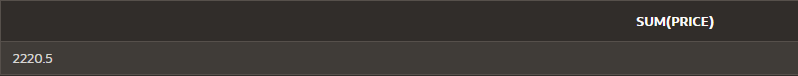
**Q3. Display the total price of all the books of “TMH’ publishing house.**

***Sql Query-***

SELECT SUM(Price)

FROM Book

WHERE Publisher = 'TMH';

***Results-***

#### —----------------------------------

**Q4. Display the details of all the books of author “Yashwant Kanetkar” and “Yule Drask”.**

***Sql Query-***

SELECT \* FROM Book

WHERE Author\_Name IN ('Yashwant K','Yule Drask');

***Results-***

#### —----------------------------------

**Q5. Delete all the books belonging to “PHI” publisher.**

***Sql Query-***

DELETE FROM Book

WHERE Publisher = 'PHI';

***Results-***



#### —----------------------------------

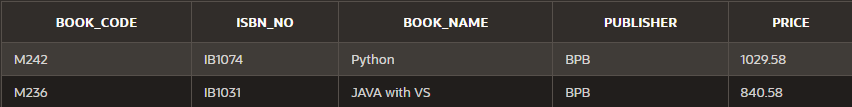
**Q6. Update the price of books by 5% which belong to ‘BPB’ publishers.**

***Sql Query-***

UPDATE Book

SET Price = ((0.05\*Price) + Price) WHERE Publisher = 'BPB';

***Results-***



#### —----------------------------------

**Q7. Create a new table “Author” with author details such as author id (primary key), name of the author, subject, contact details, research area etc.**

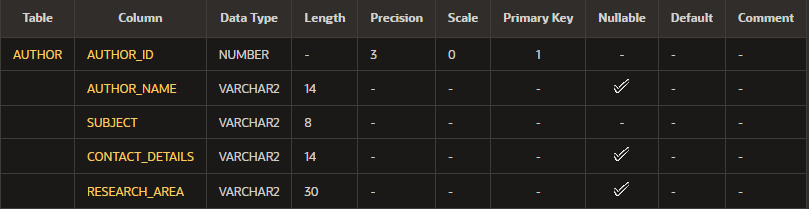
***Sql Query-***

CREATE TABLE Author(

Author\_ID NUMBER(3) PRIMARY KEY,

Author\_Name VARCHAR2(14), Subject VARCHAR(8) NOT NULL,

Contact\_Details VARCHAR2(14), Research\_Area VARCHAR2(30));

***Results-***

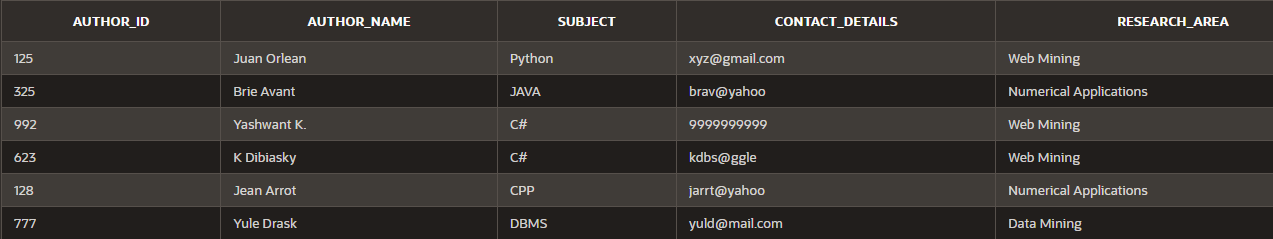
#### —----------------------------------

***Inertion in Author Table-***

INSERT INTO Author VALUES(992, 'Yashwant K.', 'C#', '9999999999', 'Web Mining');

INSERT INTO Author VALUES(125, 'Juan Orlean', 'Python', 'xyz@gmail.com', 'Web Mining'); INSERT INTO Author VALUES(325, 'Brie Avant', 'JAVA', 'brav@yahoo', 'Numerical Applications');

INSERT INTO Author VALUES(777, 'Yule Drask', 'DBMS', 'yuld@mail.com', 'Data Mining'); INSERT INTO Author VALUES(623, 'K Dibiasky', 'C#', 'kdbs@ggle', 'Web Mining');

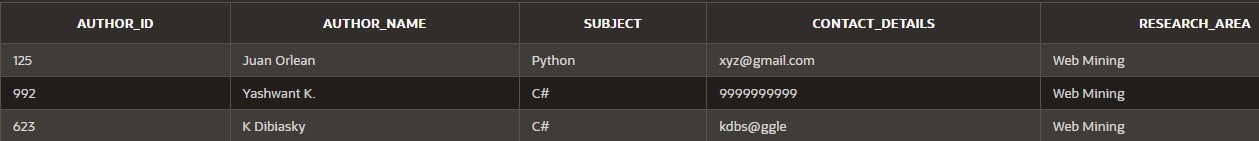
INSERT INTO Author VALUES(128, 'Jean Arrot', 'CPP', 'jarrt@yahoo', 'Numerical Applications');

#### Q8. Insert records in it and display the records of those authors who have ‘Web mining’ as their research area.

***Sql Query-***

SELECT \* FROM Author

WHERE Research\_Area = 'Web Mining';

***Results-***

#### —----------------------------------

**Q9. Add a new column “Author\_Id” in “Book” table and make it a foreign key on “author id” of “Author” table.**

***Sql Query-***

ALTER TABLE Book

ADD Author\_ID REFERENCES Author(Author\_ID); SELECT \* FROM Book;

***Results-***

#### —----------------------------------

**Q10. Update all the previous records of Book table to add information of author id (newly added column)**

***Sql Query-***

UPDATE Book SET Author\_ID = 992 WHERE Author\_Name = 'Yashwant K'; UPDATE Book SET Author\_ID = 125 WHERE Author\_Name = 'Juon Orlean'; UPDATE Book SET Author\_ID = 325 WHERE Author\_Name = 'Brie Avant'; UPDATE Book SET Author\_ID = 623 WHERE Author\_Name = 'K Dibiasky'; UPDATE Book SET Author\_ID = 128 WHERE Author\_Name = 'Jean Arrot'; UPDATE Book SET Author\_ID = 777 WHERE Author\_Name = 'Yule Drask'; UPDATE Book SET Author\_ID = 992 WHERE Author\_Name = 'Yashwant K';

***Results-***

#### —----------------------------------

**Q11. Display the records of the books where book name starts with ‘C’ or ‘G’ or ‘I’.**

***Sql Query-***

SELECT \* FROM Book

WHERE Book\_Name LIKE 'C%' OR Book\_Name LIKE 'G%'

OR Book\_Name LIKE 'I%';

***Results-***

#### —----------------------------------

**Q12. Create a view for the user to access Book Code, Name and Price of books which are published by ‘Pearson Education’**

***Sql Query-***

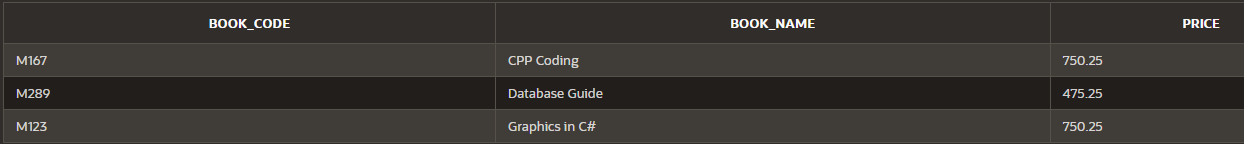
CREATE VIEW PearsonV AS

SELECT Book\_Code, Book\_Name, Price FROM Book

WHERE Publisher = 'Pearson';

SELECT \* FROM PearsonV;

***Results-***



**—----------------------------------**

# LAB ASSIGNMENT – II

#### Table : Student

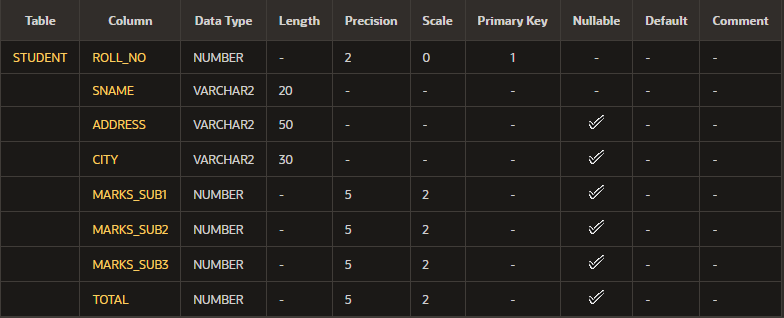
|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data type** | **Size** | **Constraint** |
| Roll\_No | Number | 2 | Primary Key |
| SName | Varchar2 | 20 | Not Null |
| Address | Varchar2 | 50 |  |
| City | Varchar2 | 30 |  |
| Marks\_Sub1 | Number | 5,2 | >=0 and <=100 |
| Marks\_Sub2 | Number | 5,2 | >=0 and <=100 |
| Marks\_Sub3 | Number | 5,2 | >=0 and <=100 |
| Total | Number | 5,2 | >=0 and <=300 |

***Table Creation Query-***

CREATE TABLE Student(

Roll\_No NUMBER(2) PRIMARY KEY, SName VARCHAR2(20) NOT NULL, Address VARCHAR2(50),

City VARCHAR2(30),

Marks\_Sub1 NUMBER(5,2) CHECK (Marks\_Sub1>=0 AND Marks\_Sub1<=100), Marks\_Sub2 NUMBER(5,2) CHECK (Marks\_Sub2>=0 AND Marks\_Sub2<=100), Marks\_Sub3 NUMBER(5,2) CHECK (Marks\_Sub3>=0 AND Marks\_Sub3<=100), Total NUMBER(5,2) CHECK (Total>=0 AND Total<=300));

**Write SQL queries for :**

#### Q1. Insert few records except total marks.

***SQL Query-***

INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (12, 'Sachin', 'Flat No 324 ABC Apartment', 'New Delhi', 89, 90.50,

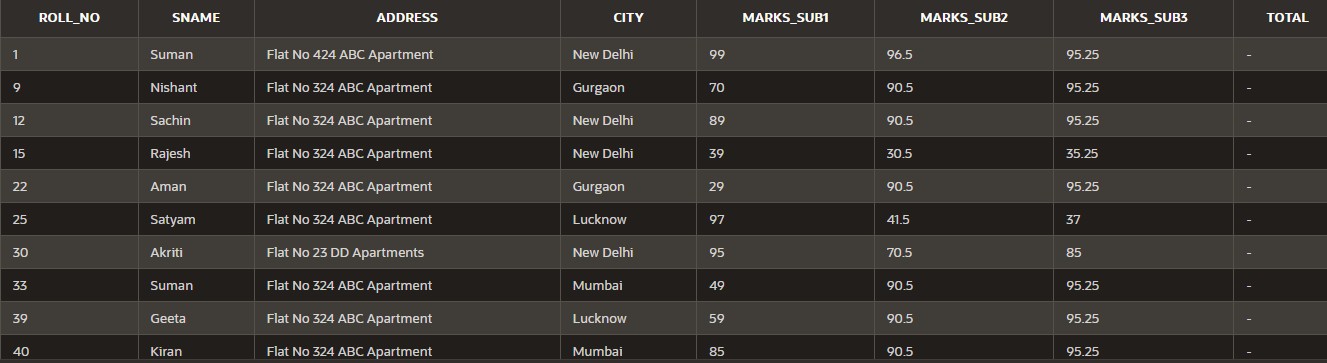
95.25);

INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (30, 'Akriti', 'Flat No 23 DD Apartments', 'New Delhi', 95, 70.50, 85); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (9, 'Nishant', 'Flat No 324 ABC Apartment', 'Gurgaon', 70, 90.50, 95.25); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (15, 'Rajesh', 'Flat No 324 ABC Apartment', 'New Delhi', 39, 30.50,

35.25);

INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (33, 'Suman', 'Flat No 324 ABC Apartment', 'Mumbai', 49, 90.50, 95.25); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (22, 'Aman', 'Flat No 324 ABC Apartment', 'Gurgaon', 29, 90.50, 95.25); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (39, 'Geeta', 'Flat No 324 ABC Apartment', 'Lucknow', 59, 90.50, 95.25); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (25, 'Satyam', 'Flat No 324 ABC Apartment', 'Lucknow', 97, 41.50, 37); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (40, 'Kiran', 'Flat No 324 ABC Apartment', 'Mumbai', 85, 90.50, 95.25); INSERT INTO Student(Roll\_No, SName, Address, City, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3) VALUES (1, 'Suman', 'Flat No 424 ABC Apartment', 'New Delhi', 99, 96.50,

95.25);

***Results-***

#### —----------------------------------

**Q2. Calculate total marks of each student and save it.**

***SQL Query-***

UPDATE Student

SET Total = (Marks\_Sub1 + Marks\_Sub2 + Marks\_Sub3);

***Results-***

#### —----------------------------------

**Q3. Display the name of the student who has got the highest marks.**

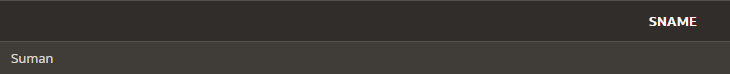
***SQL Query-***

SELECT SName

FROM Student WHERE Total=(

SELECT MAX(Total)

FROM Student);

***Results-***

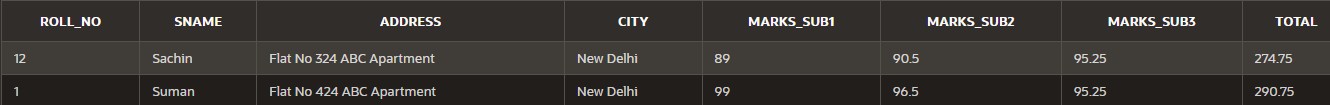
#### —----------------------------------

**Q4. Display the records of those students who have got marks more than the marks of ‘Kiran’.**

***SQL Query-***

SELECT \* FROM Student WHERE Total >(

SELECT Total FROM Student WHERE SName='Kiran');

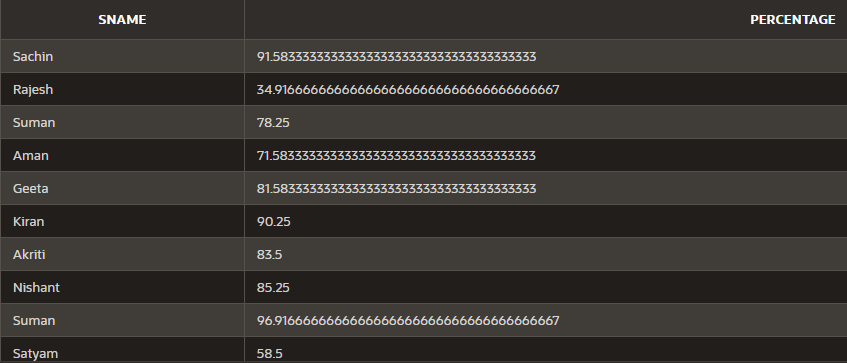
***Results-***

#### —----------------------------------

**Q5. Display the percentage of all the students with their names.**

***SQL Query-***

SELECT SName, (Total/3) AS Percentage FROM Student;

***Results-***

#### —----------------------------------

**Q6. Display the names of those students who have got maximum marks in sub1 and more than 50 in sub2 and sub3.**

***SQL Query-***

SELECT SName

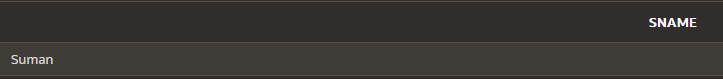
FROM Student WHERE (

Marks\_Sub1 = (

SELECT MAX(Marks\_Sub1) FROM Student WHERE (Marks\_Sub2>50 AND Marks\_Sub3>50)

)

);

***Results-***

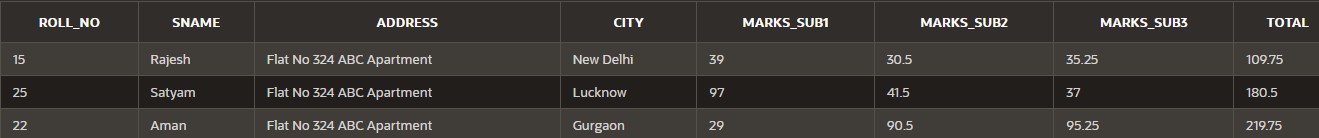
#### —----------------------------------

**Q7. Update the marks of all those students who failed, with grace marks 5. (fail means less than 40).**//Check if failed in any 1 sub or more. Add grace marks in total

***SQL Query-*** UPDATE Student SET Total = (Total+5)

WHERE (Marks\_Sub1<40) OR (Marks\_Sub2<40) OR (Marks\_Sub3<40);

***Results-***



#### —----------------------------------

**Q8. Delete the records of those students who are fail in all three subjects.**

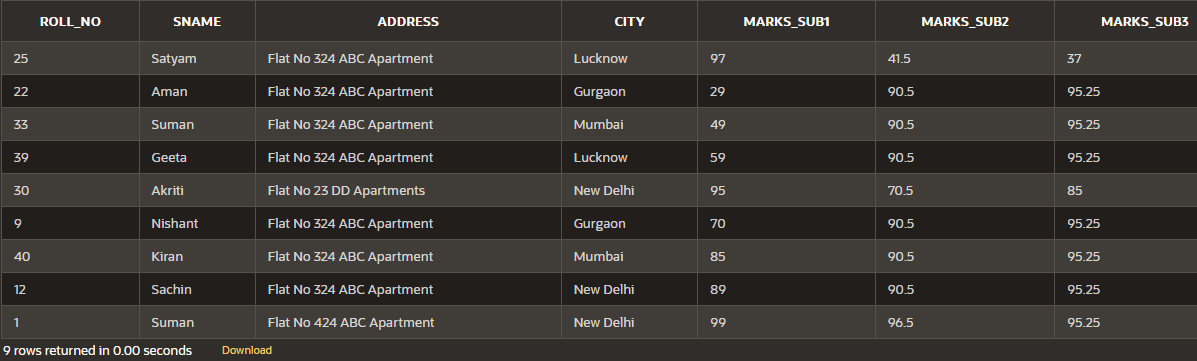
***SQL Query-***

DELETE FROM Student WHERE ((Marks\_Sub1<40)

AND (Marks\_Sub1<40) AND (Marks\_Sub1<40)

);

***Results-***



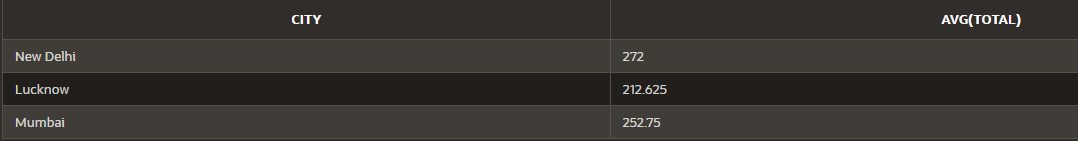
#### —----------------------------------

**Q9. Display the average total marks of all the students city wise excluding the students of ‘Gurgaon’.**

***SQL Query-***

SELECT City, AVG(Total) FROM Student

WHERE City <> 'Gurgaon' GROUP BY City;

***Results-***

#### —----------------------------------

**Q10. Get the names of students with Roll\_no less than 30 and whose total marks is more than the total marks of at least one student with Roll\_no greater than or equal to 30.**

***SQL Query-***

SELECT SName

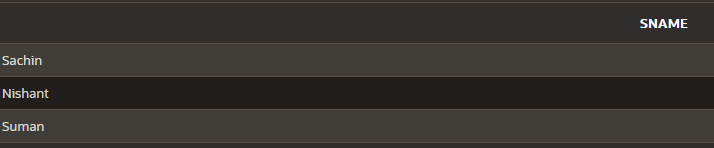
FROM Student WHERE(

(Roll\_No < 30) AND

(Total > (SELECT MIN(Total) FROM Student WHERE Roll\_No >=30)

)

);

***Results-***

#### —----------------------------------

**Q11. Create a composite index on Student table.**

***SQL Query-***

CREATE INDEX StudentI

ON Student (Roll\_No, SName);

***Results-***

#### —----------------------------------

**Q12. Create a view on the table to access roll no, name (in capitals), Marks of all 3 subjects and Percentage of all the students.**

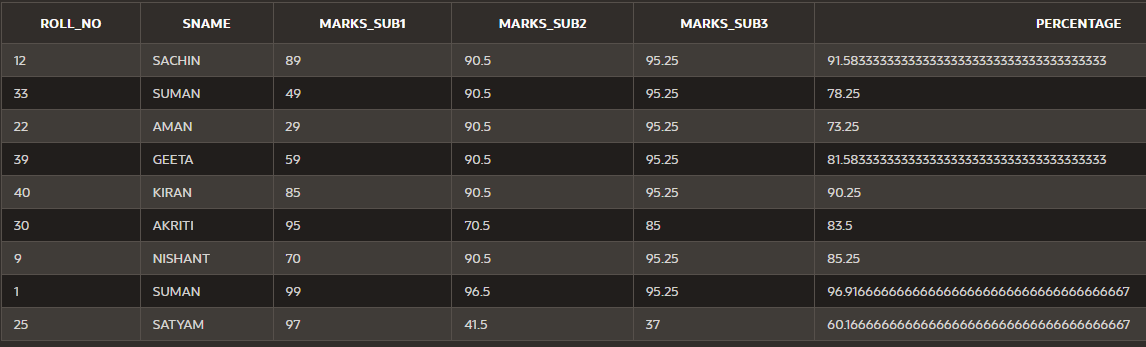
***SQL Query-***

CREATE VIEW PercentageV AS

SELECT Roll\_No, UPPER(SName) AS SName, Marks\_Sub1, Marks\_Sub2, Marks\_Sub3, (Total/3) AS Percentage

FROM Student;

SELECT \* FROM PercentageV;

***Results-***

**—----------------------------------**

# LAB ASSIGNMENT – III (Joins)

#### (Part 1)

**Consider the following three relations:**

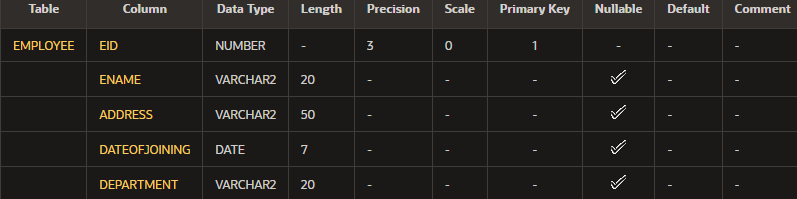
#### Employee (Eid#, EName, Address, DateOfJoining, Department) Project (Pid#, PName, StartDate, TerminationDate)

**AssignedTo (Eid, Pid)**

***Table Creation Query-***

CREATE TABLE Employee(

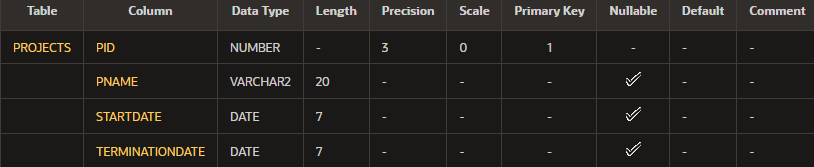
Eid NUMBER(3) PRIMARY KEY, EName VARCHAR2(20),

Address VARCHAR2(50), DateOfJoining DATE, Department VARCHAR2(20));

***Table Creation Query-***

CREATE TABLE Projects(

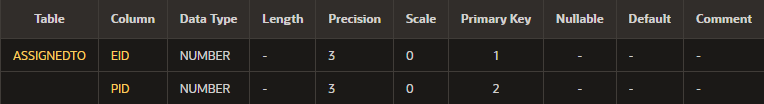
Pid NUMBER(3) PRIMARY KEY, PName VARCHAR2(20),

StartDate DATE, TerminationDate DATE);

***Table Creation Query-***

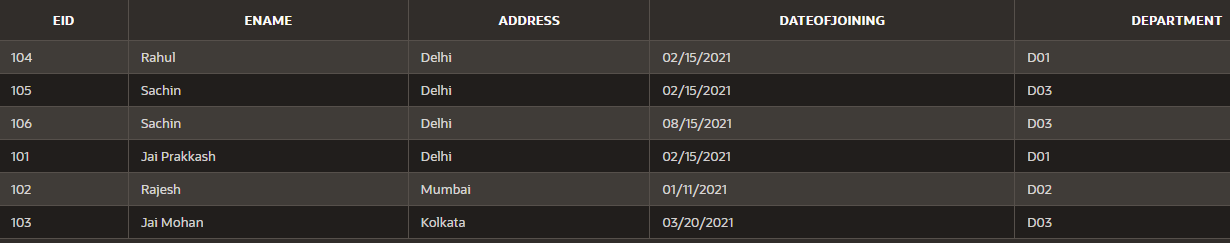
CREATE TABLE AssignedTo(

Eid REFERENCES Employee(Eid),

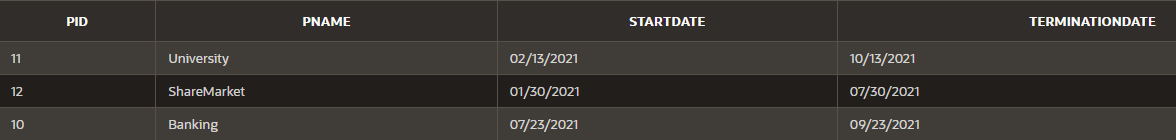
Pid REFERENCES Projects(Pid), PRIMARY KEY(Eid,Pid));

***Insertion in Table Employee-***

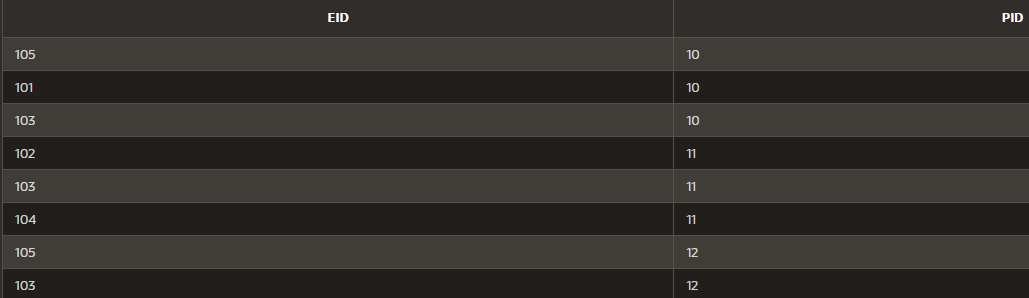
INSERT INTO Employee VALUES(101, 'Jai Prakkash', 'Delhi', '2/15/2021', 'D01'); INSERT INTO Employee VALUES(102, 'Rajesh', 'Mumbai', '1/11/2021', 'D02');

INSERT INTO Employee VALUES(103, 'Jai Mohan', 'Kolkata', '3/20/2021', 'D03'); INSERT INTO Employee VALUES(104, 'Rahul', 'Delhi', '2/15/2021', 'D01'); INSERT INTO Employee VALUES(105, 'Sachin', 'Delhi', '2/15/2021', 'D03'); INSERT INTO Employee VALUES(106, 'Sachin', 'Delhi', '8/15/2021', 'D03');

***Insertion in Table Projects-***

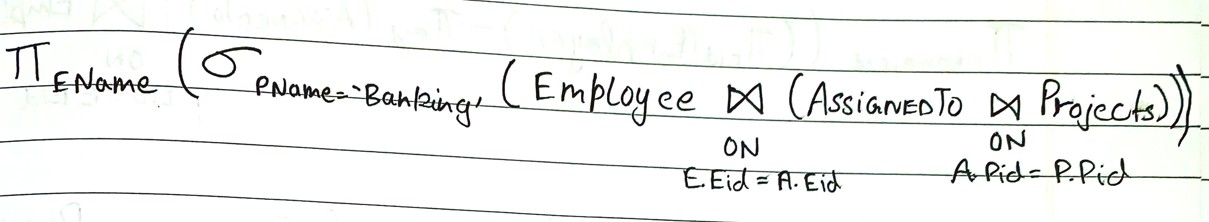
INSERT INTO Projects VALUES(10, 'Banking', '7/23/2021', '9/23/2021'); INSERT INTO Projects VALUES(11, 'University', '2/13/2021', '10/13/2021'); INSERT INTO Projects VALUES(12, 'ShareMarket', '1/30/2021', '7/30/2021');

***Insertion in AssignedTo-***

INSERT INTO AssignedTo VALUES(101, 10); INSERT INTO AssignedTo VALUES(103, 10); INSERT INTO AssignedTo VALUES(105, 10); INSERT INTO AssignedTo VALUES(102, 11); INSERT INTO AssignedTo VALUES(103, 11); INSERT INTO AssignedTo VALUES(104, 11); INSERT INTO AssignedTo VALUES(105, 12); INSERT INTO AssignedTo VALUES(103, 12);

**Now write queries for the following in relational algebra as well as in SQL:**

#### Q1. Find the employees working on ‘Banking’ project.

***Relational Algebra-***

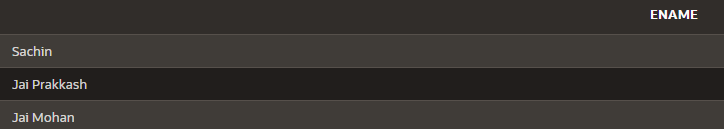
***SQL Query-***

SELECT EName

FROM Employee JOIN AssignedTo

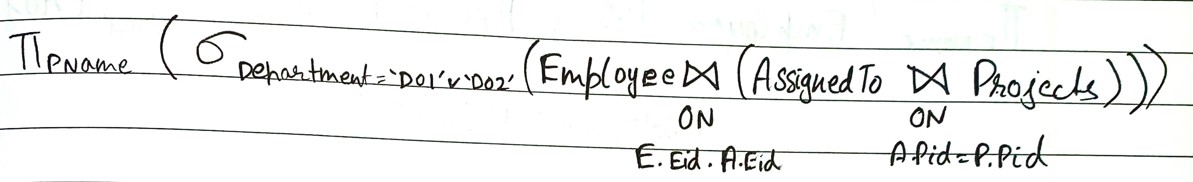
ON AssignedTo.Eid=Employee.Eid JOIN Projects

ON Projects.Pid= AssignedTo.Pid WHERE PName='Banking';

***Results-***

#### —----------------------------------

**Q2. Find the projects assigned to the employees of D01 or D02.**

***Relational Algebra-***

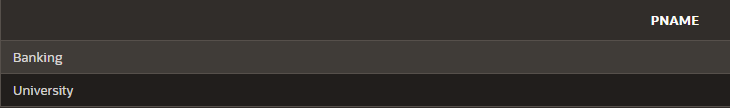
***SQL Query-***

SELECT DISTINCT PName

FROM Projects JOIN AssignedTo

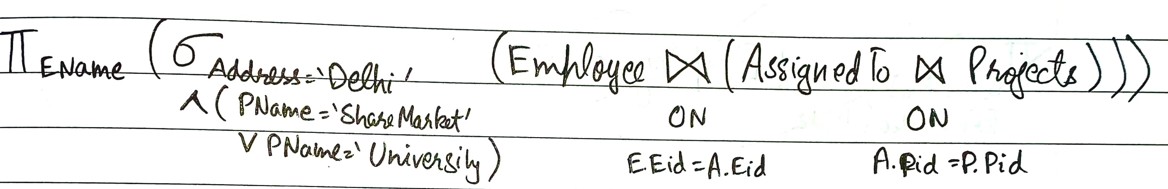
ON AssignedTo.Pid=Projects.Pid JOIN Employee

ON Employee.Eid= AssignedTo.Eid WHERE Department='D01';

***Results-***

#### —----------------------------------

**Q3. Find the employees who belong to Delhi and work on either ‘University’ project or ‘ShareMarket’ project**

***Relational Algebra-***

***SQL Query-***

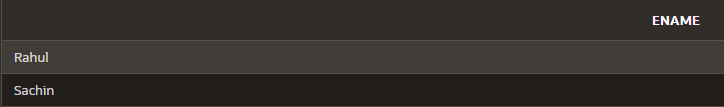
SELECT EName

FROM Employee JOIN AssignedTo

ON AssignedTo.Eid=Employee.Eid JOIN Projects

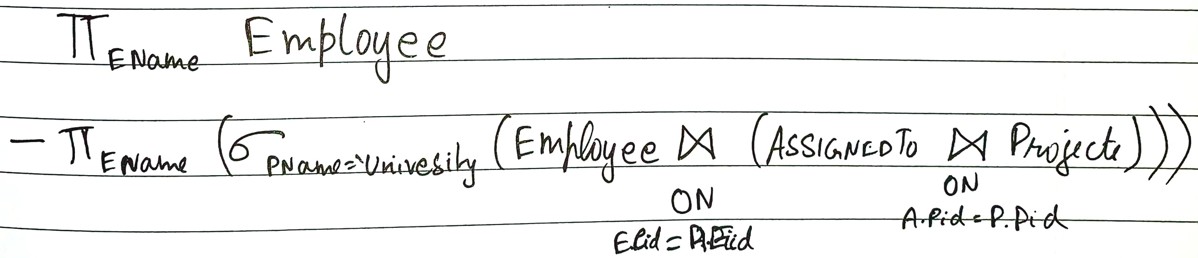
ON Projects.Pid= AssignedTo.Pid

WHERE (PName='ShareMarket' OR PName='University') AND Address='Delhi';

***Results-***

#### —----------------------------------

**Q4. Find the employees who do not work on ‘University’ project.**

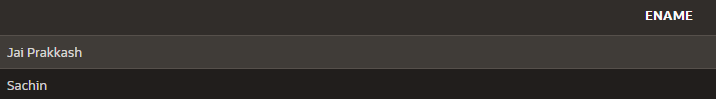
***Relational Algebra-***

***SQL Query-***

SELECT Ename FROM Employee MINUS

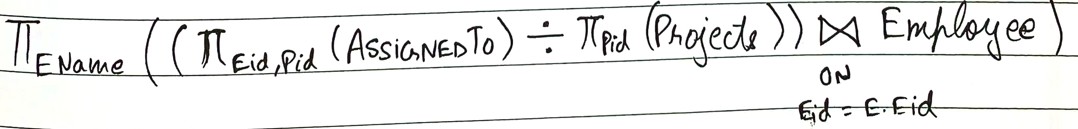
SELECT Employee.EName FROM Employee

JOIN AssignedTo ON Employee.Eid=AssignedTo.Eid JOIN PROJECTS ON AssignedTo.Pid=Projects.Pid WHERE Projects.PName ='University';

***Results-***

#### —----------------------------------

**Q5. Find the employees who work on all projects.**

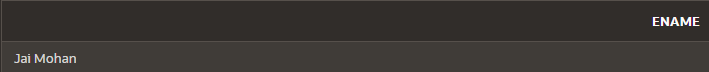
***Relational Algebra-***

***SQL Query-***

SELECT EName FROM Employee WHERE Eid IN(SELECT Eid

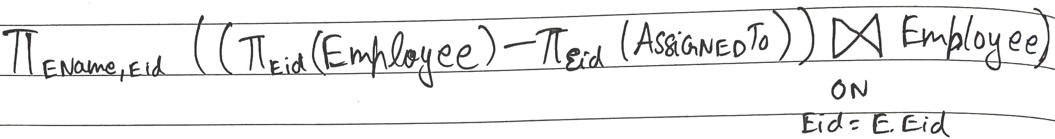
FROM AssignedTo GROUP BY Eid

HAVING COUNT(Pid)= (SELECT COUNT(Pid) FROM Projects));

***Results-***

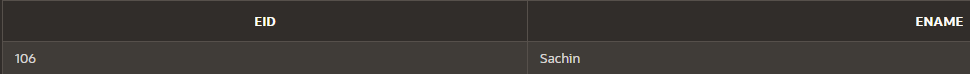
#### —----------------------------------

**Q6. List the employees who have not been assigned any project.**

***Relational Algebra-***

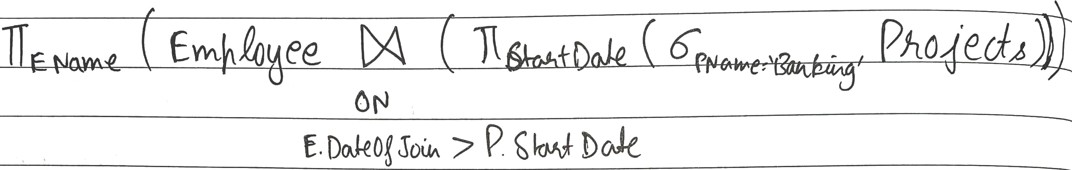
***SQL Query-*** SELECT Eid, EName FROM Employee

WHERE Eid NOT IN (SELECT Eid FROM AssignedTo);

***Results-***

#### —----------------------------------

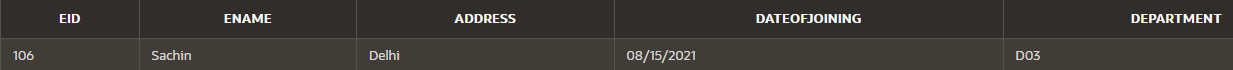
**Q7. Find the employees who joined the department after the commencement of ‘Banking’ project.**

***Relational Algebra-***

***SQL Query-***

SELECT \* FROM Employee

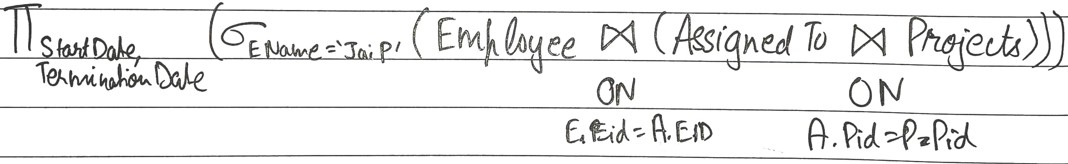
Where Employee.DateOfJoining > (SELECT StartDate from Projects WHERE Pname = 'Banking');

***Results-***

#### —----------------------------------

**Q8. Display the start and termination date of projects which are allotted to ‘Jai Prakkash’.**

***Relational Algebra-***

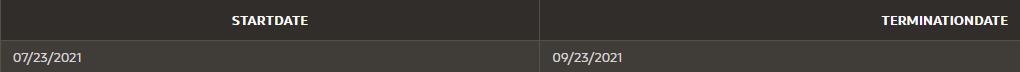


***SQL Query-***

SELECT StartDate, TerminationDate FROM Projects JOIN AssignedTo

ON AssignedTo.Pid = Projects.Pid JOIN Employee

ON AssignedTo.Eid = Employee.Eid WHERE Ename = 'Jai Prakkash';

***Results-***

#### —----------------------------------

**LAB ASSIGNMENT – III (Joins)*(Part 2)***

#### Consider the following relations:

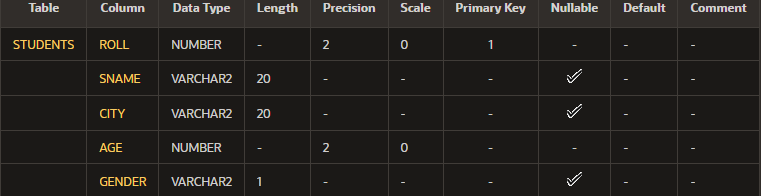
**Students (Roll#, SName, City, Age, Gender) Course (Cid#, CName, Semester, Credits, Fee) Enrollment (Roll#, Cid#, DateOfReg)**

***Table Creation Query-***

CREATE TABLE Students(

Roll NUMBER(2) PRIMARY KEY,

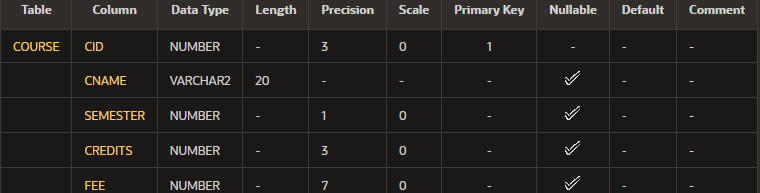
SName VARCHAR2(20), City VARCHAR2(20),

Age NUMBER(2) NOT NULL, Gender VARCHAR2(1));

***Table Creation Query-***

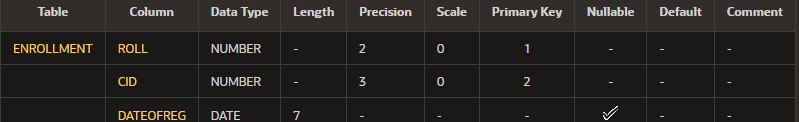
CREATE TABLE Course(

Cid NUMBER(3) PRIMARY KEY,

CName VARCHAR2(20), Semester NUMBER(1), Credits NUMBER(3), Fee NUMBER(7));

***Table Creation Query-***

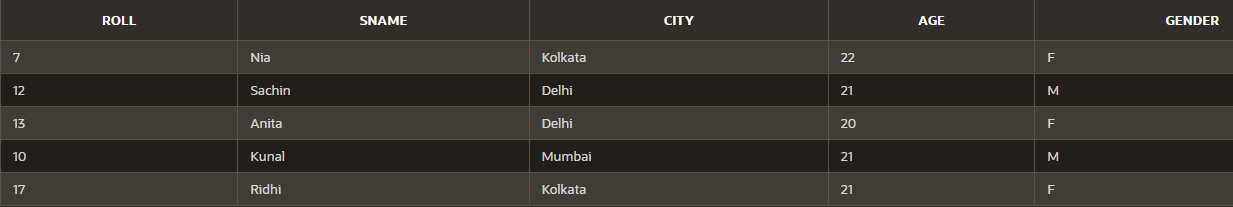
CREATE TABLE Enrollment(

Roll REFERENCES Students(Roll), Cid REFERENCES Course(Cid), PRIMARY KEY(Roll,Cid), DateOfReg DATE);

***Insertion in Table Students-***

INSERT INTO Students VALUES(12, 'Sachin', 'Delhi', 21, 'M'); INSERT INTO Students VALUES(13, 'Anita', 'Delhi', 20, 'F');

INSERT INTO Students VALUES(10, 'Kunal', 'Mumbai', 21, 'M'); INSERT INTO Students VALUES(7, 'Nia', 'Kolkata', 22, 'F');

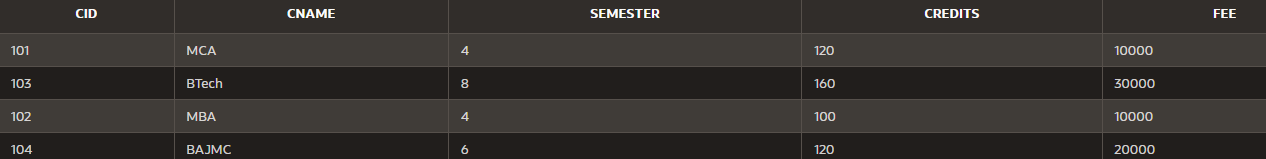
INSERT INTO Students VALUES(17, 'Ridhi', 'Kolkata', 21, 'F');

***Insertion in Table Course-***

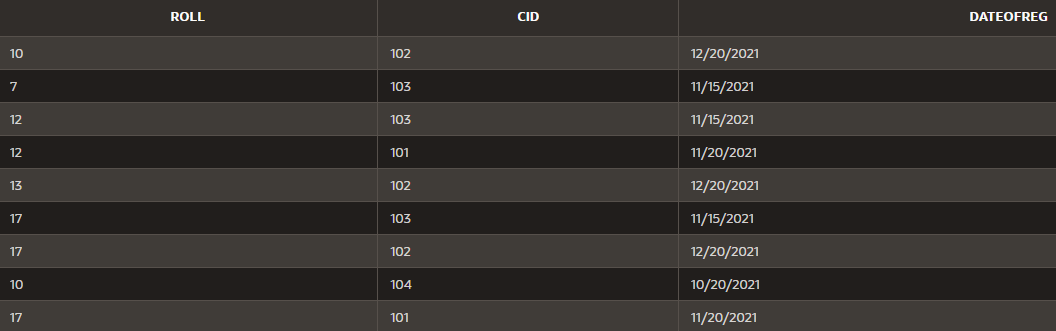
INSERT INTO Course VALUES(101, 'MCA', 4, 120, 10000);

INSERT INTO Course VALUES(102, 'MBA', 4, 100, 10000);

INSERT INTO Course VALUES(103, 'BTech', 8, 160, 30000);

INSERT INTO Course VALUES(104, 'BAJMC', 6, 120, 20000);

***Insertion in Table Enrollment-***

INSERT INTO Enrollment VALUES(12, 101, '11/20/2021'); INSERT INTO Enrollment VALUES(13, 102, '12/20/2021'); INSERT INTO Enrollment VALUES(10, 102, '12/20/2021'); INSERT INTO Enrollment VALUES(7, 103, '11/15/2021'); INSERT INTO Enrollment VALUES(17, 103, '11/15/2021'); INSERT INTO Enrollment VALUES(12, 103, '11/15/2021'); INSERT INTO Enrollment VALUES(10, 104, '10/20/2021'); INSERT INTO Enrollment VALUES(17, 102, '12/20/2021'); INSERT INTO Enrollment VALUES(17, 101, '11/20/2021');

### Now write SQL queries for the following:

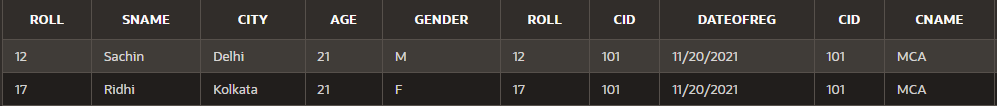
#### Q1.Find the details of students registered in MCA course.

***SQL Query-***

SELECT \* FROM Students JOIN Enrollment

ON Enrollment.Roll= Students.Roll JOIN Course

ON Course.Cid= Enrollment.Cid WHERE CName = 'MCA';

***Results-***

#### —----------------------------------

**Q2.Display the total number of students enrolled in MBA.**

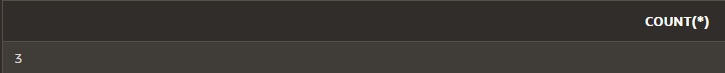
***SQL Query-***

SELECT COUNT(\*) FROM(

SELECT \* FROM Students JOIN Enrollment

ON Enrollment.Roll= Students.Roll JOIN Course

ON Course.Cid= Enrollment.Cid) WHERE CName='MBA';

***Results-***

#### —----------------------------------

**Q3.Display the names of students who are enrolled in the course in which “Anita” is registered.**

***SQL Query-***

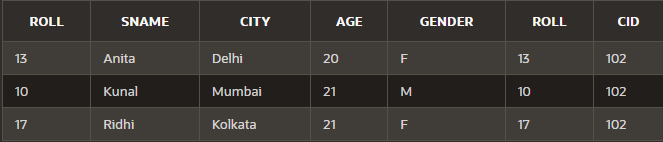
SELECT \* FROM Students JOIN Enrollment

ON Enrollment.Roll= Students.Roll JOIN Course

ON Course.Cid= Enrollment.Cid WHERE CName IN

(SELECT CName FROM Course

JOIN Enrollment ON Enrollment.Cid= Course.Cid JOIN Students ON Students.Roll= Enrollment.Roll WHERE SName='Anita');

***Results-***

#### —----------------------------------

**Q4.Display the names of all the students with their date of enrollment in the courses.**

***SQL Query-***

SELECT Sname,DateOfReg FROM Students

JOIN Enrollment ON Enrollment.Roll= Students.Roll;

***Results-***

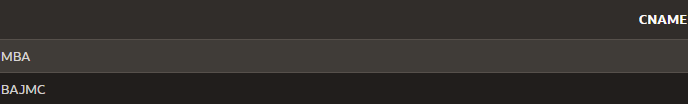
#### —----------------------------------

**Q5.Display the name of the course in which “Kunal” is registered.**

***SQL Query-***

SELECT CName FROM Course

JOIN Enrollment ON Enrollment.Cid= Course.Cid JOIN Students ON Students.Roll= Enrollment.Roll WHERE SName='Kunal';

***Results-***

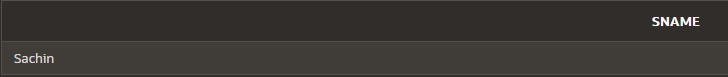
#### —----------------------------------

**Q6.Display the names of boys-students enrolled in MCA.**

***SQL Query-***

SELECT Sname FROM Students

JOIN Enrollment ON Enrollment.Roll= Students.Roll JOIN Course ON Course.Cid= Enrollment.Cid WHERE CName = 'MCA' AND Gender = 'M';

***Results-***

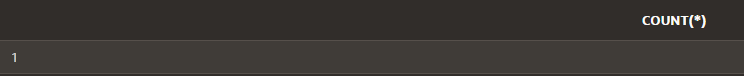
#### —----------------------------------

**Q7.Display the number of students registered in MCA course who live outside Delhi**

***SQL Query-***

SELECT COUNT(\*) FROM( SELECT \* FROM Students

JOIN Enrollment ON Enrollment.Roll= Students.Roll JOIN Course ON Course.Cid= Enrollment.Cid) WHERE CName='MCA' AND City<>'Delhi';

***Results-***

**—----------------------------------**

# LAB ASSIGNMENT – IV

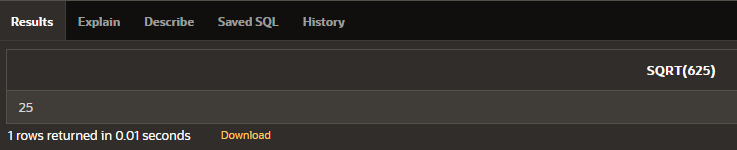
### Write an example of the following functions :

#### SQRT()

This function in SQL is used to return the square root of a specified positive number.

***Sql Query-***

SELECT SQRT(625) FROM DUAL;

***Results-***

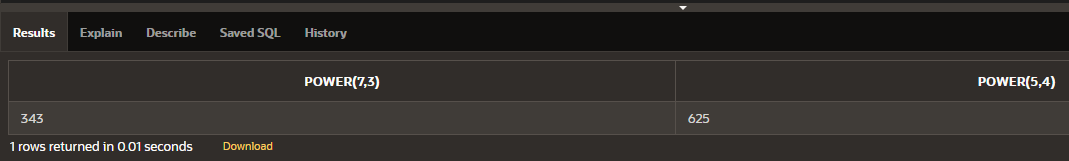
#### —----------------------------------

**POWER()**

The POWER() function returns the value of a number raised to the power of another number.

***Sql Query-***

SELECT POWER(7, 3), POWER(5, 4) FROM DUAL;

***Results-***

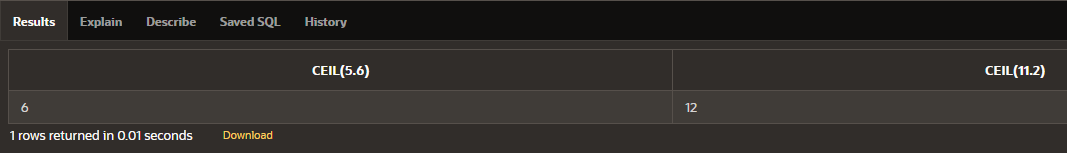
#### —----------------------------------

**CEIL()**

The CEIL() function returns the smallest integer value that is bigger than or equal to a number.

***SQL Query-***

SELECT CEIL(5.6), CEIL(11.2) FROM DUAL;

***Results-***

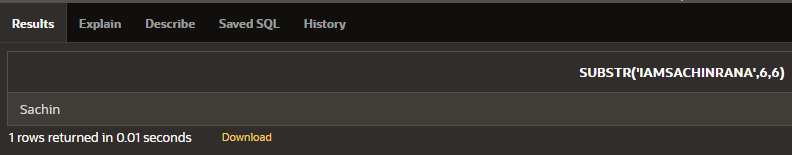
#### —----------------------------------

**SUBSTR()**

The SUBSTR() function extracts a substring from a string (starting at any position).

***SQL Query-***

SELECT SUBSTR('I am Sachin Rana', 6, 6) FROM DUAL;

***Results-***

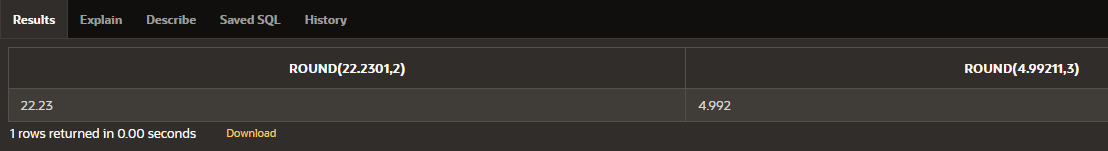
#### —----------------------------------

**ROUND()**

The ROUND() function rounds a number to a specified number of decimal places.

***SQL Query-***

SELECT ROUND(22.2301, 2), ROUND(4.99211, 3) FROM DUAL;

***Results-***

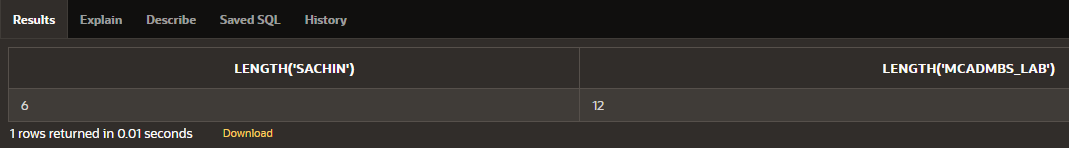
#### —----------------------------------

**LENGTH()**

The LENGTH() function returns the length of a string (in bytes).

***SQL Query-***

SELECT LENGTH('sachin'), LENGTH('MCA DMBS\_LAB') FROM DUAL;

***Results-***

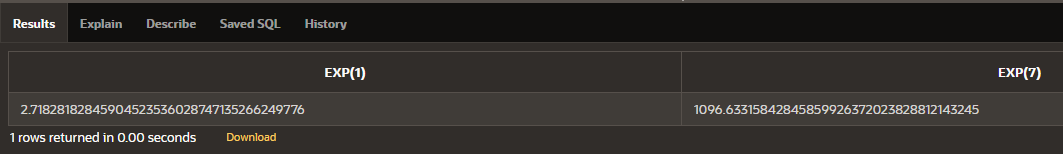
#### —----------------------------------

**EXP()**

The EXP() function returns e raised to the power of the specified number.

***SQL Query-***

SELECT EXP(1), EXP(7) FROM DUAL;

***Results-***

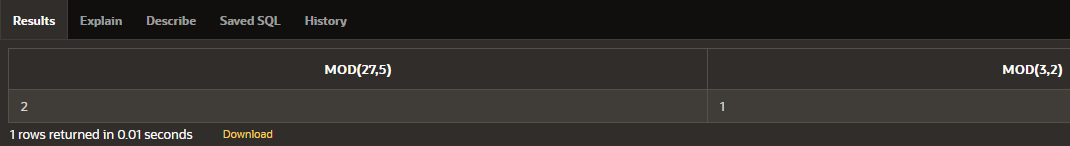
#### —----------------------------------

**MOD()**

The MOD() function returns the remainder of a number divided by another number.

***SQL Query-***

SELECT MOD(27,5), MOD(3,2) FROM DUAL;

***Results-***

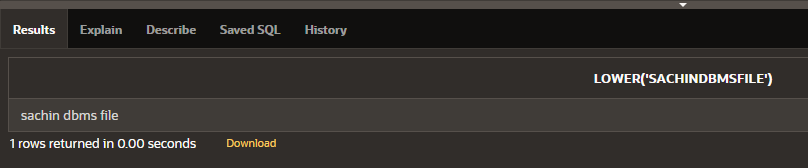
#### —----------------------------------

**LOWER()**

The LOWER() function converts a string to lower-case.

***SQL Query-***

SELECT LOWER('Sachin DBMS FILE') FROM DUAL;

***Results-***

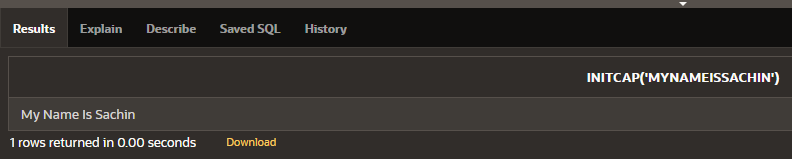
#### —----------------------------------

**INITCAP()**

INITCAP() Function return capitalize string/char (capitalize first letter of each word).

***SQL Query-***

SELECT INITCAP('my name is sachin') FROM DUAL;

***Results-***

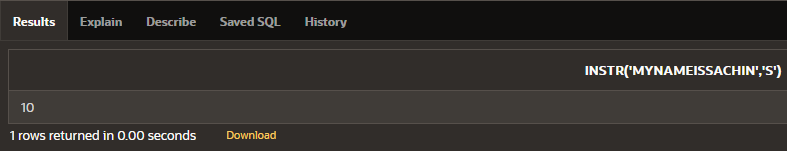
#### —----------------------------------

**INSTR()**

The INSTR() function returns the position of the first occurrence of a string in another string.This function performs a case-insensitive search.

***SQL Query-***

SELECT INSTR('my name is sachin', 's') FROM DUAL;

***Results-***

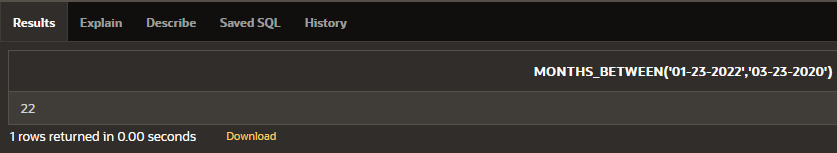
#### —----------------------------------

**MONTHS\_BETWEEN()**

The MONTHS\_BETWEEN() function is used to get the number of months between dates (date1, date2).

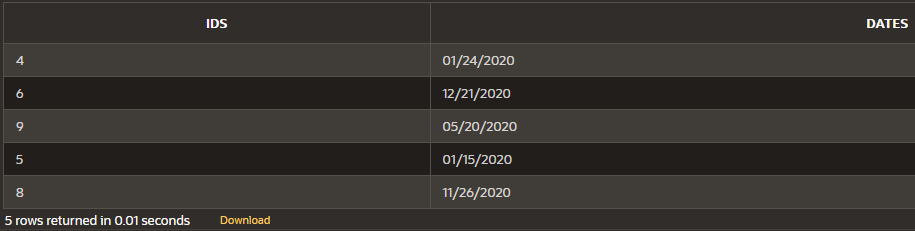
***SQL Query-***

SELECT MONTHS\_BETWEEN ('01-23-2022', '03-23-2020')FROM DUAL;

***Results-***

#### —----------------------------------

**Table Used for Following 6 Functions (max(), min(), avg(), count(), to\_date(), to\_char())**

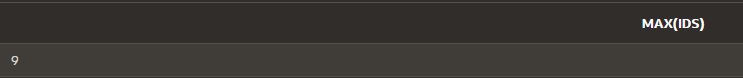
CREATE TABLE Sample4 (Ids NUMBER(2) PRIMARY KEY, Dates DATE);

#### MAX()

The MAX() function returns the maximum value in a set of values.

***SQL Query-***

SELECT MAX(Ids) FROM Sample4

***Results-***

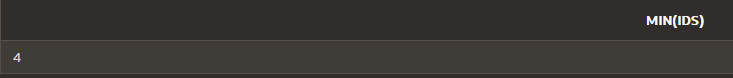
#### —----------------------------------

**MIN()**

The MIN() function returns the minimum value in a set of values.

***SQL Query-***

SELECT MIN(Ids) FROM Sample4

***Results-***

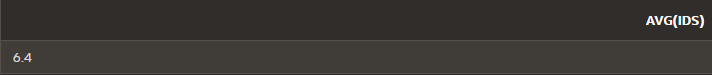
#### —----------------------------------

**AVG()**

The AVG() function returns the average value of a numeric column.

***SQL Query-***

SELECT AVG(Ids) FROM Sample4

***Results-***

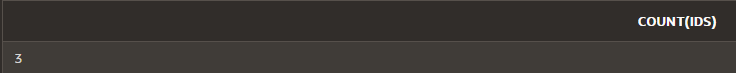
#### —----------------------------------

**COUNT()**

The COUNT() function returns the number of rows that matches a specified criterion.

***SQL Query-***

SELECT COUNT(Ids) FROM Sample4 WHERE Ids>5

***Results-***

#### —----------------------------------

**TO\_DATE()**

The TO\_DATE() function converts date strings in various formats to a date integer value, with data type DATE in the format specified.

***SQL Query-***

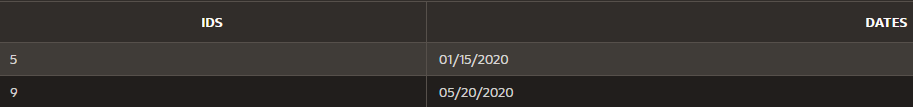
*–converts the given date string into date/month/year format*

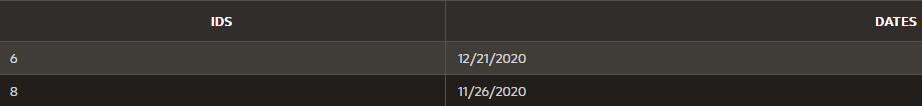
INSERT INTO Sample4 VALUES (5, TO\_DATE('15/01/2020', 'DD/MM/YYYY')); INSERT INTO Sample4 VALUES (9, TO\_DATE('20/05/2020', 'DD/MM/YYYY'));

*–converts the given date string into date/month/year format*

INSERT INTO Sample4 VALUES (8, TO\_DATE('11/26/2020', 'MM/DD/YYYY')); INSERT INTO Sample4 VALUES (6, TO\_DATE('12/21/2020', 'MM/DD/YYYY'));

***Results-***

*–considers 15/01/2020 and 20/05/2020 in date/month/year format*

*–considers 11/26/2020 and 12/21/2020 in month/date/year format*

#### —----------------------------------

**TO\_CHAR()**

TO\_CHAR converts a date of DATE datatype to a value of VARCHAR2 datatype in the format specified by the date format.

***SQL Query-***

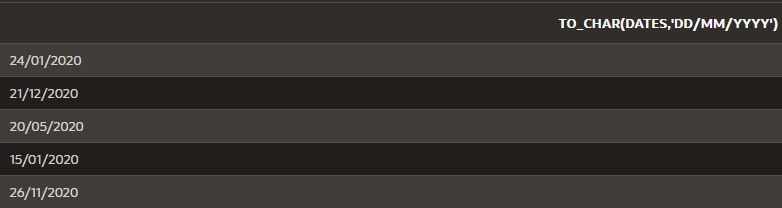
*–to display date in date/month/year format*

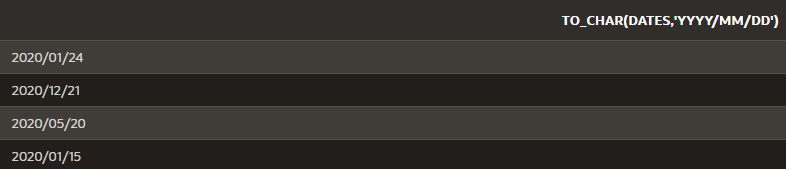
SELECT TO\_CHAR(Dates, 'DD/MM/YYYY') FROM Sample4

*–to display date in month/date/year format*

SELECT TO\_CHAR(Dates, 'YYYY/MM/DD') FROM Sample4

***Results-***

*–displaying dates in date/month/year format*

*–displaying dates in year/month/date format*

**—----------------------------------**

**LAB ASSIGNMENT – V**

### PL/SQL:

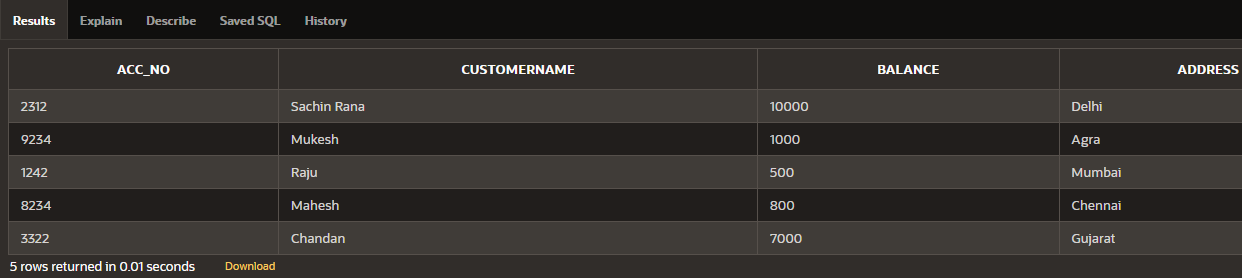
#### Q1. Write a PL/SQL code that accepts the account number from the user and check the users balance. If it is greater than 1000 , only then deduct Rs. 200/- from the balance . The process is fired on the Acct\_Mstr table.

**Solution-**

CREATE TABLE Acct\_Mstr (Acc\_No NUMBER(4) PRIMARY KEY, CustomerName VARCHAR2(20), Balance NUMBER(10), Address VARCHAR2(20));

INSERT INTO Acct\_Mstr VALUES(2312, 'Sachin Rana', 10000, 'Delhi'); INSERT INTO Acct\_Mstr VALUES(9234, 'Mukesh', 1000, 'Agra');

INSERT INTO Acct\_Mstr VALUES(8234, 'Mahesh', 800, 'Chennai'); INSERT INTO Acct\_Mstr VALUES(1242, 'Raju', 500, 'Mumbai');

INSERT INTO Acct\_Mstr VALUES(3322, 'Chandan', 7000, 'Gujarat'); SELECT \* FROM Acct\_Mstr;

***--PL/SQL***

#### DECLARE

V\_Acc\_No Acct\_Mstr.Acc\_No %Type; V\_CustomerName Acct\_Mstr.CustomerName %Type; V\_Balance Acct\_Mstr.Balance %Type;

V\_Address Acct\_Mstr.Address %Type;

#### BEGIN

V\_Acc\_No:=:V\_Acc\_No;

SELECT Acc\_No, CustomerName, Balance, Address INTO V\_Acc\_No, V\_CustomerName, V\_Balance, V\_Address

FROM Acct\_Mstr

WHERE Acc\_No=V\_Acc\_No;

DBMS\_OUTPUT.PUT\_LINE('Before:');

DBMS\_OUTPUT.PUT\_LINE('Acc No= '||V\_Acc\_No||' Customer Name= '||V\_CustomerName||' Balance= '||V\_Balance||' Address= '||V\_Address);

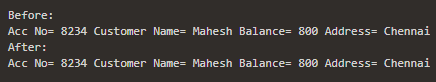
IF(V\_Balance>1000) THEN V\_Balance := V\_Balance - 200; UPDATE Acct\_Mstr

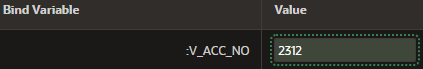
SET Balance = Balance - 200 WHERE Acc\_No = V\_Acc\_No;

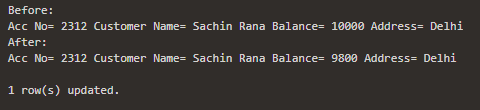
END IF; DBMS\_OUTPUT.PUT\_LINE('After:');

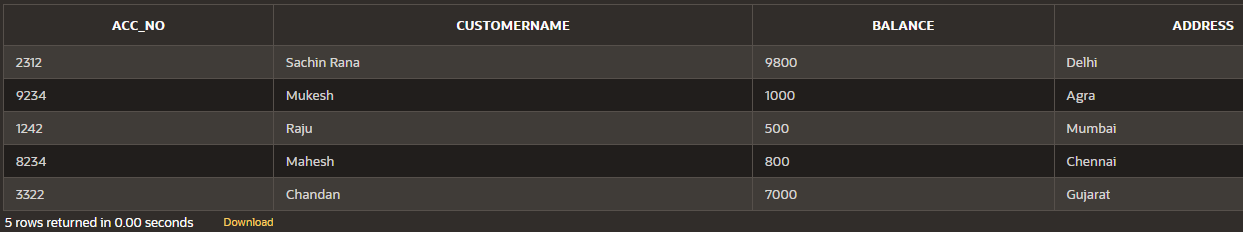
DBMS\_OUTPUT.PUT\_LINE('Acc No= '||V\_Acc\_No||' Customer Name= '||V\_CustomerName||' Balance= '||V\_Balance||' Address= '||V\_Address);

#### END;









**Q2. Write a PL/SQL code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named Areas , consisting of two columns Radius and Area.**

#### Solution-

CREATE TABLE Areas (Radius NUMBER(1) PRIMARY KEY, Area NUMBER(5,2));

***--PL/SQL***

#### DECLARE

V\_Radius NUMBER;

V\_pi constant NUMBER:=3.14; V\_Area NUMBER(5,2);

#### BEGIN

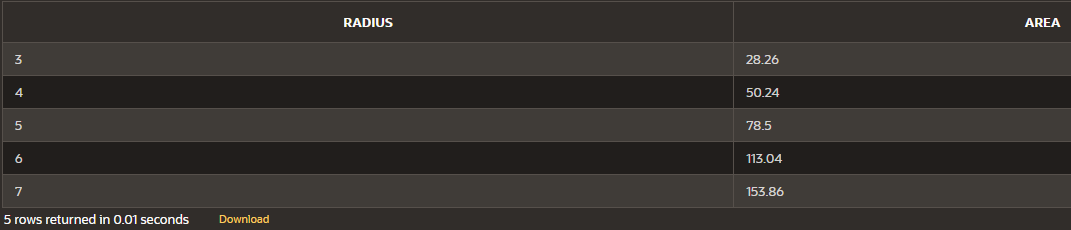
FOR i IN 3..7 LOOP

V\_Radius := i;

V\_Area:=V\_pi \* V\_Radius \* V\_Radius; DBMS\_OUTPUT.PUT\_LINE('Radius= '||V\_Radius||' || Area= '||' '||V\_Area); INSERT INTO Areas VALUES(V\_Radius, V\_Area);

END LOOP;

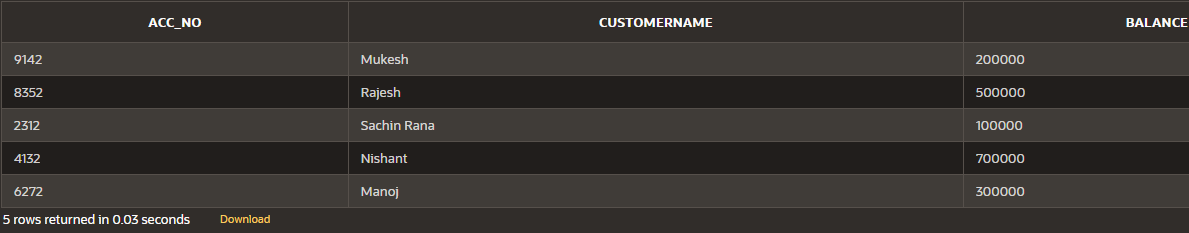
#### END;

SELECT \* FROM Areas;

#### Q3. Write a PL/SQL code that first withdraws an amount of Rs. 1500/-. Then deposits an amount of Rs. 50,000.Update the current balance . Then check to see that the sum of the current balance of all the accounts in the bank does not exceed Rs. 60,00,000. If the balance exceeds Rs 60,00,000 then undo the deposit just made.

**Solution-**

CREATE TABLE Accounts (Acc\_No NUMBER(4) PRIMARY KEY, CustomerName VARCHAR2(20), Balance NUMBER(10));

INSERT INTO Accounts VALUES(2312, 'Sachin Rana', 100000); INSERT INTO Accounts VALUES(9142, 'Mukesh', 200000); INSERT INTO Accounts VALUES(8352, 'Rajesh', 500000); INSERT INTO Accounts VALUES(6272, 'Manoj', 300000); INSERT INTO Accounts VALUES(4132, 'Nishant', 700000);

***--PL/SQL***

#### DECLARE

V\_Sum NUMBER(10) := 0;

#### BEGIN

UPDATE Accounts

SET Accounts.Balance = Accounts.Balance - 1500; UPDATE Accounts

SET Accounts.Balance = Accounts.Balance + 50000; SELECT SUM(Accounts.Balance) INTO V\_Sum FROM Accounts;

IF(V\_Sum > 6000000) THEN

UPDATE Accounts

SET Accounts.Balance = Accounts.Balance - 50000; END IF;

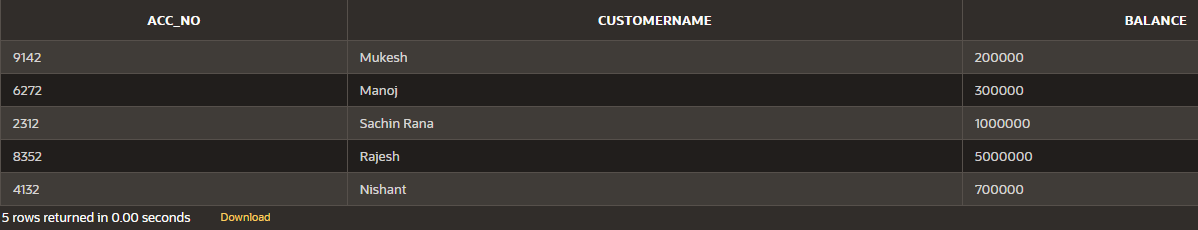
#### END;

***(When Sum exceeds 60,00,000)***

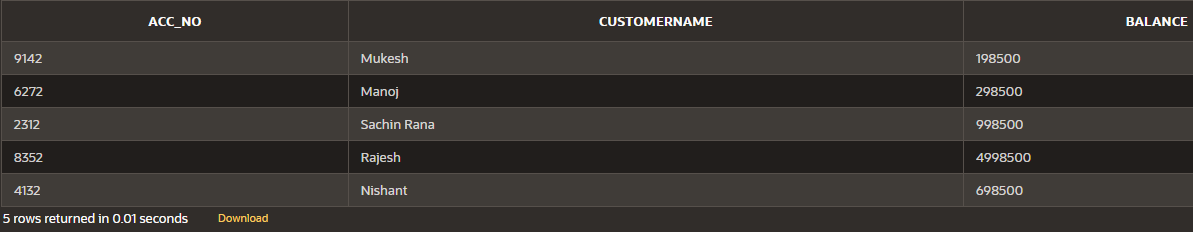
CREATE TABLE Accounts (Acc\_No NUMBER(4) PRIMARY KEY, CustomerName VARCHAR2(20), Balance NUMBER(10));

INSERT INTO Accounts VALUES(2312, 'Sachin Rana', 1000000); INSERT INTO Accounts VALUES(9142, 'Mukesh', 200000); INSERT INTO Accounts VALUES(8352, 'Rajesh', 5000000); INSERT INTO Accounts VALUES(6272, 'Manoj', 300000); INSERT INTO Accounts VALUES(4132, 'Nishant', 700000);

SELECT \* FROM Accounts;



***(After PL/SQL)***



#### Q4. Write a PL/SQL code that will reverse a given number like 12345 to 54321 Solution-

**DECLARE**

V\_Num NUMBER;

V\_Result NUMBER := 0;

#### BEGIN

V\_Num:=:V\_Num;

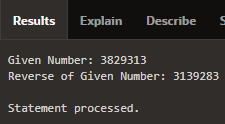
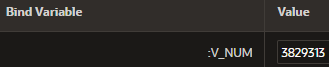
DBMS\_OUTPUT.PUT\_LINE('Given Number: '||V\_Num); WHILE V\_Num>0 LOOP

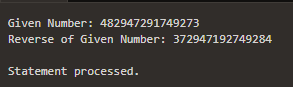
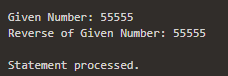
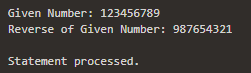
V\_Result := (V\_Result\*10) + MOD(V\_Num,10); V\_Num := FLOOR(V\_Num/10);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Reverse of Given Number: '||V\_Result);

**END;**





## LAB ASSIGNMENT – V

### Use of Cursors :

#### Q1. The bank manager of Delhi branch decides to mark the status of all those accounts as ‘Inactive’ on which there are no transactions performed in the last 6 months.

**Whenever any such update takes place the corresponding record is inserted in another table ‘InactiveAccounts’ with the name and account no of the account holder and his balance.**

#### (Table : Account (AccNo, Name, Address, PANNo, TDate, Status) Solution-

***--Creating Accounts Table***

CREATE TABLE AccountsPL (

AccNo NUMBER(3) PRIMARY KEY, CName VARCHAR2(20),

PANNo VARCHAR2(10),

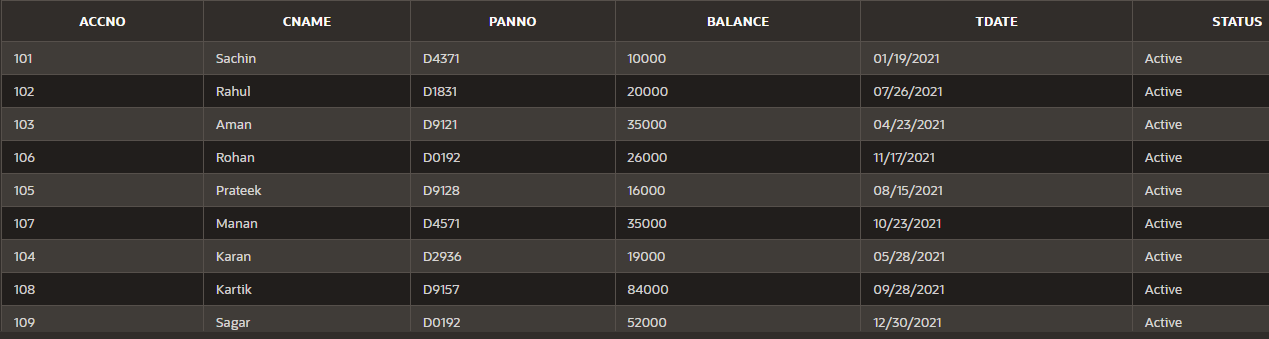
Balance NUMBER(8,2), TDate DATE,

Status VARCHAR2(10));

***--Inserting Values***

INSERT INTO AccountsPL VALUES (101, 'Sachin', 'D4371', 10000.00, '01/19/2021', 'Active'); INSERT INTO AccountsPL VALUES (102, 'Rahul', 'D1831', 20000.00, '07/26/2021', 'Active'); INSERT INTO AccountsPL VALUES (103, 'Aman', 'D9121', 35000.00, '04/23/2021', 'Active'); INSERT INTO AccountsPL VALUES (104, 'Karan', 'D2936', 19000.00, '05/28/2021', 'Active'); INSERT INTO AccountsPL VALUES (105, 'Prateek', 'D9128', 16000.00, '08/15/2021', 'Active'); INSERT INTO AccountsPL VALUES (106, 'Rohan', 'D0192', 26000.00, '11/17/2021', 'Active'); INSERT INTO AccountsPL VALUES (107, 'Manan', 'D4571', 35000.00, '10/23/2021', 'Active'); INSERT INTO AccountsPL VALUES (108, 'Kartik', 'D9157', 84000.00, '09/28/2021', 'Active'); INSERT INTO AccountsPL VALUES (109, 'Sagar', 'D0192', 52000.00, '12/30/2021', 'Active');

***--Displaying Values***

SELECT \* FROM AccountsPL;

***--Creating Inactive Accounts Table***

CREATE TABLE InactiveAccounts (AccNo NUMBER(3) PRIMARY KEY, CName VARCHAR2(20), Balance NUMBER(8,2));

***--Displaying Values***

SELECT \* FROM InactiveAccounts;

***--PL/SQL***

#### DECLARE

V\_AccNo AccountsPL.AccNo%TYPE; V\_CName AccountsPL.CName%TYPE; V\_Balance AccountsPL.Balance%TYPE; CURSOR C\_InAccounts IS

SELECT AccNo, CName, Balance FROM AccountsPL WHERE ((SYSDATE-TDate) > 180);

#### BEGIN

OPEN C\_InAccounts; LOOP

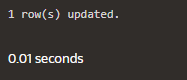
FETCH C\_InAccounts INTO V\_AccNo, V\_CName, V\_Balance; EXIT WHEN C\_InAccounts%notfound;

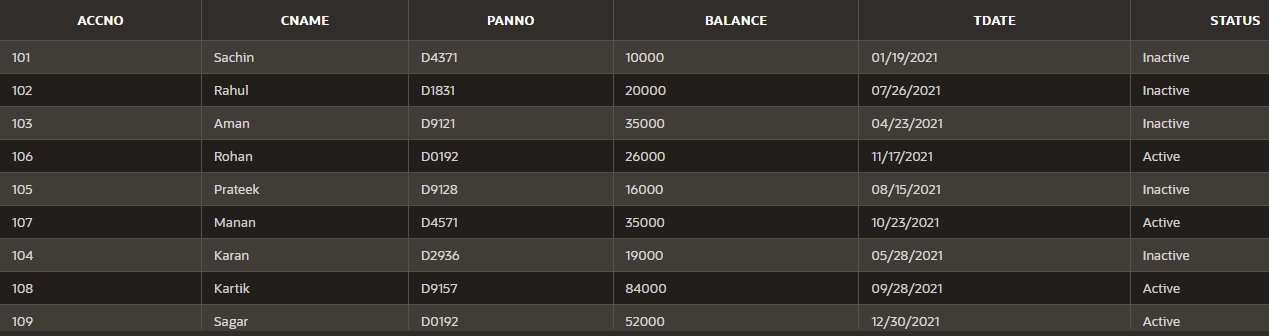
UPDATE AccountsPL SET Status = 'Inactive';

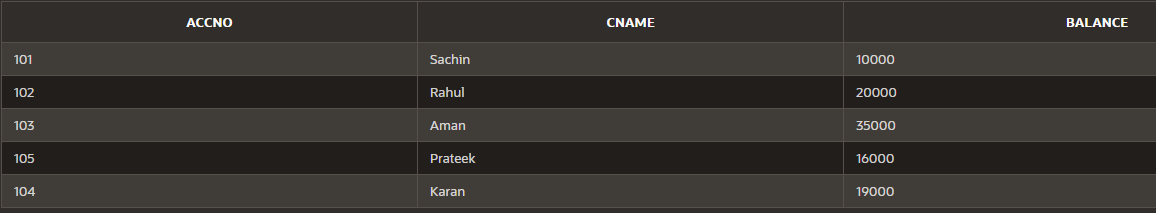
INSERT INTO InactiveAccounts VALUES (V\_AccNo, V\_CName, V\_Balance); END LOOP;

CLOSE C\_InAccounts;

#### END;



***--Displaying AccountsPL Table***

***--Displaying InactiveAccounts***

#### Q2. Write a PL/SQL code that updates the balance of the account holder of ‘delhi’ branch active accounts and displays the number of records updated. (Use implicit cursor).

**Solution-**

***(Using Table From Above Question)***

#### DECLARE

V\_NoOfRecord NUMBER(2);

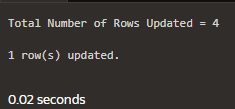
#### BEGIN

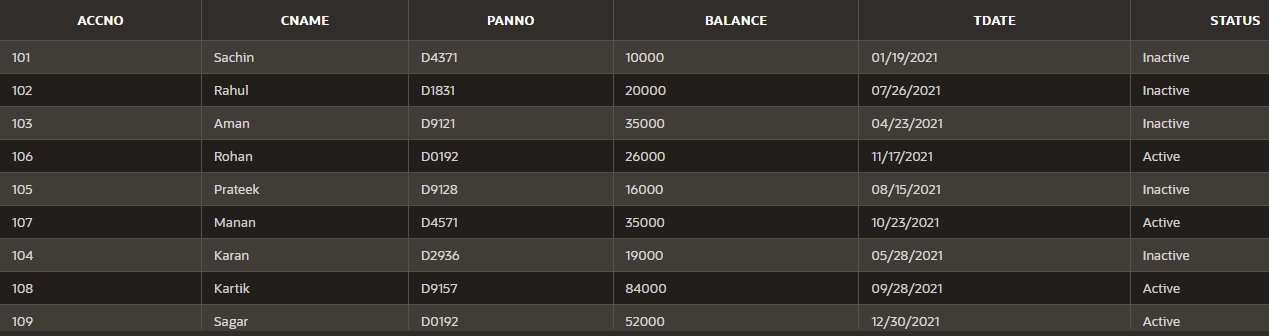
UPDATE AccountsPL

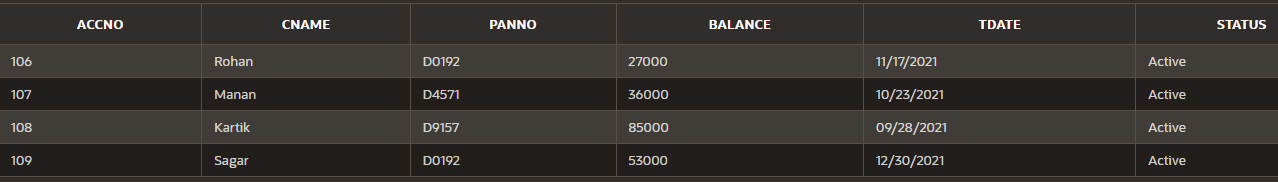
SET Balance = Balance + 1000 WHERE Status = 'Active'; V\_NoOfRecord := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Total Number of Rows Updated = ' || V\_NoOfRecord);

#### END;



***Before-***

***After (Updated 4 Active Records)-***

## LAB ASSIGNMENT – V

### Cursor and Error Handing

#### Q1. Write a Cursor( PL/SQL code) to display the Employee\_name, DateOfBirth , Designation whose basic salary is greater than 15000, if not found then show the proper error message. (Use Exception handling)

**Solution-**

***--Creating EmployeeError Table***

CREATE TABLE EmployeeError (

EID NUMBER(3) PRIMARY KEY, EName VARCHAR2(20), DOB DATE,

Designation VARCHAR2(20), BasicSalary NUMBER(10)

);

***--Inserting Values***

INSERT INTO EmployeeError VALUES(1, 'Sachin', '09/12/2000', 'Junior Developer', 10000); INSERT INTO EmployeeError VALUES(2, 'Aman', '04/19/1990', 'Senior Developer', 12000); INSERT INTO EmployeeError VALUES(3, 'Rahul', '03/26/1995', 'Junior Developer', 10000); INSERT INTO EmployeeError VALUES(4, 'Sachin', '11/22/1999', 'Manager', 13000);

INSERT INTO EmployeeError VALUES(5, 'Sachin', '07/17/2000', 'Associate Developer', 11000);

***--Displaying Values***

***--PL/SQL***

#### DECLARE

V\_EName EmployeeError.EName%type; V\_DOB EmployeeError.DOB%type;

V\_Designation EmployeeError.Designation%type; V\_BasicSalary EmployeeError.BasicSalary%type; temp NUMBER:= 0;

CURSOR C\_Employee IS

SELECT EName, DOB, Designation, BasicSalary FROM EmployeeError;

#### BEGIN

OPEN C\_Employee; LOOP

FETCH C\_Employee INTO V\_EName, V\_DOB, V\_Designation, V\_BasicSalary; EXIT WHEN C\_Employee%NOTFOUND;

IF V\_BasicSalary>15000 THEN

DBMS\_OUTPUT.PUT\_LINE(V\_EName || ' ' || V\_DOB || ' '|| V\_Designation || ' ' || V\_BasicSalary || ' > 15000');

temp:= 1; END IF;

END LOOP;

IF temp<=0 THEN

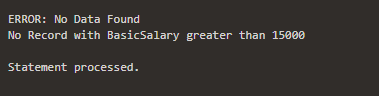
RAISE NO\_DATA\_FOUND; END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('ERROR: No Data Found');

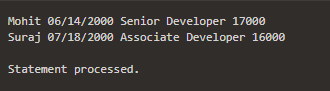
DBMS\_OUTPUT.PUT\_LINE('No Record with BasicSalary greater than 15000'); CLOSE C\_Employee;

#### END;

***--Output when no such record***

***--Inserting records with salary greater than 15000***

INSERT INTO EmployeeError VALUES(6, 'Suraj', '07/18/2000', 'Associate Developer', 16000); INSERT INTO EmployeeError VALUES(7, 'Mohit', '06/14/2000', 'Senior Developer', 17000);

***--Output when there are such records***

## LAB ASSIGNMENT – V

### Procedures and Functions

#### Q1. Create a procedure that accepts the emp id from the calling procedure and displays his/her record.

**Solution-**

#### (Using following table for this question)

***--PL/SQL***

**CREATE OR REPLACE PROCEDURE** EmpID\_Q1 ( V\_SearchID IN EmployeePF.EID%type) **IS**

V\_EID EmployeePF.EID%type; V\_EName EmployeePF.EName%type; V\_EAge EmployeePF.EAge%type;

CURSOR C\_EmployeePF IS SELECT EID, EName, EAge FROM EmployeePF;

#### BEGIN

OPEN C\_EmployeePF; LOOP

FETCH C\_EmployeePF INTO V\_EID, V\_EName, V\_EAge; EXIT WHEN C\_EmployeePF%NOTFOUND;

IF V\_EID = V\_SearchID THEN DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || V\_EID);

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || V\_EName); DBMS\_OUTPUT.PUT\_LINE('Employee Age: ' || V\_EAge); END IF;

END LOOP;

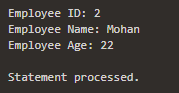
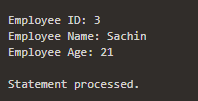
CLOSE C\_EmployeePF;

#### END;

***--Calling Procedure with EID as parameter***

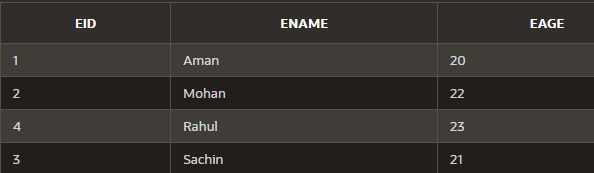
BEGIN

EmpID\_Q1(3); ***--EmpID\_Q1(2);***

END;

#### Q2. Create a function that accepts emp id from the calling procedure and returns his age.

**Solution-**

***--(Using Table)***

***--PL/SQL***

**CREATE OR REPLACE FUNCTION** EmpID\_Q2 ( V\_SearchID IN EmployeePF.EID%type) RETURN EmployeePF.EAge%type

IS

V\_ReturnAge EmployeePF.EAge%type;

#### BEGIN

SELECT EAge INTO V\_ReturnAge FROM EmployeePF WHERE EID = V\_SearchID; RETURN V\_ReturnAge;

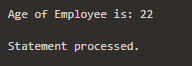
#### END;

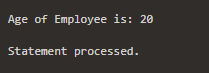
***--Calling Function with EID as parameter and storing returned value***

DECLARE

V\_GetAge EmployeePF.EAge%type; BEGIN

V\_GetAge := EmpID\_Q2(2);

DBMS\_OUTPUT.PUT\_LINE('Age of Employee is: ' || V\_GetAge); END;

***--with EID = 1***

#### Q3. Write a PL/SQL function ODDEVEN to return value TRUE if the number passed to it is EVEN else will return FALSE.

**Solution-**

***--PL/SQL***

**CREATE OR REPLACE FUNCTION** ODDEVEN ( V\_Num IN NUMBER) RETURN BOOLEAN

IS

V\_Even BOOLEAN;

#### BEGIN

IF MOD(V\_Num,2) = 0 THEN RETURN TRUE;

ELSE

RETURN FALSE; END IF;

#### END;

***--Calling Function with a odd number as parameter***

#### DECLARE

V\_IsEven BOOLEAN;

#### BEGIN

V\_IsEven := ODDEVEN(87);

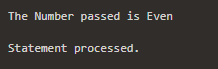
IF V\_IsEven THEN

DBMS\_OUTPUT.PUT\_LINE('The Number passed is Even'); ELSE

DBMS\_OUTPUT.PUT\_LINE('The Number passed is Odd'); END IF;

#### END;

***--Calling Function with an even number as parameter***

***--V\_IsEven := ODDEVEN(56);***

#### Q4. Write a PL/SQL procedure called MULTI\_TABLE that takes two numbers as parameters and displays the multiplication of the first parameter till the second parameter.

**Solution-**

**CREATE OR REPLACE PROCEDURE** MULTI\_TABLE ( V\_Num1 IN NUMBER, V\_Num2 IN NUMBER)

IS

#### BEGIN

DBMS\_OUTPUT.PUT\_LINE('Multiplication of first parameter till second parameter:-'); FOR i IN 1..V\_Num2

LOOP

DBMS\_OUTPUT.PUT\_LINE( V\_Num1 ||' x '|| i ||' = '|| V\_Num1\*i ); END LOOP;

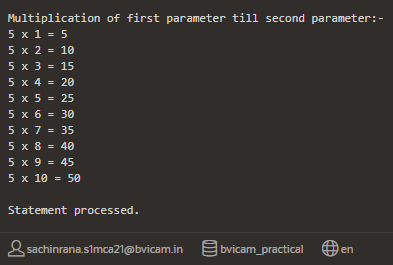
#### END;

***--Calling the procedure MULTI\_TABLE***

#### BEGIN

MULTI\_TABLE(5, 10);

**END;**



## LAB ASSIGNMENT – V

### Trigger

#### Q1. Create a Trigger in PL/SQL which will work before deletion or updation in employee\_master table and inserts a duplicate copy of the record in another table for future reference.

**Solution-**

***--Employee\_Master Table***

***--Employee\_Master\_Backup***



***--Creating Trigger***

**CREATE OR REPLACE TRIGGER** Before\_Delete\_Update BEFORE DELETE OR UPDATE ON Employee\_Master FOR EACH ROW

#### BEGIN

INSERT INTO Employee\_Master\_Backup VALUES (:old.EmpID, :old.EmpName,

:old.EmpSalary);

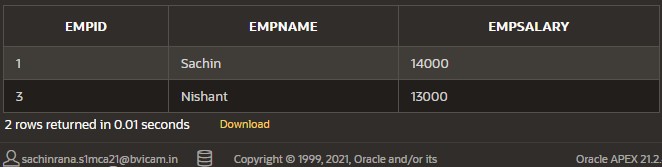
#### END;

***--Deleting a record from Employee\_Master Table***

DELETE FROM Employee\_Master WHERE EmpID = 3

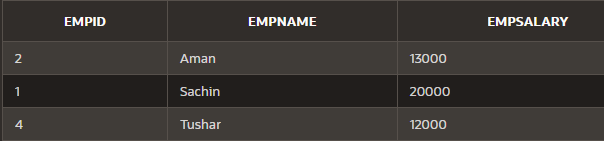
***--Updating a record from Employee\_Master Table***

UPDATE Employee\_Master SET EmpSalary = 20000 WHERE EmpID = 1

***--Employee\_Master\_Backup Table***

#### Q2. Create a Trigger in PL/SQL which will work after update or delete on employee\_master and store the old data and deleted data in a table named as backup\_employee\_master .

**Solution-**

***--Employee\_Master Table***

***--Backup\_Employee\_Master Table***



***--Creating Trigger***

**CREATE OR REPLACE TRIGGER** After\_Delete\_Update AFTER DELETE OR UPDATE ON Employee\_Master FOR EACH ROW

#### BEGIN

INSERT INTO Employee\_Master\_Backup VALUES (:old.EmpID, :old.EmpName,

:old.EmpSalary);

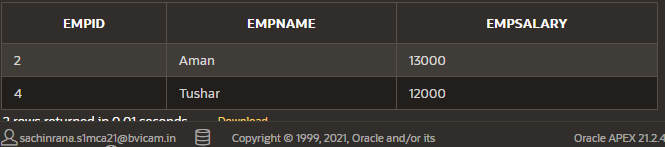
#### END;

***--Deleting a record from Employee\_Master Table***

DELETE FROM Employee\_Master WHERE EmpID= 4;

***--Updating a record from Employee\_Master Table***

UPDATE Employee\_Master SET EmpName = 'Amann' WHERE EmpID = 2;

***--Backup\_Employee\_Master Table***