Group A

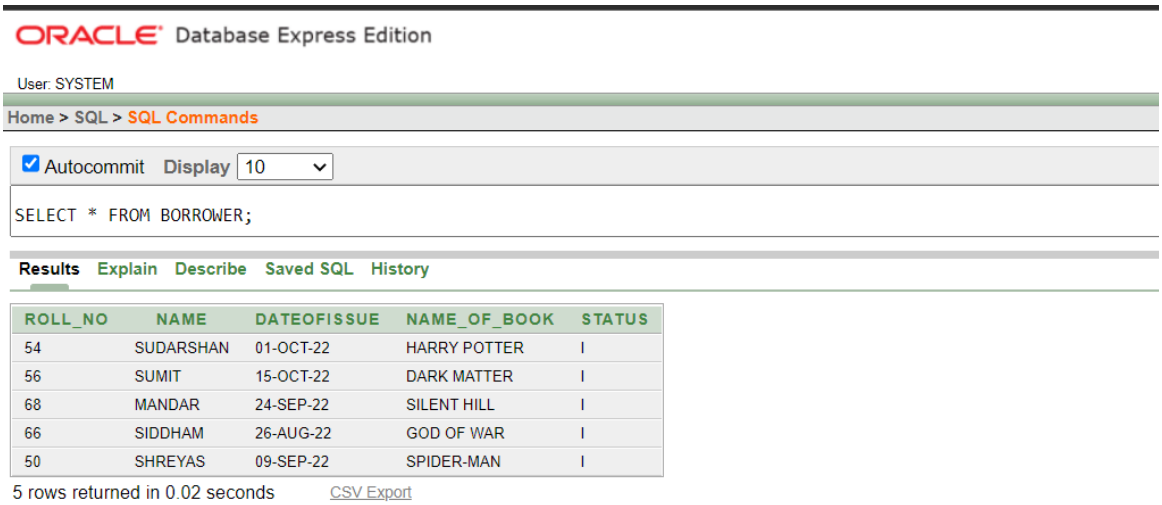
Assignment Number 4

PL/SQL

Problem Statement:
Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.
Schema :
1. Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)
2. Fine(Roll_no,Date,Amt) Accept roll_no & name of book from user.
Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will beRs 5per day.
If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
After submitting the book, status will change from I to R.
If condition of fine is true, then details will be stored into fine table.

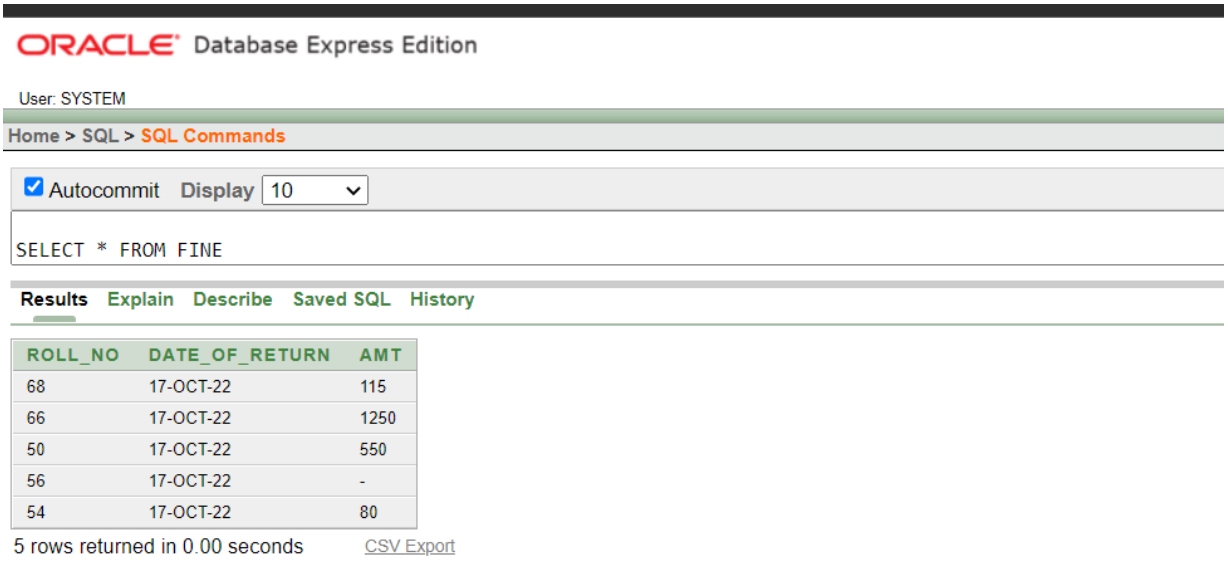
PROGRAM INPUT

```
CREATE TABLE  BORROWER
(
roll_no NUMBER,
name VARCHAR2(25),
dateofissue DATE,
name_of_book VARCHAR2(25),
status VARCHAR2(20)
);

CREATE TABLE FINE
(
roll_no NUMBER,
date_of_return DATE,
amt NUMBER
);

INSERT INTO borrower VALUES(54,'SUDARSHAN',TO_DATE('01-10-2022','DD-MM-
YYYY'),'HARRY POTTER','I');

INSERT INTO borrower VALUES(56,'SUMIT',TO_DATE('15-10-2022','DD-MM-
YYYY'),'DARK MATTER','I');

INSERT INTO borrower VALUES(68,'MANDAR',TO_DATE('24-09-2022','DD-MM-
YYYY'),'SILENT HILL','I');

INSERT INTO borrower VALUES(66,'SIDDHAM',TO_DATE('26-08-2022','DD-MM-
YYYY'),'GOD OF WAR','I');

INSERT INTO borrower VALUES(50,'SHREYAS',TO_DATE('09-09-2022','DD-MM-
YYYY'),'SPIDER-MAN','I');
```




```
DECLARE
        i_roll_no NUMBER := 54;
        name_of_book VARCHAR2(25);
        no_of_days NUMBER;
        return_date DATE := TO_DATE(SYSDATE,'DD-MM-YYYY');
        temp NUMBER;
        doi DATE;
        fine NUMBER;

BEGIN

        i_roll_no := i_roll_no;
        name_of_book := '&nameofbook';
        --dbms_output.put_line(return_date);
        SELECT to_date(borrower.dateofissue,'DD-MM-YYYY') INTO doi FROM borrower
WHERE borrower.roll_no = i_roll_no AND borrower.name_of_book = name_of_book;
        no_of_days := return_date-doi;
        dbms_output.put_line(no_of_days);
        IF (no_of_days >15 AND no_of_days <=30) THEN
                fine := 5*no_of_days;

        ELSIF (no_of_days>30 ) THEN
                temp := no_of_days-30;
                fine := 150 + temp*50;
        END IF;
        dbms_output.put_line(fine);
        INSERT INTO fine VALUES(i_roll_no,return_date,fine);
        UPDATE borrower SET status = 'R' WHERE borrower.roll_no = i_roll_no;

END;
/
```

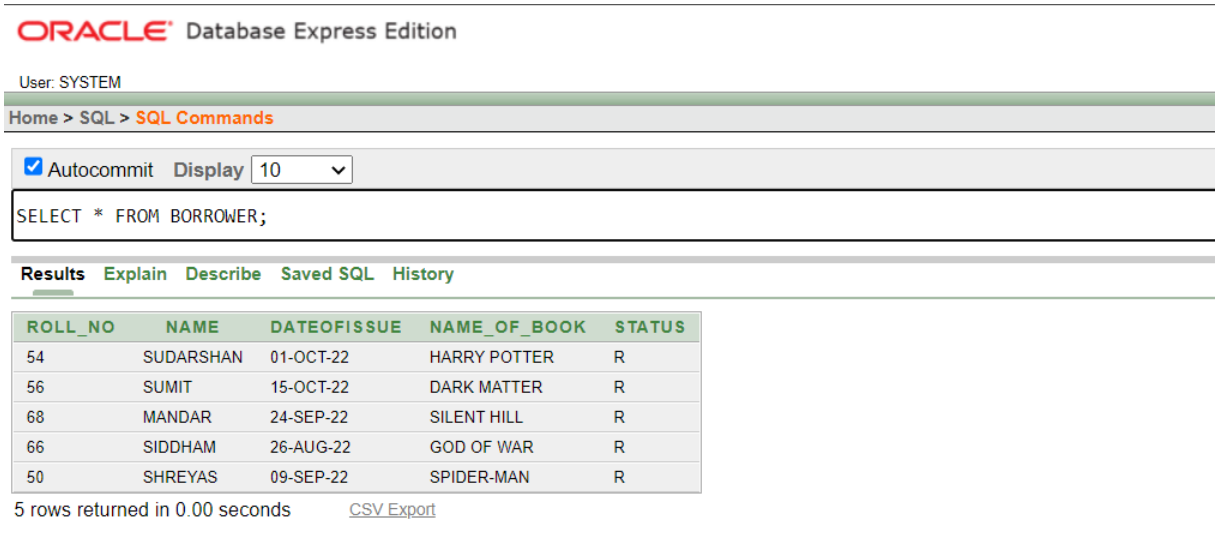
FINE TABLE AFTER SUBMITTING :



ROLL_NO	DATE_OF_RETURN	AMT
68	17-OCT-22	115
66	17-OCT-22	1250
50	17-OCT-22	550
56	17-OCT-22	-
54	17-OCT-22	80

5 rows returned in 0.00 seconds CSV Export

BORROWER TABLE AFTER SUBMITTING :



ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
54	SUDARSHAN	01-OCT-22	HARRY POTTER	R
56	SUMIT	15-OCT-22	DARK MATTER	R
68	MANDAR	24-SEP-22	SILENT HILL	R
66	SIDDHAM	26-AUG-22	GOD OF WAR	R
50	SHREYAS	09-SEP-22	SPIDER-MAN	R

5 rows returned in 0.00 seconds CSV Export

Group A

Assignment Number 5

PL/SQL

Problem Statement:

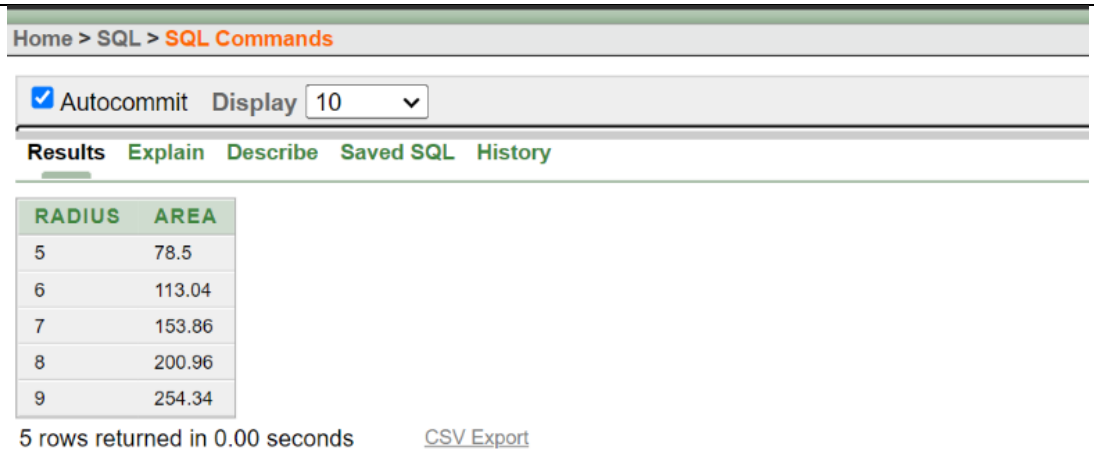
1. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

Lab Exercise:

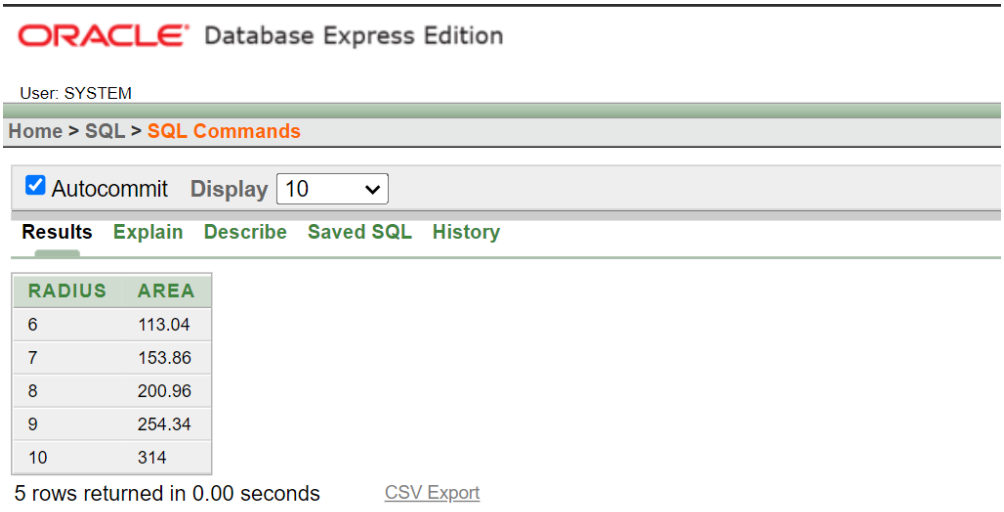
1. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 6 to 10. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.
2. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 10 to 15. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.
3. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 4 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

PROGRAM INPUT & OUTPUT

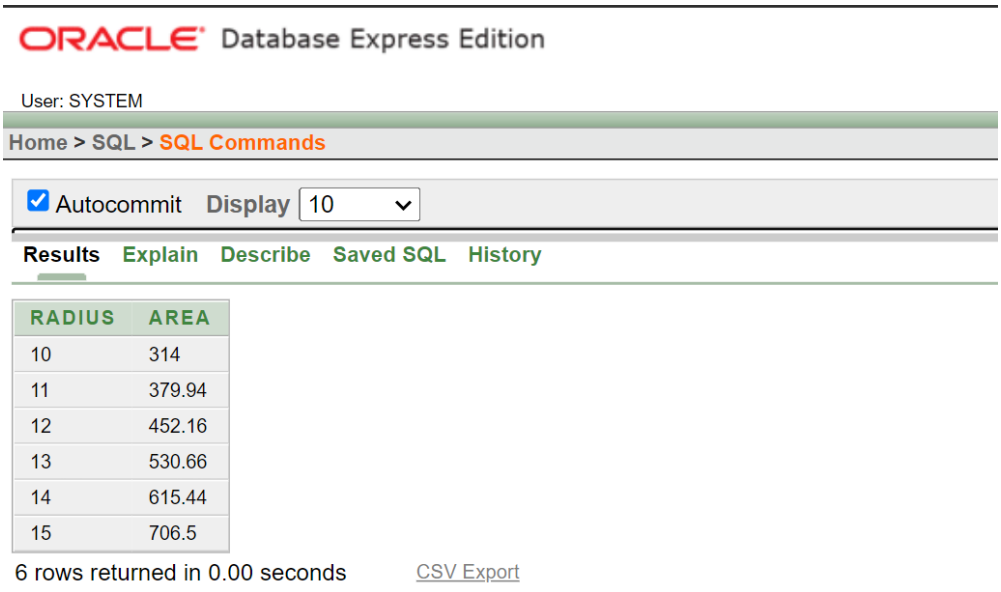
```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2));

DECLARE
pi constant number(4,2):=3.14;
radius number(5);
area number(14,2);

BEGIN
radius :=5;
while radius <=9
loop
area := pi*power(radius,2);
insert into areas values(radius,area);
radius := radius+1;
end loop;
end;
/
SELECT * FROM AREAS;
```




```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2));

DECLARE
pi constant number(4,2):=3.14;
radius number(5);
area number(14,2);

BEGIN
radius :=6;
while radius <=10
loop
area := pi*power(radius,2);
insert into areas values(radius,area);
radius := radius+1;
end loop;
end;
/
SELECT * FROM AREAS;
```



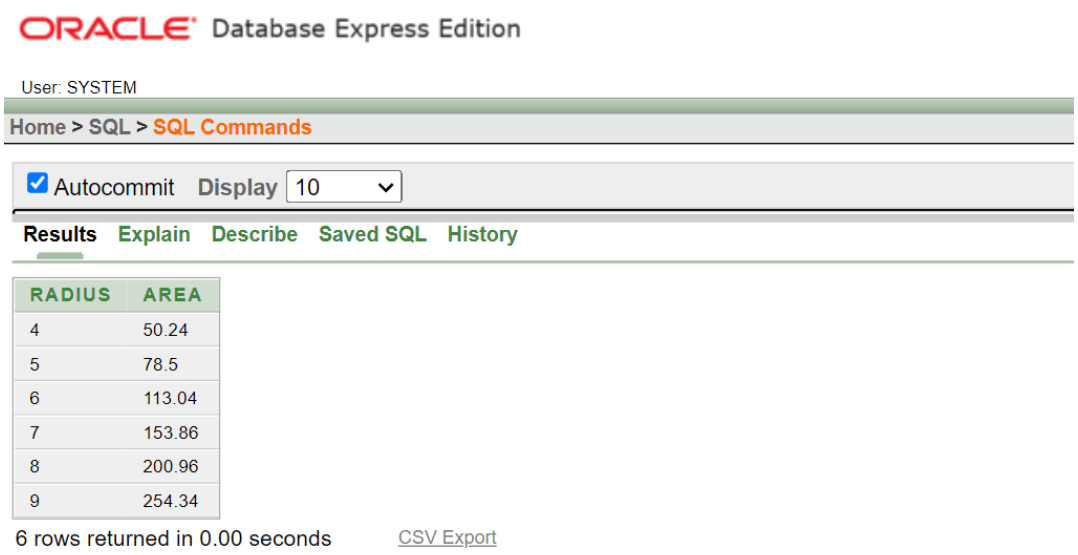
```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2));

DECLARE
pi constant number(4,2):=3.14;
radius number(5);
area number(14,2);

BEGIN
radius :=10;
while radius <=15
loop
area := pi*power(radius,2);
insert into areas values(radius,area);
radius := radius+1;
end loop;
end;
/
SELECT * FROM AREAS;
```



RADIUS	AREA
10	314
11	379.94
12	452.16
13	530.66
14	615.44
15	706.5

6 rows returned in 0.00 seconds CSV Export

```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2));

DECLARE
pi constant number(4,2):=3.14;
radius number(5);
area number(14,2);

BEGIN
radius :=4;
while radius <=9
loop
area := pi*power(radius,2);
insert into areas values(radius,area);
radius := radius+1;
end loop;
end;
/
SELECT * FROM AREAS;
```



Group A

Assignment Number 6

PL/SQL

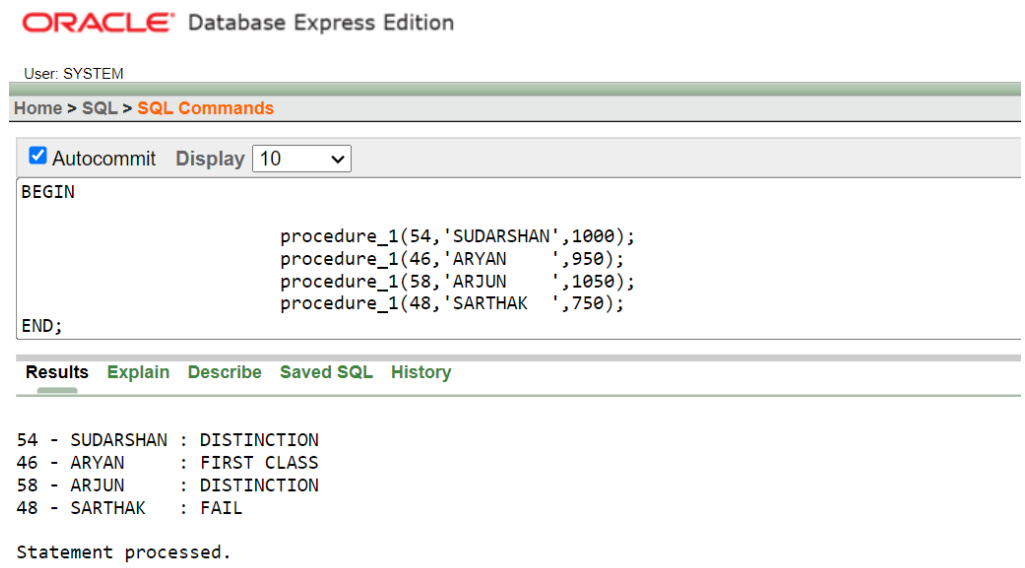
Problem Statement:
Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.
Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class).

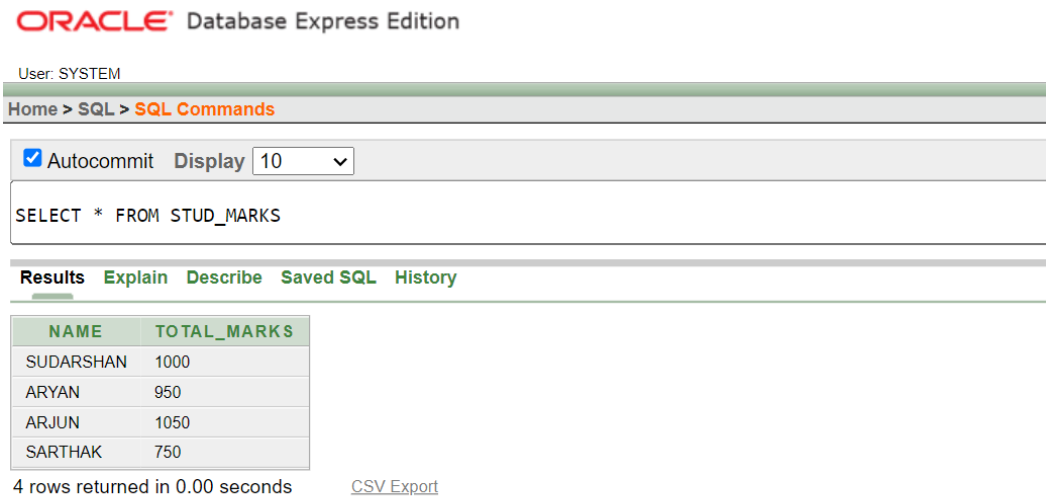
PROGRAM INPUT

```
CREATE TABLE stud_marks
(
name VARCHAR2(25),
total_marks NUMBER
);

CREATE TABLE result
(
roll_number NUMBER ,
name VARCHAR2(25),
class VARCHAR2(30)
);

CREATE OR REPLACE PROCEDURE procedure_1
( roll_no IN NUMBER, name IN VARCHAR2 ,marks IN NUMBER)
AS
BEGIN
        IF (marks<=1500 and marks>=990) THEN
                DBMS_OUTPUT.PUT_LINE (roll_no||' - '||name||' : DISTINCTION');
                INSERT INTO result VALUES (roll_no,name,'DISTINCTION');
        ELSIF (marks<=989 and marks>=900) THEN
                DBMS_OUTPUT.PUT_LINE (roll_no||' - '||name||' : FIRST CLASS');
                INSERT INTO result VALUES (roll_no,name,'FIRST CLASS');
        ELSIF (marks<=899 and marks>825) THEN
                DBMS_OUTPUT.PUT_LINE(roll_no||' - '||name||' : HIGHER SECOND CLASS');
                INSERT INTO result VALUES (roll_no,name,'HIGHER SECOND CLASS');
        ELSE
                DBMS_OUTPUT.PUT_LINE (roll_no||' - '||name||' : FAIL');
                INSERT INTO result VALUES (roll_no,name,'FAIL');
        END IF;
        INSERT INTO stud_marks VALUES (name,marks);
END procedure_1;
/
```

```
BEGIN

                        procedure_1(54,'SUDARSHAN',1000);
                        procedure_1(46,'ARYAN    ',950);
                        procedure_1(58,'ARJUN    ',1050);
                        procedure_1(48,'SARTHAK  ',750);
END;
/
```

OUTPUT





ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT * FROM RESULT
```

Results Explain Describe Saved SQL History

ROLL_NUMBER	NAME	CLASS
54	SUDARSHAN	DISTINCTION
46	ARYAN	FIRST CLASS
58	ARJUN	DISTINCTION
48	SARTHAK	FAIL

4 rows returned in 0.00 seconds CSV Export

Group A

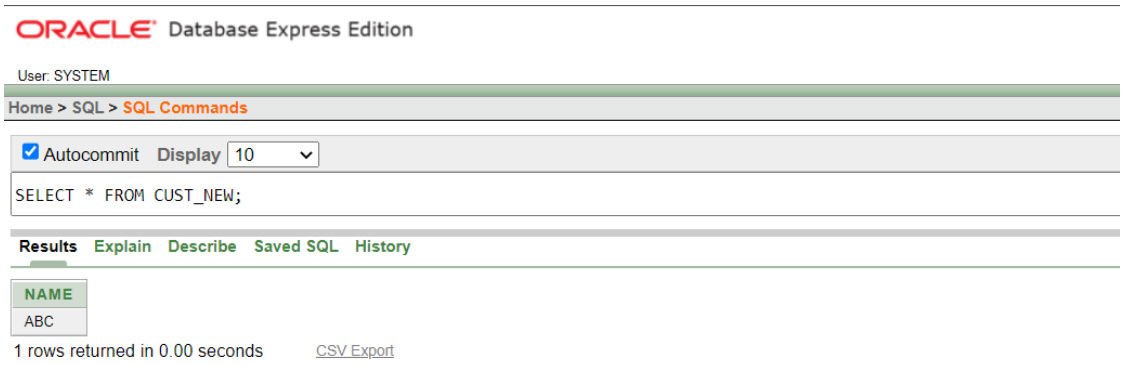
Assignment Number 7

Cursors

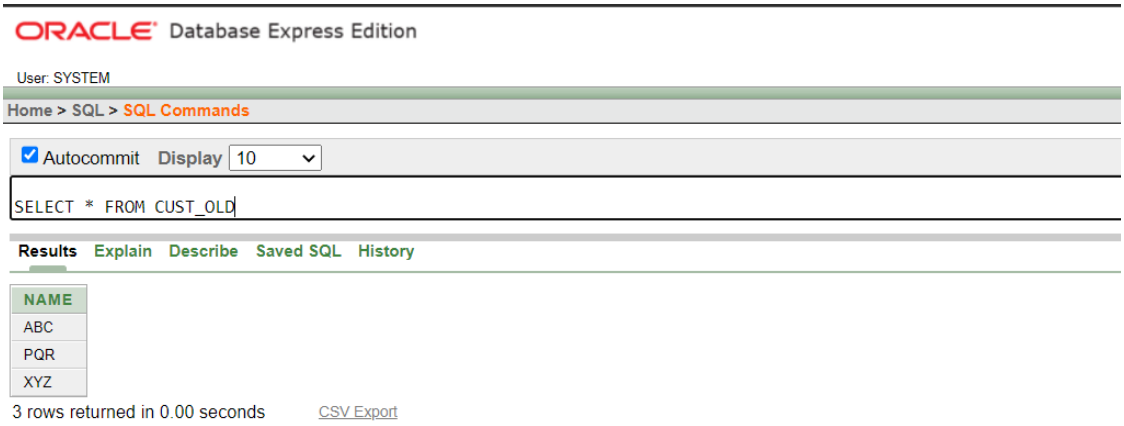
Problem Statement:
Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)
Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table Cust_New with the data available in the table Cust_Old. If the data in the first table already exist in the second table then that data should be skipped.

PROGRAM INPUT & OUTPUT

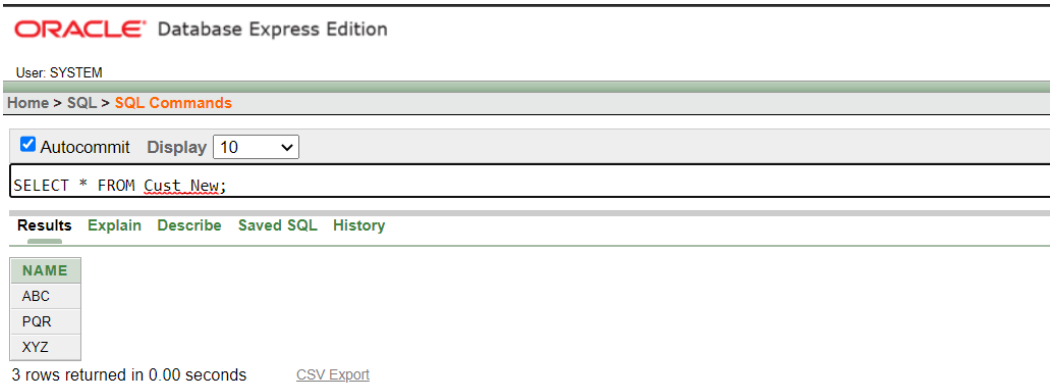
```
CREATE TABLE Cust_New
(
Name VARCHAR2(15)
);

INSERT INTO Cust_New VALUES ('ABC');
```




```
CREATE TABLE Cust_Old
(
Name VARCHAR2(15)
);

INSERT INTO Cust_Old VALUES ('ABC');
INSERT INTO Cust_Old VALUES ('PQR');
INSERT INTO Cust_Old VALUES ('XYZ');
```



```
DECLARE

CURSOR cur1 IS
SELECT Name from Cust_Old;

CURSOR cur2 IS
SELECT Name from Cust_New;

R VARCHAR(15);
C_Name VARCHAR(15);

BEGIN
OPEN cur1;
OPEN cur2;

LOOP
Fetch cur1 into C_Name;
Fetch cur2 into R;

EXIT WHEN cur1%FOUND = FALSE;
IF R <> C_Name THEN
INSERT INTO Cust_New VALUES (C_Name);
END IF;
END LOOP;
CLOSE cur1;
END;
/

SELECT * FROM Cust_New;
```



Group A

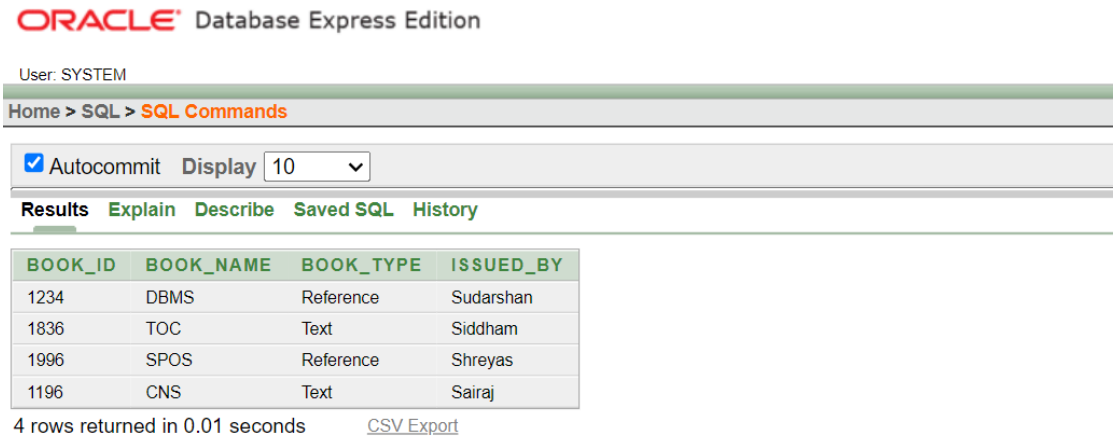
Assignment Number 8

Trigger

Problem Statement:
Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

PROGRAM INPUT

```
; Table Creation

CREATE TABLE Library
(
Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),
Issued_By VARCHAR2(20)
);

; Table Insertion

INSERT INTO Library VALUES (1234,'DBMS','Reference','Sudarshan');
INSERT INTO Library VALUES (1836,'TOC','Text','Siddham');
INSERT INTO Library VALUES (1996,'SPOS','Reference','Shreyas');
INSERT INTO Library VALUES (1196,'CNS','Text','Sairaj');
```




```
; Table Creation

CREATE TABLE Back_UP
(
Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),
```

```
Issued_By VARCHAR2(20)
);

; Trigger Creation

CREATE TRIGGER Update_Rec
AFTER UPDATE OR DELETE ON Library
FOR EACH ROW

BEGIN
INSERT INTO Back_UP
(Book_Id, Book_Name, Book_Type, Issued_By)
VALUES
(:old.Book_Id, :old.Book_Name, :old.Book_Type, :old.Issued_By);
END;
/

UPDATE LIBRARY
SET Issued_By = 'Sairaj'
WHERE Issued_By = 'Sumit';

SELECT * FROM Back_UP;
```