SISTER NIVEDITA UNIVERSITY

DATABASE MANAGEMENT SYSTEM

SUBMITTED BY: **PIYUSH CHANDRA CHANDRA**

DEPT- BTECH (CSE), ROLL- 1027

SUBMITTED TO: **SWARUP KUMAR GHOSH &**

DEBANJAN DAS

ASSIGNMENT 8

16/12/2020 – 22/12/2020

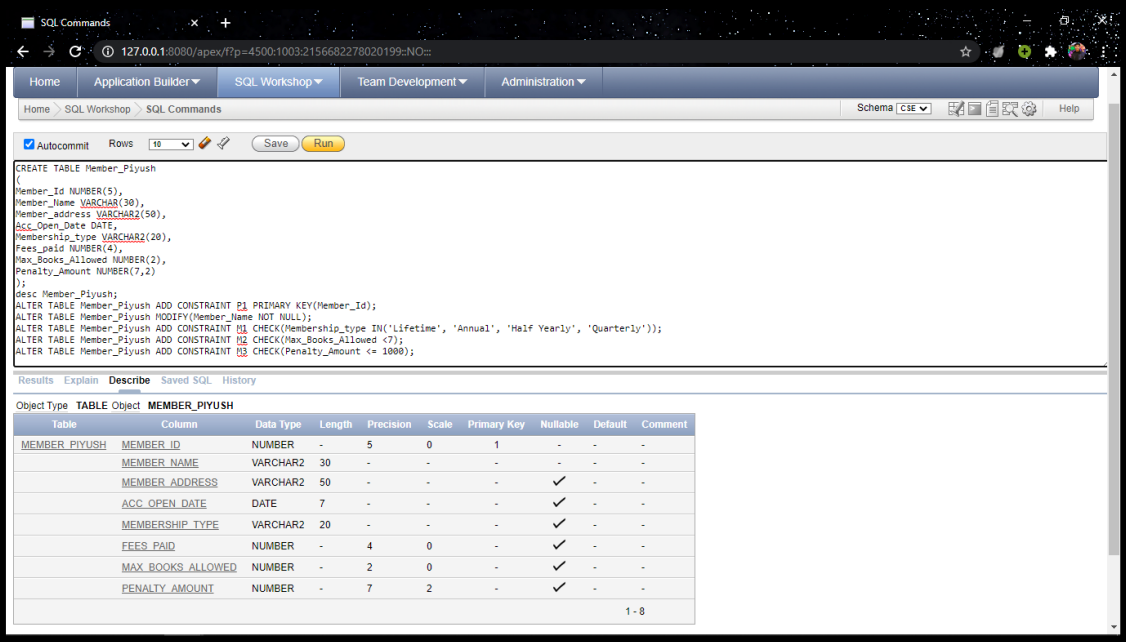
A) **Table Name : Member**

COLUMN NAME	DATA TYPE	DESCRIPTION
Member_Id	Number(5)	Unique Member ID
Member_Name	Varchar2(30)	Name of the Library member
Member_address	Varchar2(50)	Address of the member
Acc_Open_Date	Date	Date of membership
Membership_type	Varchar2(20)	Type of the membership such as 'Lifetime',' Annual', 'Half Yearly',' Quarterly'
Fees_paid	Number(4)	Membership fees paid
Max_Books_Allowed	Number(2)	Total Number of books that can be issued to the member.
Penalty_Amount	Number(7,2)	Penalty amount due

CONSTRAINT:

a. Member_Id – Primary Key
b. Member_Name – NOT NULL
c. Membership_type - 'Lifetime',' Annual', 'Half Yearly',' Quarterly'
d. Max_books_allowed <7
e. Penalty_amt maximum 1000

```
CREATE TABLE Member_Piyush
(
Member_Id NUMBER(5),
Member_Name VARCHAR(30),
Member_address VARCHAR2(50),
Acc_Open_Date DATE,
Membership_type VARCHAR2(20),
Fees_paid NUMBER(4),
Max_Books_Allowed NUMBER(2),
Penalty_Amount NUMBER(7,2)
);
desc Member_Piyush;
ALTER TABLE Member_Piyush ADD CONSTRAINT P1 PRIMARY KEY(Member_Id);
ALTER TABLE Member_Piyush MODIFY(Member_Name NOT NULL);
ALTER TABLE Member_Piyush ADD CONSTRAINT M1 CHECK(Membership_type
IN('Lifetime', 'Annual', 'Half Yearly', 'Quarterly'));
ALTER TABLE Member_Piyush ADD CONSTRAINT M2 CHECK(Max_Books_Allowed
<7);
ALTER TABLE Member_Piyush ADD CONSTRAINT M3 CHECK(Penalty_Amount <=
1000);
SELECT SYSDATE FROM DUAL;
```


SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:2156682278020199::NO:::
Home
Application Builder
SQL Workshop
Team Development
Administration
Home > SQL Workshop > SQL Commands
Schema CSE
Help
Autocommit
Rows 10
Save
Run
CREATE TABLE Member_Piyush
(
Member_Id NUMBER(5),
Member_Name VARCHAR(30),
Member_address VARCHAR2(50),
Acc_Open_Date DATE,
Membership_type VARCHAR2(20),
Fees_paid NUMBER(4),
Max_Books_Allowed NUMBER(2),
Penalty_Amount NUMBER(7,2)
);
desc Member_Piyush;
ALTER TABLE Member_Piyush ADD CONSTRAINT P1 PRIMARY KEY(Member_Id);
ALTER TABLE Member_Piyush MODIFY(Member_Name NOT NULL);
ALTER TABLE Member_Piyush ADD CONSTRAINT M1 CHECK(Membership_type IN('Lifetime', 'Annual', 'Half Yearly', 'Quarterly'));
ALTER TABLE Member_Piyush ADD CONSTRAINT M2 CHECK(Max_Books_Allowed <7);
ALTER TABLE Member_Piyush ADD CONSTRAINT M3 CHECK(Penalty_Amount <= 1000);
Results Explain Describe Saved SQL History
Object Type TABLE Object MEMBER_PIYUSH
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment
MEMBER_PIYUSH MEMBER_ID NUMBER - 5 0 1 - - -
MEMBER_NAME VARCHAR2 30 - - - - - -
MEMBER_ADDRESS VARCHAR2 50 - - - ✓ - -
ACC_OPEN_DATE DATE 7 - - - ✓ - -
MEMBERSHIP_TYPE VARCHAR2 20 - - - ✓ - -
FEES_PAID NUMBER - 4 0 - ✓ - -
MAX_BOOKS_ALLOWED NUMBER - 2 0 - ✓ - -
PENALTY_AMOUNT NUMBER - 7 2 - ✓ - -
1 - 8

B) Table Name : BOOKS

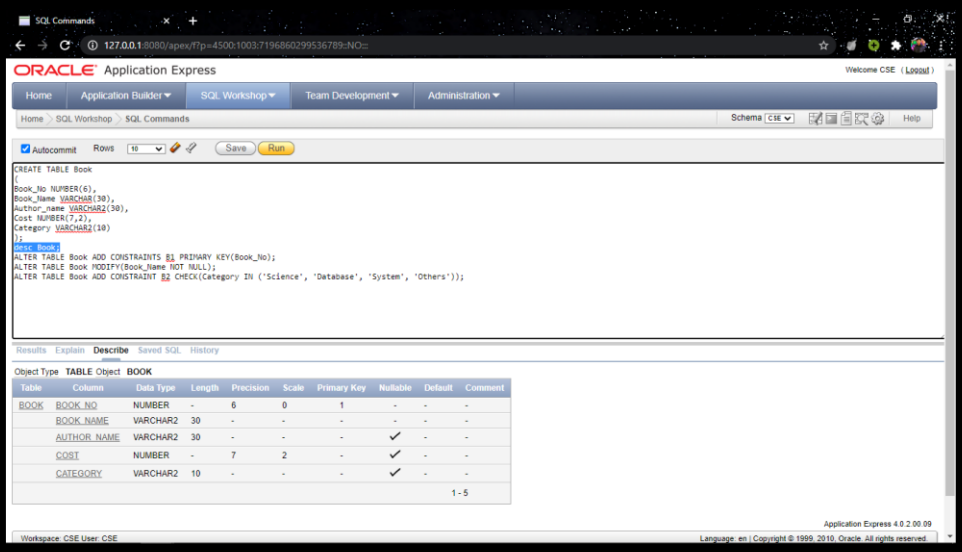
COLUMN NAME	DATA TYPE	DESCRIPTION
Book_No	Number(6)	Book identification number
Book_Name	VarChar2(30)	Name of the book
Author_name	Varchar2(30)	Author of the book
Cost	Number(7,2)	Cost of the book
Category	Char(10)	Category like Science , Fiction etc.

CONSTRAINT:

a. Book_No – Primary Key
b. Book_Name – Not Null
c. Category – Science, Database, System, Others.

- CREATE TABLE Book_Piyush

```
(
Book_No Number(6) PRIMARY KEY,
Book_Name VarChar2(30) NOT NULL,
Author_name Varchar2(30),
Cost Number(7,2),
Category VARCHAR2(10),
);
DESC Book_Piyush;
ALTER TABLE Books ADD CONSTRAINTS B1 PRIMARY KEY(Book_no);
ALTER TABLE Books MODIFY(Book_name NOT NULL);
ALTER TABLE Books ADD CONSTRAINT B2 CHECK(Category IN ('Science', 'Database', 'System', 'Others'));
```



C) **Table Name : ISSUE**

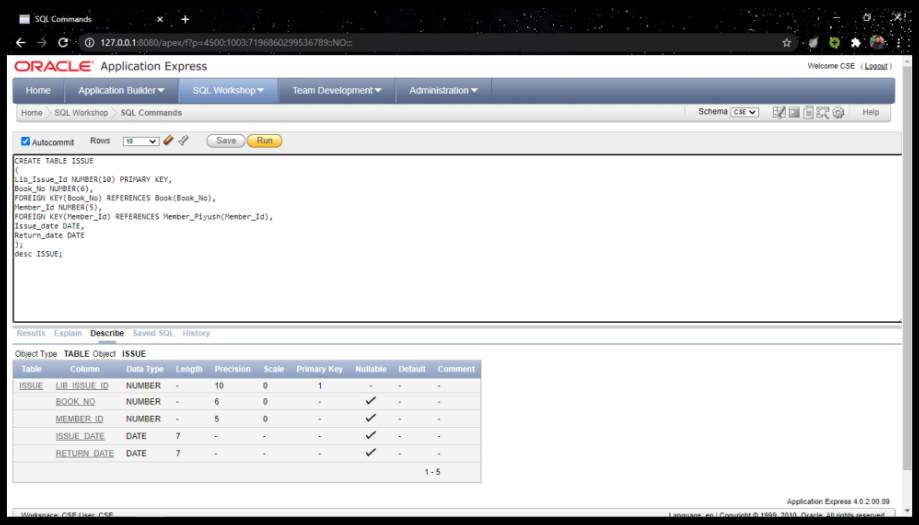
COLUMN NAME	DATA TYPE	DESCRIPTION
Lib_Issue_Id	Number(10)	Library Book Issue No
Book_No	Number(6)	The ID of book, which is issued
Member_Id	Number(5)	Member that issued the book
Issue_Date	Date	Date of Issue
Return_date	Date	Return date

CONSTRAINT:

a. Lib_Issue_Id -Primary key
b. Book_No - foreign key
c. Member_id - foreign key

- CREATE TABLE ISSUE

```
(
Lib_Issue_Id NUMBER(10) PRIMARY KEY,
Book_No NUMBER(6),
FOREIGN KEY(Book_No) REFERENCES Book(Book_No),
Member_Id NUMBER(5),
FOREIGN KEY(Member_Id) REFERENCES Member_Piyush(Member_Id),
Issue_date DATE,
Return_date DATE
);
desc ISSUE;
```



- **Insert the following data to the appropriate table using SQL command.**
  A. Table Name : Member

MEMBER_ID	MEMBER_NAME	MEMBER_ADDRESS	ACC_OPEN_DATE	MEMBERSHIP_TYPE	FEES_PAID	MAX_BOOKS_ALLOWED	PENALTY_AMOUNT
1	Sayantan Sinha	Pune	10-Dec-10	Lifetime	2000	6	50
2	Abhirup Sarkar	Kolkata	19-jan-11	Annual	1400	3	0
3	Ritesh Bhuniya	Gujarat	20-feb-11	Quarterly	350	2	100
4	Paresh sen	Tripura	21-mar-11	Half yearly	700	1	200
5	Sohini Haldar	Birbhum	11-apr-11	Lifetime	2000	6	10
6	Suparna Biswas	Kolkata	12-apr-11	Half Yearly	700	1	0
7	Suranjana Basu	Purulia	30-june-11	Annual	1400	3	50
8	Arpita Roy	Kolkata	31-july-11	Half yearly	700	1	0

INSERT INTO Member_Piyush VALUES(1,'SAYANTAN SINHA','PUNE','12/10/2010','Lifetime',2000,6,50);

INSERT INTO Member_Piyush VALUES(2,'ABHIRUP SARKAR','KOLKATA','01/19/2011','Annual',1400,3,0);

INSERT INTO Member_Piyush VALUES(3,'Ritesh Bhuniya','Gujarat','02/20/2011','Quarterly',350,2,100);

INSERT INTO Member_Piyush VALUES(4,'Paresh sen','Tripura','03/21/2011','Half Yearly',700,1,200);

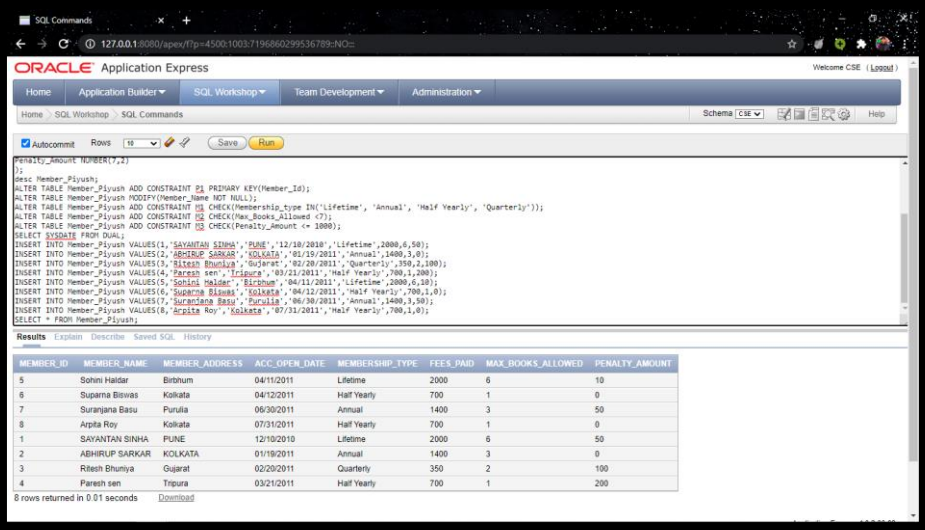
INSERT INTO Member_Piyush VALUES(5,'Sohini Haldar','Birbhum','04/11/2011','Lifetime',2000,6,10);

INSERT INTO Member_Piyush VALUES(6,'Suparna Biswas','Kolkata','04/12/2011','Half Yearly',700,1,0);

INSERT INTO Member_Piyush VALUES(7,'Suranjana Basu','Purulia','06/30/2011','Annual',1400,3,50);

INSERT INTO Member_Piyush VALUES(8,'Arpita Roy','Kolkata','07/31/2011','Half Yearly',700,1,0);

SELECT * FROM Member_Piyush;



B. Table Name : BOOKS

BOOK_NO	BOOK_NAME	AUTHOR_NAME	COST	CATEGORY
101	Let us C	Denis Ritchie	450	Others
102	Oracle – Complete Ref	Loni	550	Database
103	Visual Basic 10	BPB	700	Others
104	Mastering SQL	Loni	450	Database
105	PL SQL-Ref	Scott Urman	750	Database
106	UNIX	Sumitava Das	300	System
107	Optics	Ghatak	600	Science
108	Data Structure	G.S. Baluja	350	Others

- INSERT INTO Book VALUES(101, 'Let us C', 'Denis Ritchie', 450 ,'Others')

  INSERT INTO Book VALUES(102, 'Oracle – Complete Ref','Loni' ,550, 'Database')

  INSERT INTO Book VALUES(103, 'Visual Basic 10', 'BPB' ,700 ,'Others')

  INSERT INTO Book VALUES(104, 'Mastering SQL', 'Loni', 450 ,'Database')

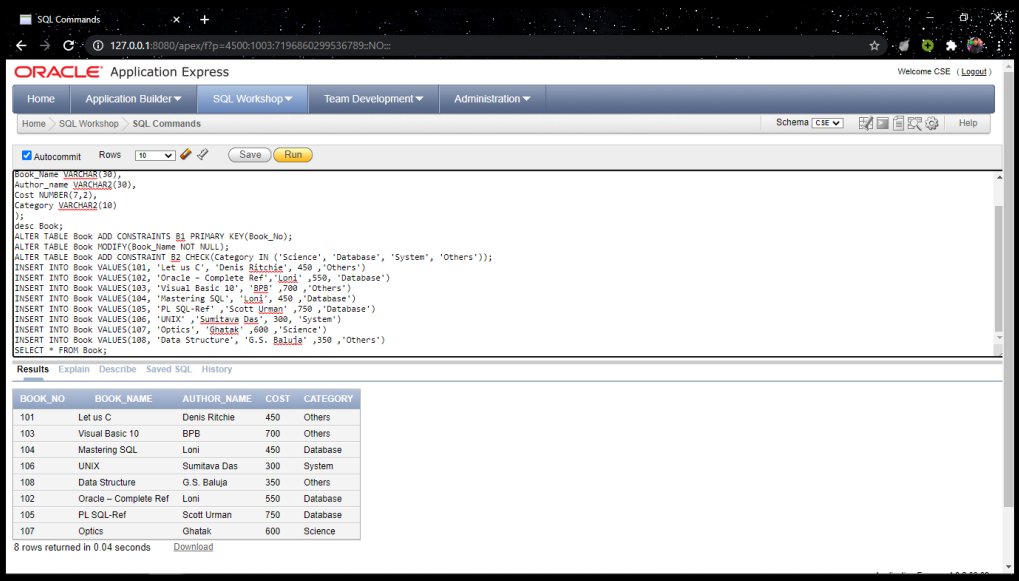
  INSERT INTO Book VALUES(105, 'PL SQL-Ref' ,'Scott Urman' ,750 ,'Database')

  INSERT INTO Book VALUES(106, 'UNIX' ,'Sumitava Das', 300, 'System')

  INSERT INTO Book VALUES(107, 'Optics', 'Ghatak' ,600 ,'Science')

  INSERT INTO Book VALUES(108, 'Data Structure', 'G.S. Baluja' ,350 ,'Others')

  SELECT * FROM Book;



C. Table Name : **ISSUE**

LIB_ISSUE_ID	BOOK_NO	MEMBER_ID	ISSUE_DATE	RETURN_DATE
7001	101	1	10-jan-11	
7002	102	2	25-jan-11	
7003	104	1	1-Feb-11	
7004	104	2	15-Mar-11	
7005	101	4	04-Apr-11	
7006	108	5	12-apr-11	
7007	101	8	1-Aug-11	

- INSERT INTO ISSUE VALUES(7001, 101, 1, '01/10/11', NULL)

  INSERT INTO ISSUE VALUES(7002, 102, 2, '01/25/11', NULL)

  INSERT INTO ISSUE VALUES(7003, 104, 1, '02/01/11', NULL)

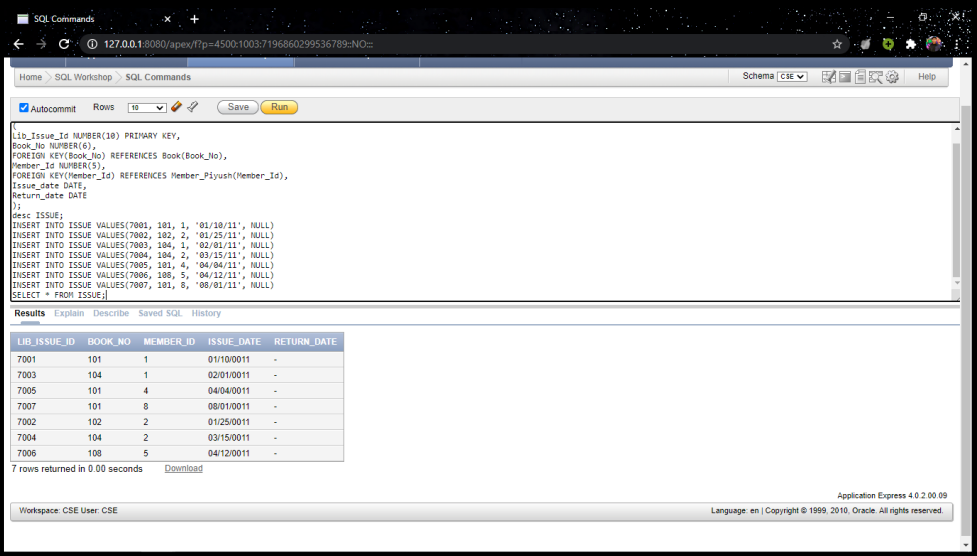
  INSERT INTO ISSUE VALUES(7004, 104, 2, '03/15/11', NULL)

  INSERT INTO ISSUE VALUES(7005, 101, 4, '04/04/11', NULL)

  INSERT INTO ISSUE VALUES(7006, 108, 5, '04/12/11', NULL)

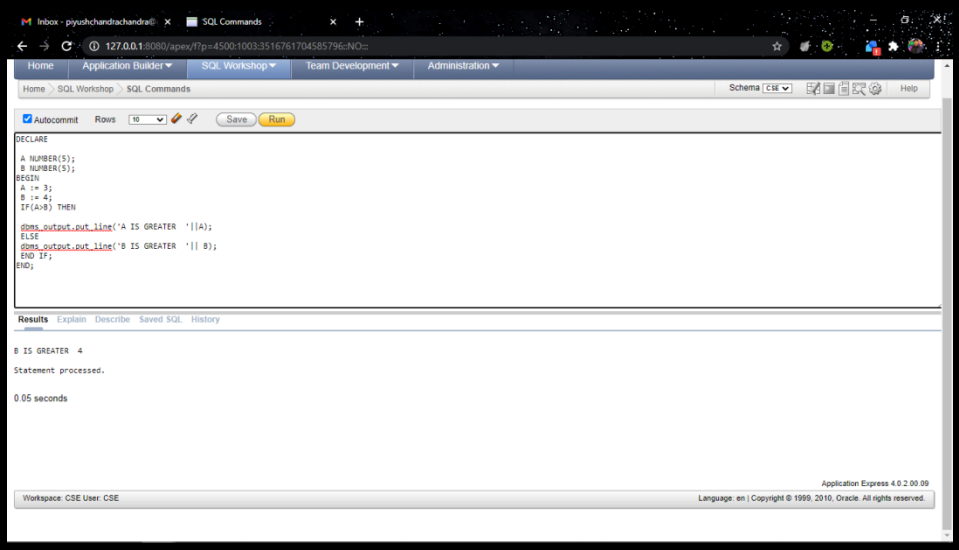
  INSERT INTO ISSUE VALUES(7007, 101, 8, '08/01/11', NULL)

  SELECT * FROM ISSUE;



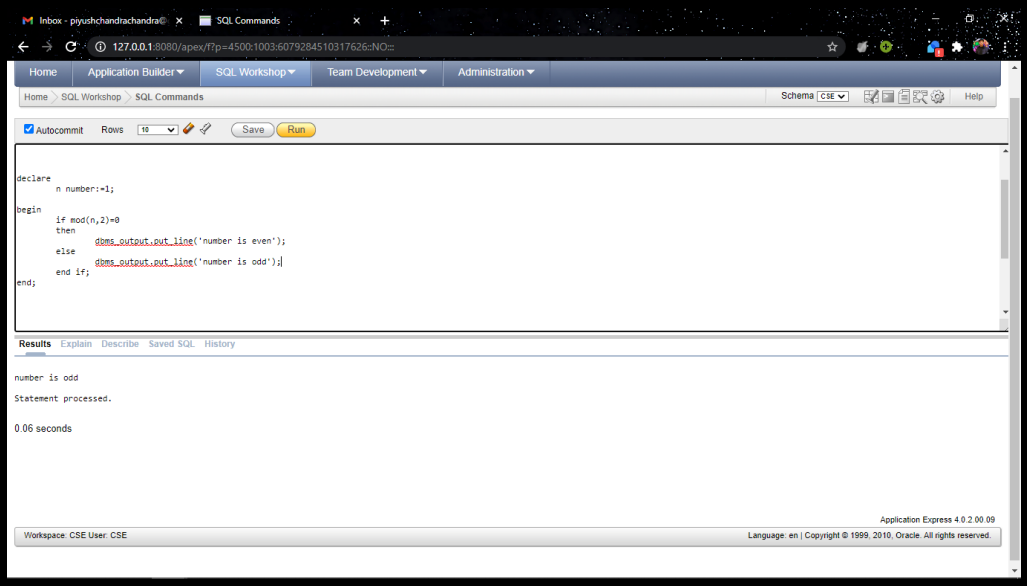
1) **Write a PL/SQL program where take input of two numbers and display the largest number.**

```
➢ DECLARE
    A NUMBER(5);
    B NUMBER(5);
  BEGIN
    A := 3;
    B := 4;
    IF(A>B) THEN

    dbms_output.put_line('A IS GREATER  '||A);
    ELSE
    dbms_output.put_line('B IS GREATER  '|| B);
    END IF;
  END;
```



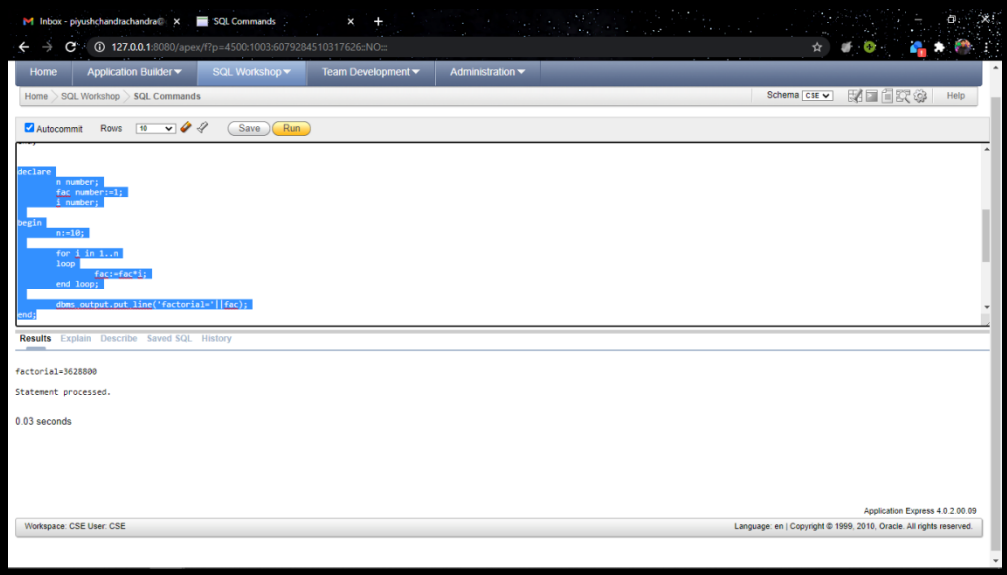
2) Write a PL/SQL program where take input of any number and display whether it is even or odd.

```
➢ declare
      n number:=1;

  begin
      if mod(n,2)=0
      then
            dbms_output.put_line('number is even');
      else
            dbms_output.put_line('number is odd');
      end if;
  end;
```



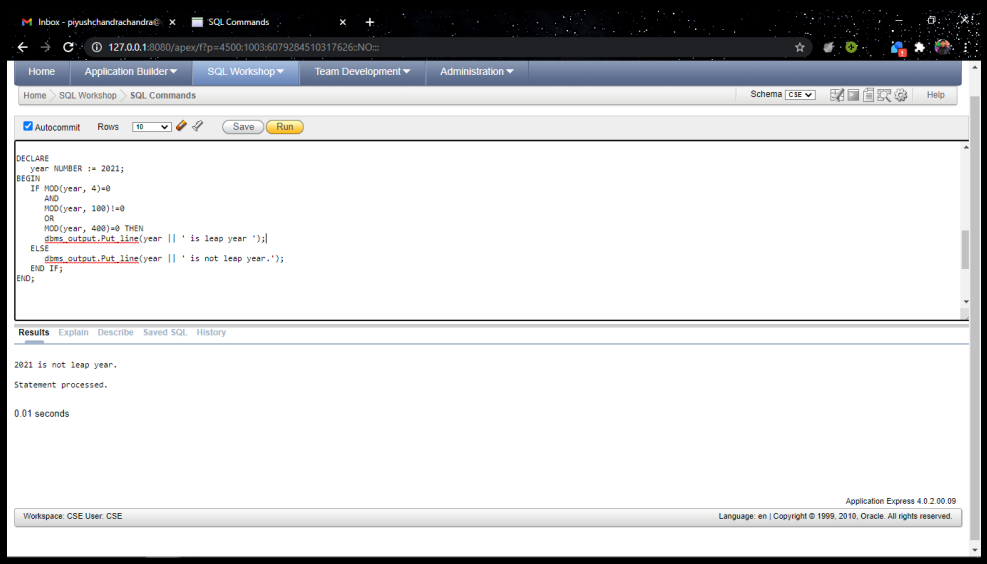
3) Write a PL/SQL program where take input of any number and find factorial of the given number.

```
➢ declare
        n number;
        fac number:=1;
        i number;

  begin
        n:=10;

        for i in 1..n
        loop
                fac:=fac*i;
        end loop;

        dbms_output.put_line('factorial='||fac);
  end;
```



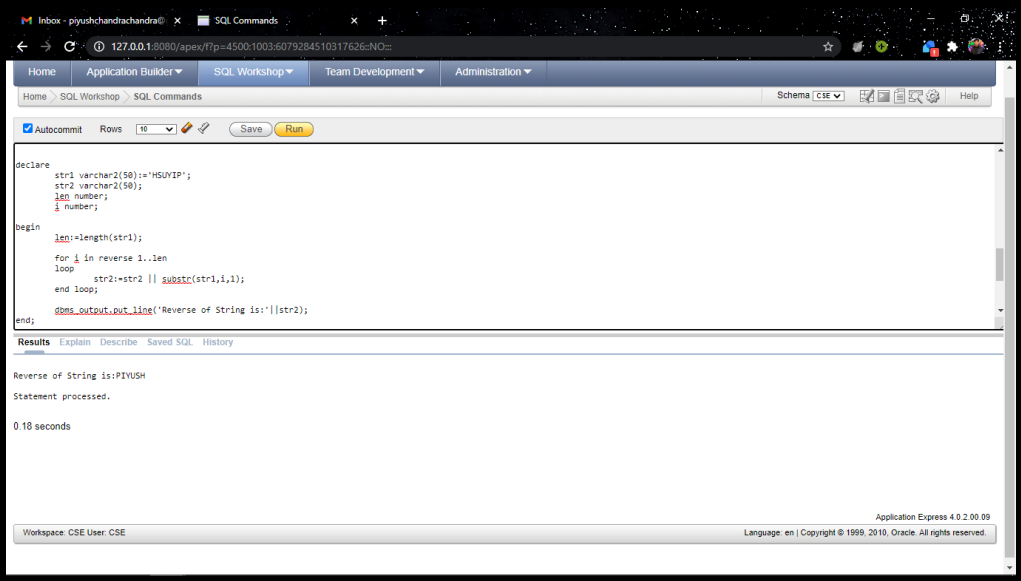
4) Write a PL/SQL program where check a year leap year or not. Take a year as user input and check it is leap year or not.

```
➢ DECLARE
    year NUMBER := 2021;
  BEGIN
    IF MOD(year, 4)=0
      AND
      MOD(year, 100)!=0
      OR
      MOD(year, 400)=0 THEN
      dbms_output.Put_line(year || ' is leap year ');
    ELSE
      dbms_output.Put_line(year || ' is not leap year.');
    END IF;
  END;
```



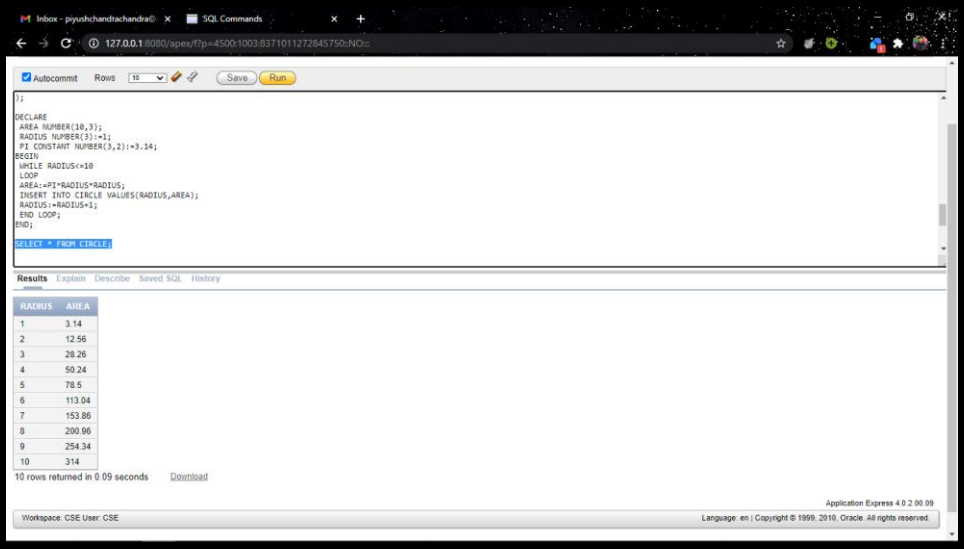
5) Write a PL/SQL program where take a string as input and print the reverse of it.

```
➢ declare
        str1 varchar2(50):='HSUYIP';
        str2 varchar2(50);
        len number;
        i number;

begin
        len:=length(str1);

        for i in reverse 1..len
        loop
                str2:=str2 || substr(str1,i,1);
        end loop;

        dbms_output.put_line('Reverse of String is:'||str2);
end;
```



6) **Create a table named CIRCLE with two attributes RADIUS number (3) and AREA number(10,3), then write a PL/SQL program which can calculate area for every radius up to 10 and insert into the table. (Use while and for loop individually).**

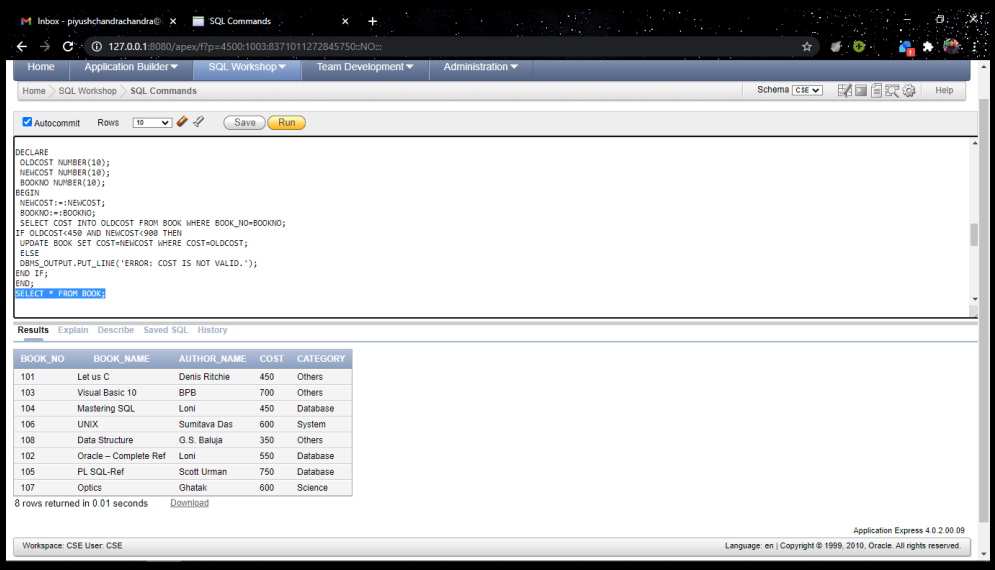
```
CREATE TABLE CIRCLE
(
RADIUS NUMBER(3),
AREA NUMBER(10,3)
);

DECLARE
 AREA NUMBER(10,3);
 RADIUS NUMBER(3):=1;
 PI CONSTANT NUMBER(3,2):=3.14;
BEGIN
 WHILE RADIUS<=10
 LOOP
 AREA:=PI*RADIUS*RADIUS;
 INSERT INTO CIRCLE VALUES(RADIUS,AREA);
 RADIUS:=RADIUS+1;
 END LOOP;
END;

SELECT * FROM CIRCLE;
```



7) Write a PL/SQL program which can update cost value of corresponding book number of the BOOKS_COPY Table.

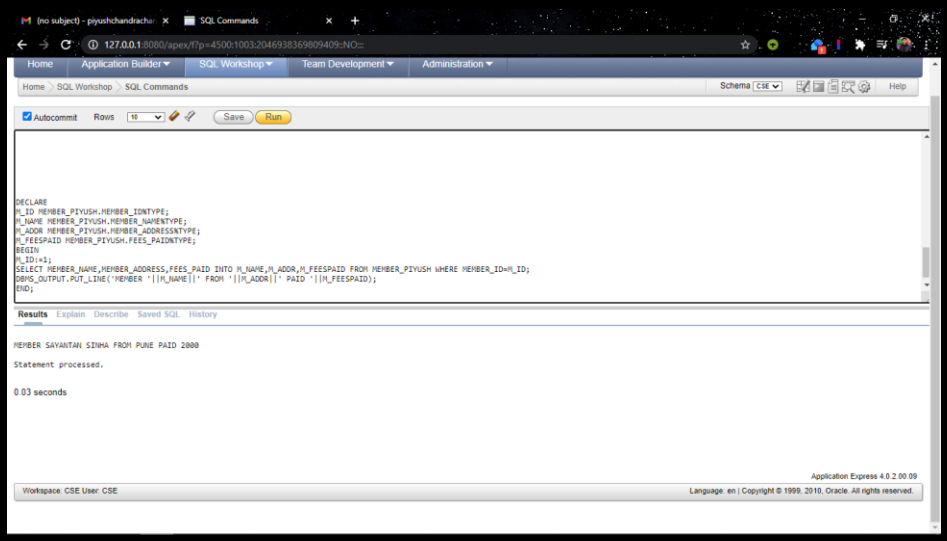
- ☐ **INPUT: BOOK_NO, NEW COST,**
- ☐ **CONDITION: Old cost value will less than 450 and new cost value will less Than 900 otherwise provide an error massage.**

```
➢ DECLARE
   OLDCOST NUMBER(10);
   NEWCOST NUMBER(10);
   BOOKNO NUMBER(10);
  BEGIN
   NEWCOST:=:NEWCOST;
   BOOKNO:=:BOOKNO;
   SELECT COST INTO OLDCOST FROM BOOK WHERE
  BOOK_NO=BOOKNO;
  IF OLDCOST<450 AND NEWCOST<900 THEN
   UPDATE BOOK SET COST=NEWCOST WHERE COST=OLDCOST;
   ELSE
   DBMS_OUTPUT.PUT_LINE('ERROR: COST IS NOT VALID.');
  END IF;
  END;
```



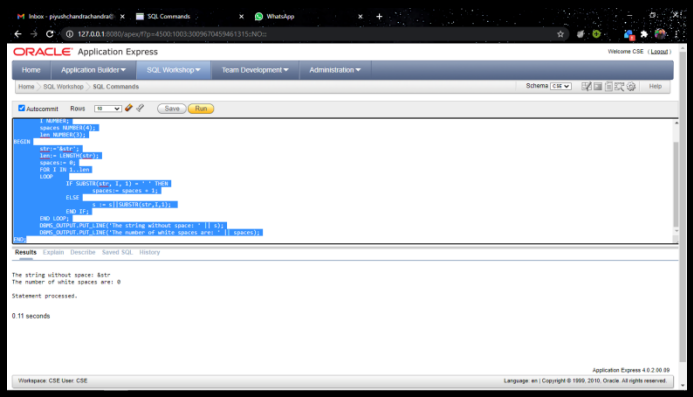
8. Write a PL/SQL Program which take MEMBER_ID as input and provide the corresponding MEMBER_NAME, MEMBER_ADDRESS AND FEES PAID.

```
DECLARE
M_ID MEMBER_PIYUSH.MEMBER_ID%TYPE;
M_NAME MEMBER_PIYUSH.MEMBER_NAME%TYPE;
M_ADDR MEMBER_PIYUSH.MEMBER_ADDRESS%TYPE;
M_FEESPAID MEMBER_PIYUSH.FEES_PAID%TYPE;
BEGIN
M_ID:=1;
SELECT MEMBER_NAME,MEMBER_ADDRESS,FEES_PAID INTO
M_NAME,M_ADDR,M_FEESPAID FROM MEMBER_PIYUSH WHERE
MEMBER_ID=M_ID;
DBMS_OUTPUT.PUT_LINE('MEMBER '||M_NAME||' FROM '||M_ADDR||'
PAID '||M_FEESPAID);
END;
```



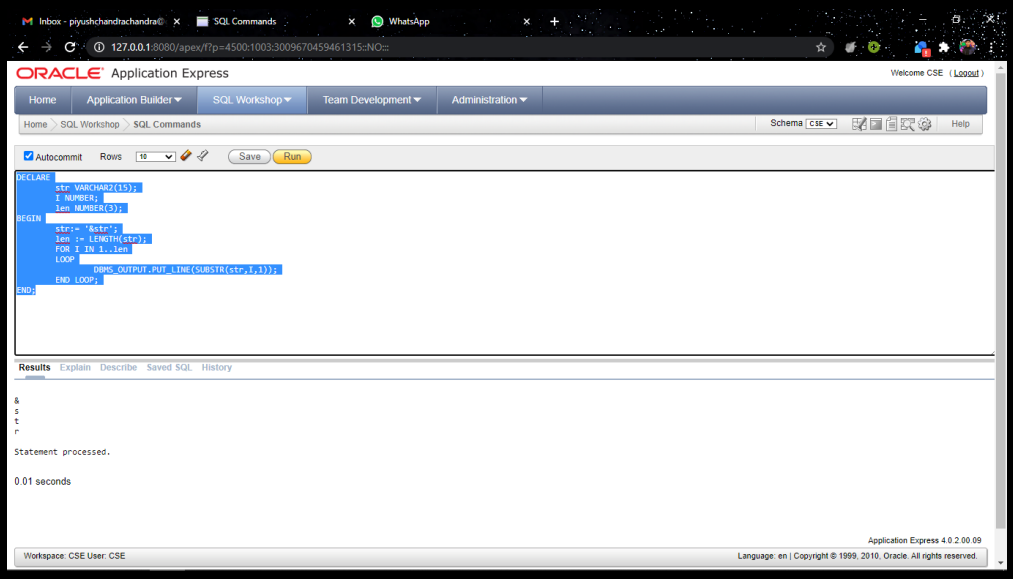
9) Write a PL/SQL program which can take a String as an input and display it without any space and also count the no of space available in the input string.

```
➢ DECLARE
        str VARCHAR2(15);
        s VARCHAR2(15);
        I NUMBER;
        spaces NUMBER(4);
        len NUMBER(3);
  BEGIN
        str:='&str';
        len:= LENGTH(str);
        spaces:= 0;
        FOR I IN 1..len
        LOOP
            IF SUBSTR(str, I, 1) = ' ' THEN
                spaces:= spaces + 1;
            ELSE
                s := s||SUBSTR(str,I,1);
            END IF;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('The string without space: ' || s);
        DBMS_OUTPUT.PUT_LINE('The number of white spaces are: ' || spaces);
  END;
```



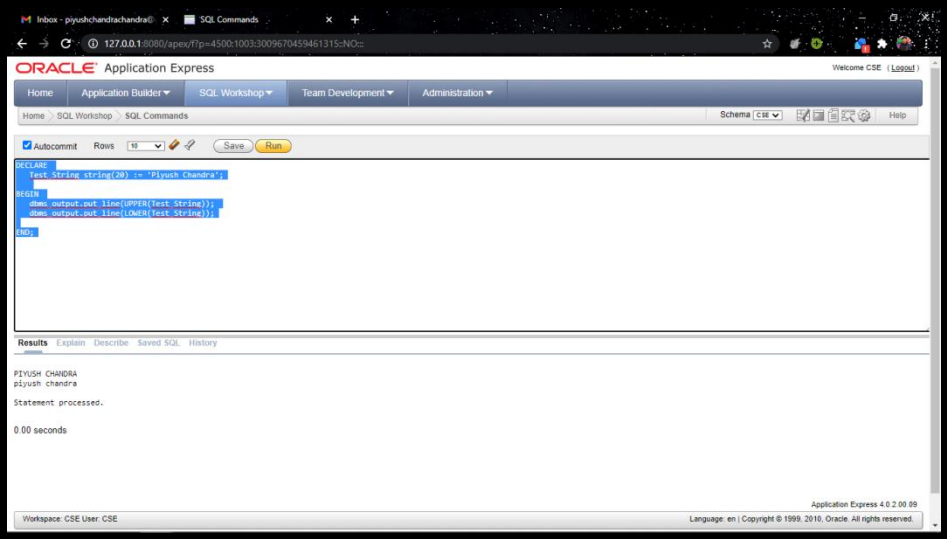
10) Take an input of any string and display each word in a separate line.

```
➢ DECLARE
        str VARCHAR2(15);
        I NUMBER;
        len NUMBER(3);
  BEGIN
        str:= '&str';
        len := LENGTH(str);
        FOR I IN 1..len
  LOOP
        DBMS_OUTPUT.PUT_LINE(SUBSTR(str,I,1));
  END LOOP;
END;
```



11) Take an input of any Member Number and display the Member Name in upper case and lower case.

```
DECLARE
   Test_String string(20) := 'Piyush Chandra';
BEGIN
   dbms_output.put_line(UPPER(Test_String));
   dbms_output.put_line(LOWER(Test_String));
END;
```

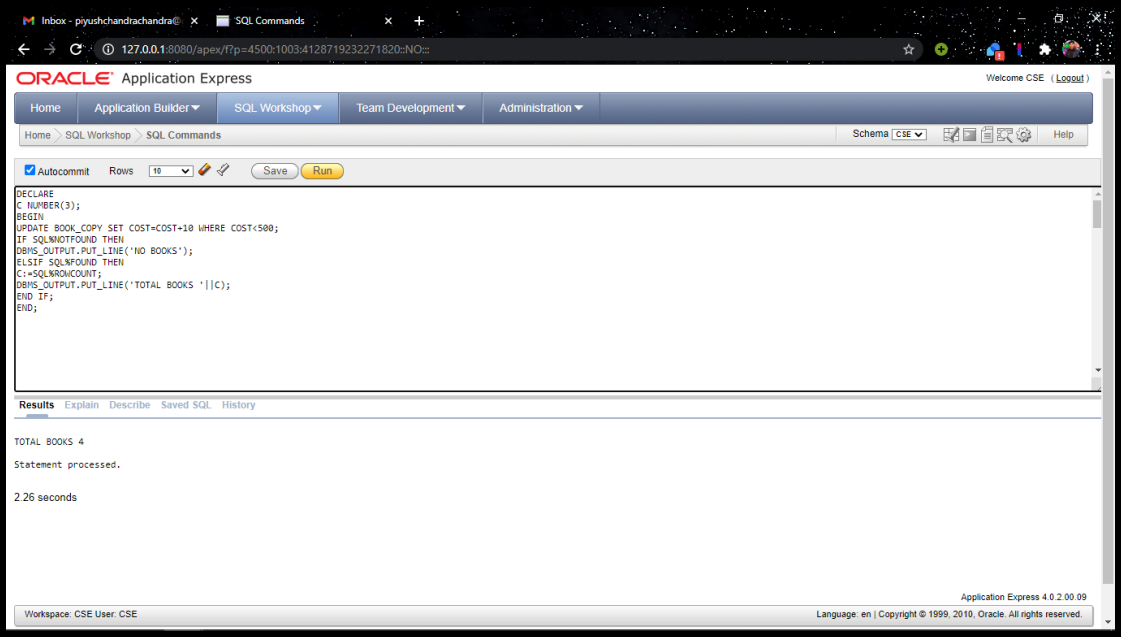


ASSIGNMENT 9

23/12/2020 – 29/12/2020

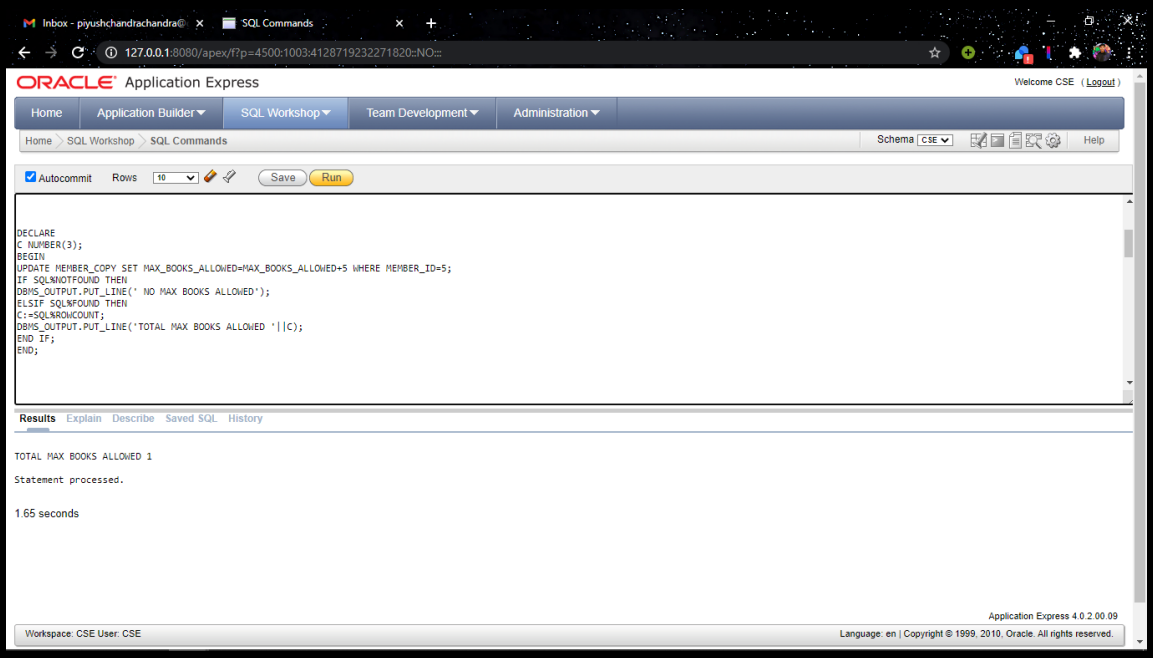
1. Write a PL/SQL program which can update the cost of BOOKS_COPY table with 10 more cost where cost is less than 500 and show how many rows are affected (Use Implicit Cursor SQL%ROWCOUNT).

➢
```
DECLARE
C NUMBER(3);
BEGIN
UPDATE BOOK_COPY SET COST=COST+10 WHERE COST<500;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('NO BOOKS');
ELSIF SQL%FOUND THEN
C:=SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE('TOTAL BOOKS '||C);
END IF;
END;
```



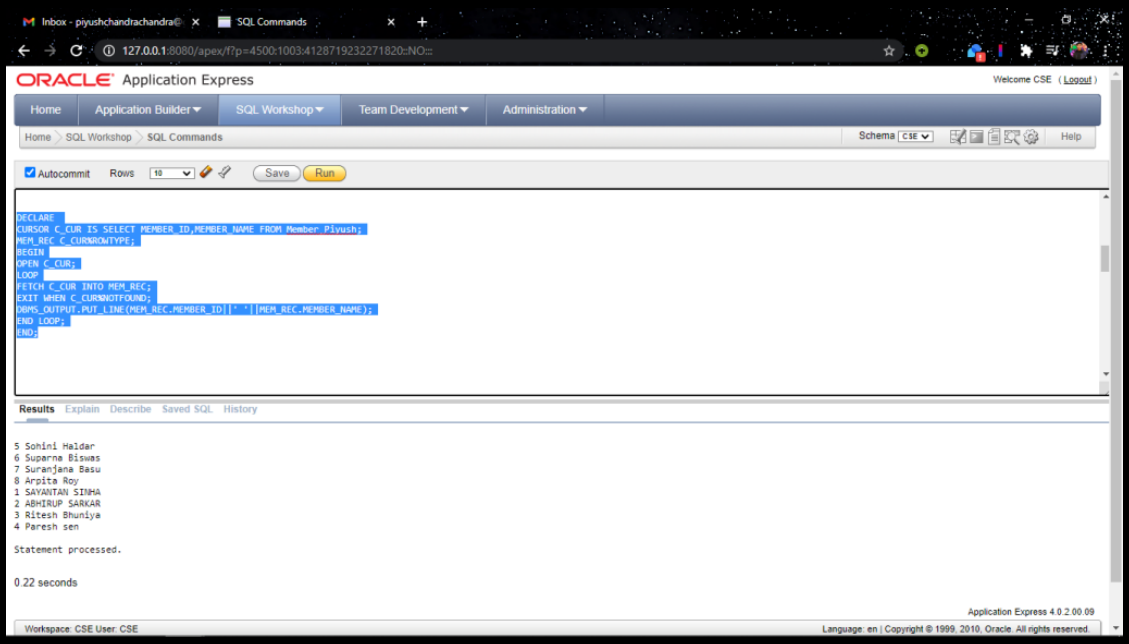
2. Write a PL/SQL program which can increment the value of MAX_BOOKS_ALLOWED of MEMBER_COPY table with 2 where MEMBER_ID = 5, and show a message if update is possible. (Use Implicit Cursor SQL%FOUND).

➢ DECLARE
C NUMBER(3);
BEGIN
UPDATE MEMBER_COPY SET
MAX_BOOKS_ALLOWED=MAX_BOOKS_ALLOWED+5 WHERE
MEMBER_ID=5;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE(' NO MAX BOOKS ALLOWED');
ELSIF SQL%FOUND THEN
C:=SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE('TOTAL MAX BOOKS ALLOWED '||C);
END IF;
END;



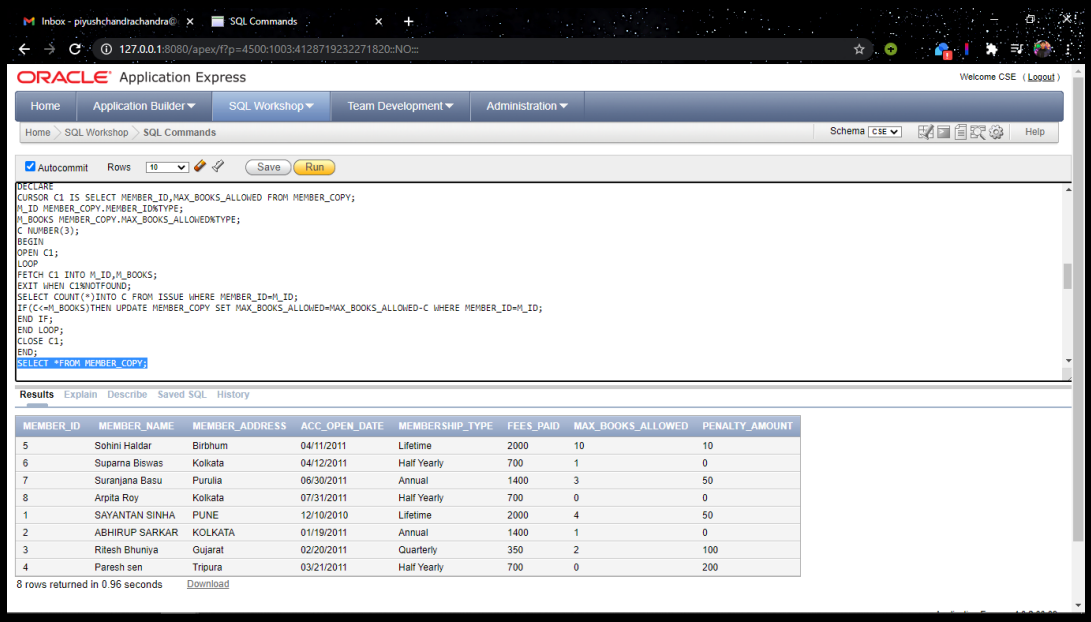
3. Write a PL/SQL Program using Explicit Cursor and show the Member_ID, Member Name for every attribute of Member.

➢ DECLARE
CURSOR C_CUR IS SELECT MEMBER_ID,MEMBER_NAME FROM MEMBER;
MEM_REC C_CUR%ROWTYPE;
BEGIN
OPEN C_CUR;
LOOP
FETCH C_CUR INTO MEM_REC;
EXIT WHEN C_CUR%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(MEM_REC.MEMBER_ID||' '||MEM_REC.MEMBER_NAME);
END LOOP;
END;



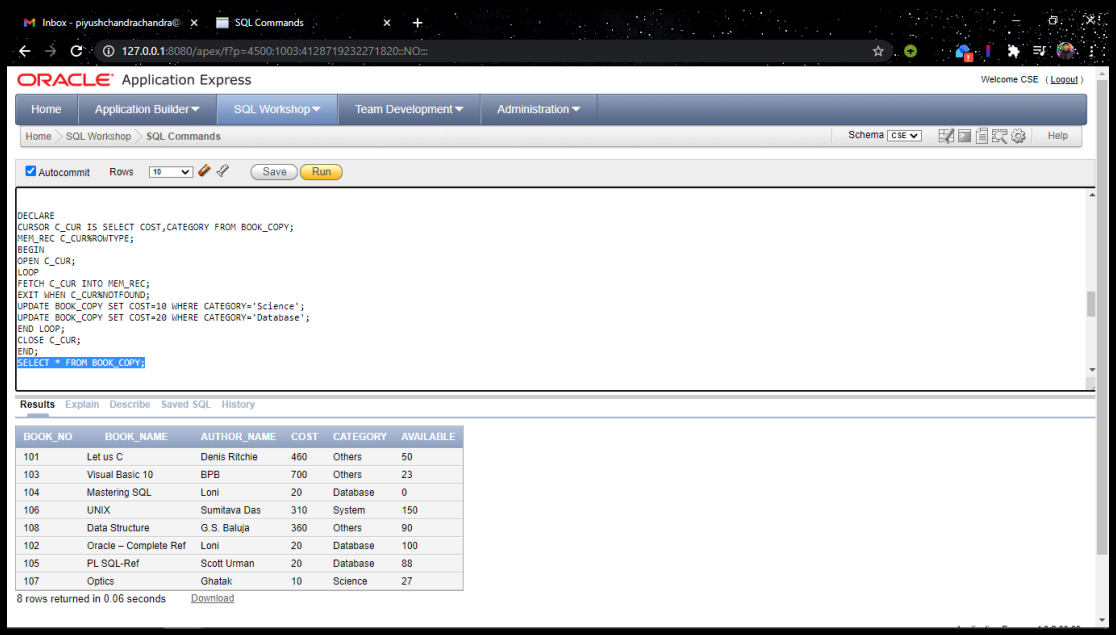
4. Write a PL/SQL program using Explicit Cursor which deducts the value of Max_Books_Allowed from MEMBER_COPY table. Deduct value means the value that how many times this member accesses the books. After deduction if value of Max_Books_Allowed is less than 0 the do not update it and show an error message.

➢ DECLARE
CURSOR C1 IS SELECT MEMBER_ID,MAX_BOOKS_ALLOWED FROM MEMBER_COPY;
M_ID MEMBER_COPY.MEMBER_ID%TYPE;
M_BOOKS MEMBER_COPY.MAX_BOOKS_ALLOWED%TYPE;
C NUMBER(3);
BEGIN
OPEN C1;
LOOP
FETCH C1 INTO M_ID,M_BOOKS;
EXIT WHEN C1%NOTFOUND;
SELECT COUNT(*)INTO C FROM ISSUE WHERE MEMBER_ID=M_ID;
IF(C<=M_BOOKS)THEN UPDATE MEMBER_COPY SET MAX_BOOKS_ALLOWED=MAX_BOOKS_ALLOWED-C WHERE MEMBER_ID=M_ID;
END IF;
END LOOP;
CLOSE C1;
END;
SELECT *FROM MEMBER_COPY;



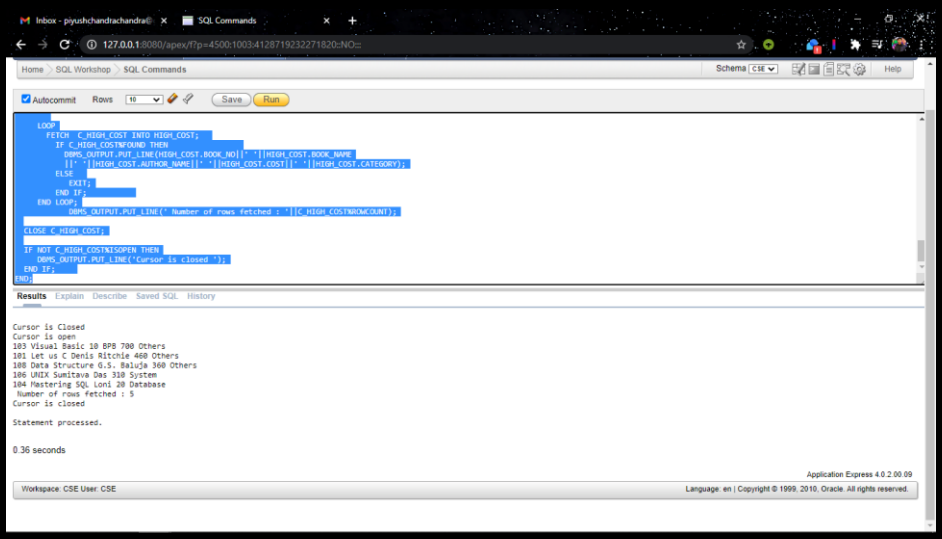
5. Create a table BOOK_UPDATE with attribute BOOK_NO, BOOK_NAME, INCREAMENT VALUE, UPDATE_DATE and write a PL/SQL program using Explicit Cursor which can update the cost value of BOOKS_COPY table with 10 and 20 where category is Science and database respectively, and if update is possible then insert BOOK_NO, BOOK_NAME, INCREAMENT VALUE, SYSDATE to the BOOK_UPDATE table.

➢ DECLARE
CURSOR C_CUR IS SELECT COST,CATEGORY FROM BOOK_COPY;
MEM_REC C_CUR%ROWTYPE;
BEGIN
OPEN C_CUR;
LOOP
FETCH C_CUR INTO MEM_REC;
EXIT WHEN C_CUR%NOTFOUND;
UPDATE BOOK_COPY SET COST=10 WHERE CATEGORY='Science';
UPDATE BOOK_COPY SET COST=20 WHERE CATEGORY='Database';
END LOOP;
CLOSE C_CUR;
END;



6. Write a PL/SQL program using Explicit Cursor which can display the all information of 5 books from BOOK_COPY table according to the higher cost.

```
➢ DECLARE
  CURSOR C_HIGH_COST IS
  SELECT * FROM(SELECT BOOK_NO,BOOK_NAME,AUTHOR_NAME,COST,CATEGORY
  FROM BOOK_COPY ORDER BY COST DESC)
  WHERE ROWNUM<6;
  HIGH_COST C_HIGH_COST%ROWTYPE;
  BEGIN
    IF NOT C_HIGH_COST%ISOPEN THEN
      DBMS_OUTPUT.PUT_LINE('Cursor is Closed');
    END IF;

   OPEN C_HIGH_COST;
     IF C_HIGH_COST%ISOPEN THEN
        DBMS_OUTPUT.PUT_LINE('Cursor is open');
     END IF;

     LOOP
       FETCH  C_HIGH_COST INTO HIGH_COST;
         IF C_HIGH_COST%FOUND THEN
           DBMS_OUTPUT.PUT_LINE(HIGH_COST.BOOK_NO||' '||HIGH_COST.BOOK_NAME
           ||' '||HIGH_COST.AUTHOR_NAME||' '||HIGH_COST.COST||' '||HIGH_COST.CATEGORY);
         ELSE
            EXIT;
         END IF;
     END LOOP;
            DBMS_OUTPUT.PUT_LINE(' Number of rows fetched : '||C_HIGH_COST%ROWCOUNT);

   CLOSE C_HIGH_COST;

   IF NOT C_HIGH_COST%ISOPEN THEN
     DBMS_OUTPUT.PUT_LINE('Cursor is closed ');
   END IF;
  END;
```

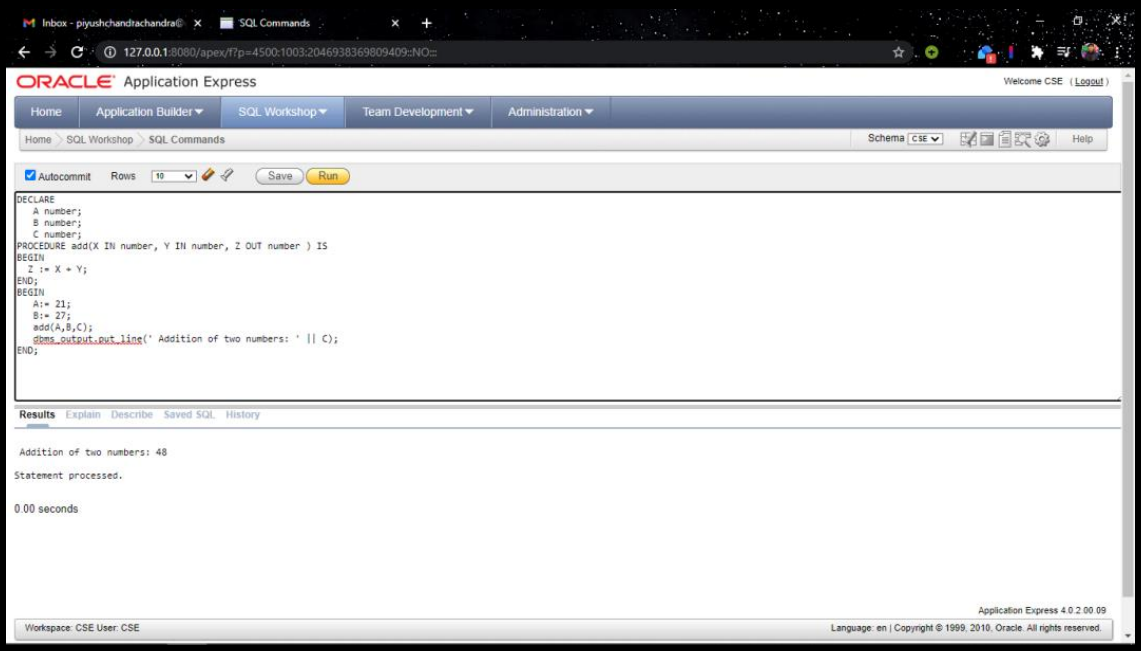


ASSIGNMENT 10

01/01/2021 – 10/01/2021

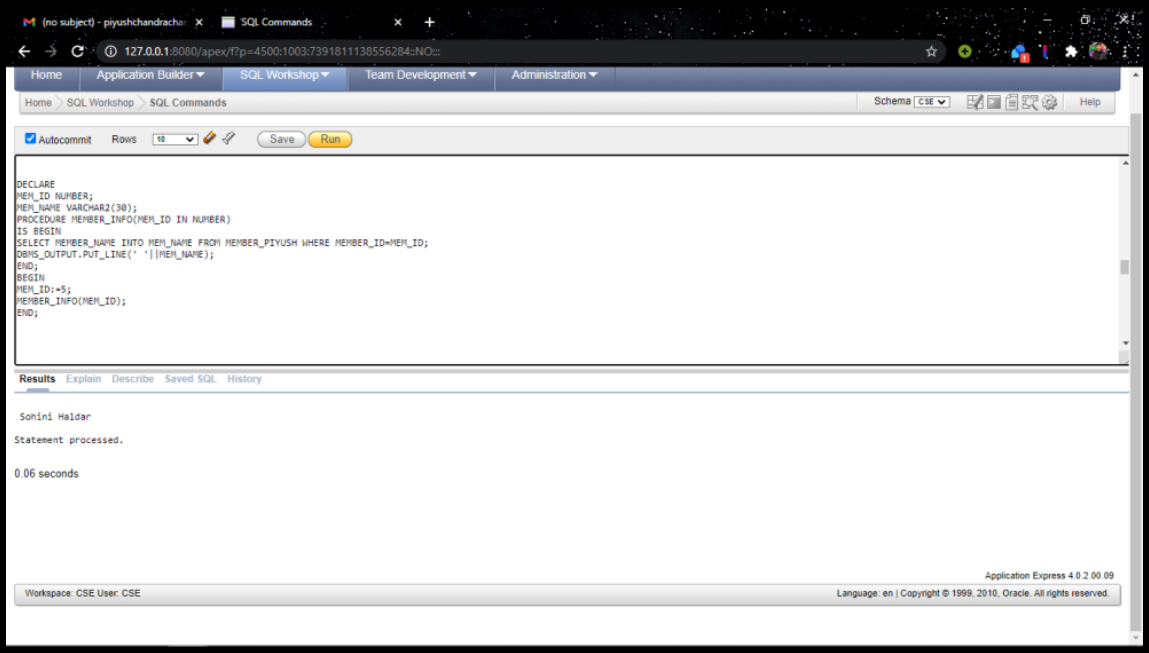
1) Write a PL/SQL PROCEDURE to compute addition between two numbers and use the different modes (IN, OUT, IN OUT) of data type in input variables which are used on creating PROCEDURE.

```
➢ DECLARE
    A number;
    B number;
    C number;
  PROCEDURE add(X IN number, Y IN number, Z OUT number ) IS
  BEGIN
    Z := X + Y;
  END;
  BEGIN
    A:= 21;
    B:= 27;
    add(A,B,C);
    dbms_output.put_line(' Addition of two numbers: ' || C);
  END;
```



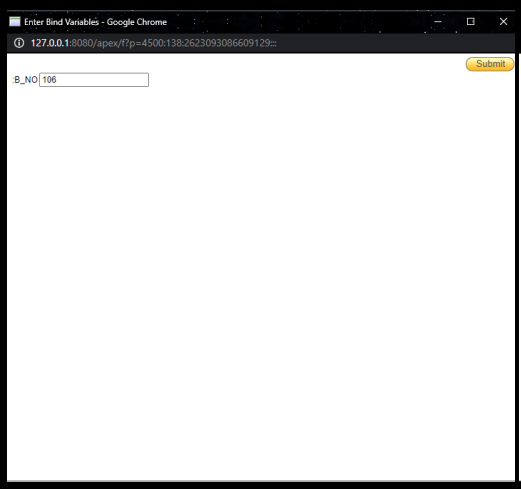
2) Write a PL/SQL Procedure which take MEMBER_ID as an input and show the corresponding MEMBER_NAME.

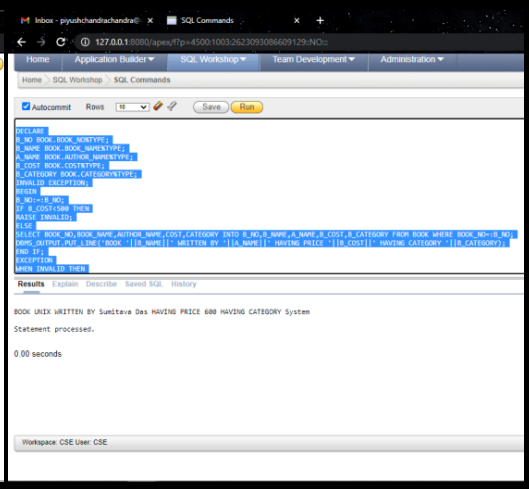
➢
```
DECLARE
MEM_ID NUMBER;
MEM_NAME VARCHAR2(30);
PROCEDURE MEMBER_INFO(MEM_ID IN NUMBER)
IS BEGIN
SELECT MEMBER_NAME INTO MEM_NAME FROM MEMBER_PIYUSH
WHERE MEMBER_ID=MEM_ID;
DBMS_OUTPUT.PUT_LINE(' '||MEM_NAME);
END;
BEGIN
MEM_ID:=5;
MEMBER_INFO(MEM_ID);
END;
```



3) Write a PL/SQL procedure which displays the details of books having cost greater than 500.

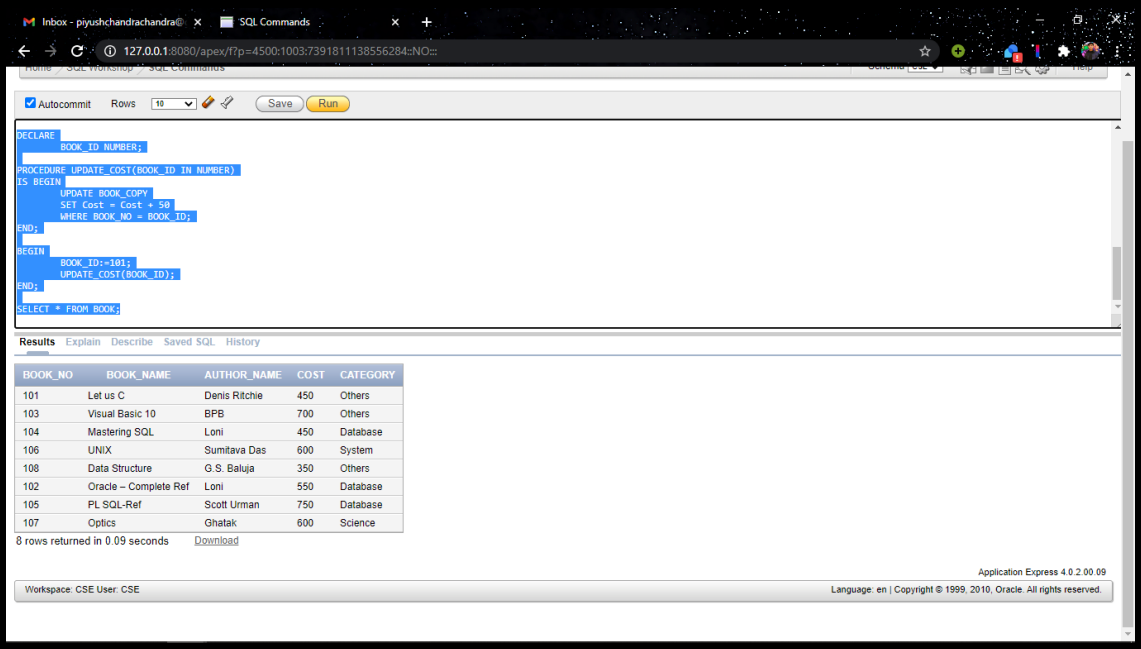
- DECLARE
  B_NO BOOK.BOOK_NO%TYPE;
  B_NAME BOOK.BOOK_NAME%TYPE;
  A_NAME BOOK.AUTHOR_NAME%TYPE;
  B_COST BOOK.COST%TYPE;
  B_CATEGORY BOOK.CATEGORY%TYPE;
  INVALID EXCEPTION;
  BEGIN
  B_NO:=:B_NO;
  IF B_COST<500 THEN
  RAISE INVALID;
  ELSE
  SELECT BOOK_NO,BOOK_NAME,AUTHOR_NAME,COST,CATEGORY
  INTO B_NO,B_NAME,A_NAME,B_COST,B_CATEGORY FROM BOOK
  WHERE BOOK_NO=:B_NO;
  DBMS_OUTPUT.PUT_LINE('BOOK '||B_NAME||' WRITTEN BY '||A_NAME||'
  HAVING PRICE '||B_COST||' HAVING CATEGORY '||B_CATEGORY);
  END IF;
  EXCEPTION
  WHEN INVALID THEN
  DBMS_OUTPUT.PUT_LINE('COST SHOULD BE GREATER THAN 500');
  WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('NO SUCH BOOKS!!');
  END;





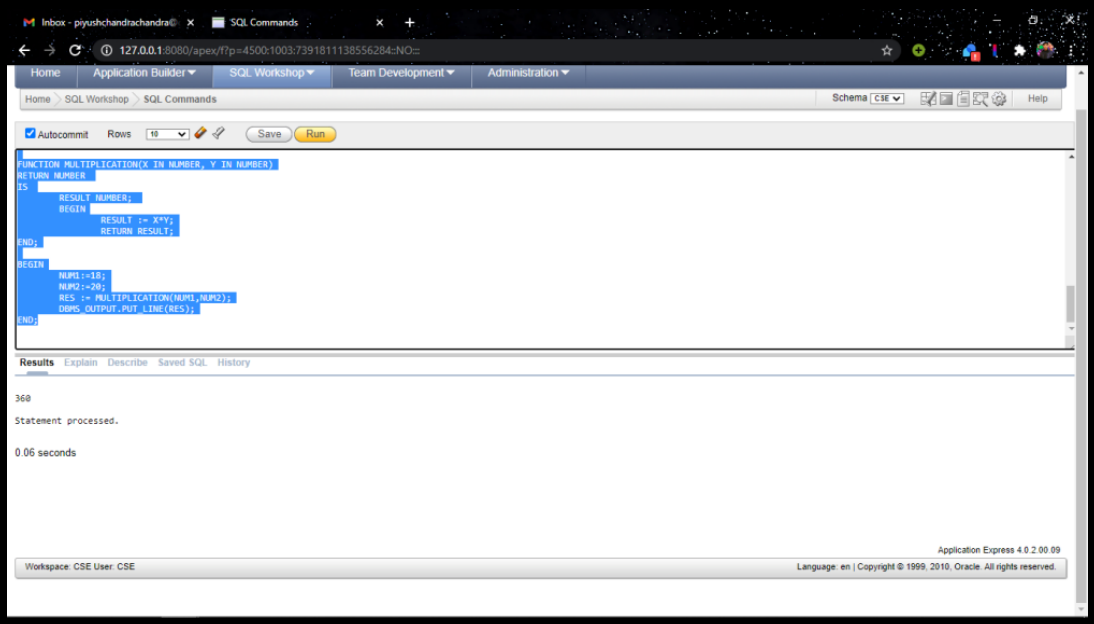
4) Write a PL/SQL procedure which can take the BOOK_ID as user input and update the cost with 50 more cost in BOOKS_COPY table.

```
DECLARE
BOOK_ID NUMBER;

PROCEDURE UPDATE_COST(BOOK_ID IN NUMBER)
IS BEGIN
        UPDATE BOOK_COPY
        SET Cost = Cost + 50
        WHERE BOOK_NO = BOOK_ID;
END;

BEGIN
        BOOK_ID:=101;
        UPDATE_COST(BOOK_ID);
END;

SELECT * FROM BOOK;
```



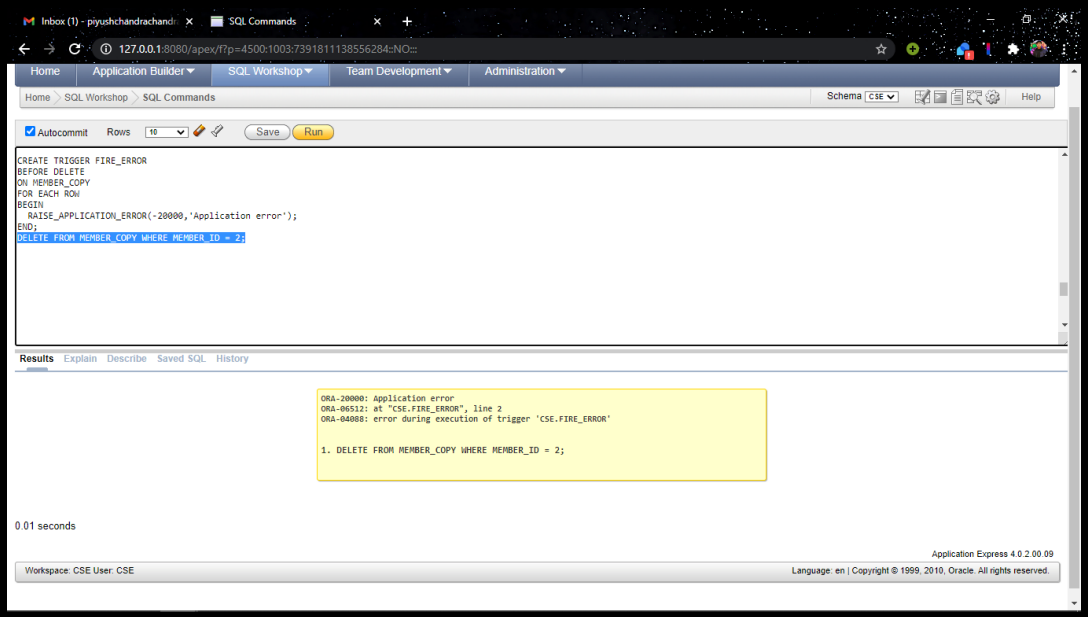
5) Write the PL/SQL function which can compute the multiplication between two numbers and return the value to the normal PL/SQL program which can show the output.

```
DECLARE
NUM1 NUMBER;
NUM2 NUMBER;
RES NUMBER;

FUNCTION MULTIPLICATION(X IN NUMBER, Y IN NUMBER)
RETURN NUMBER
IS
        RESULT NUMBER;
        BEGIN
                RESULT := X*Y;
                RETURN RESULT;
END;

BEGIN
        NUM1:=18;
        NUM2:=20;
        RES := MULTIPLICATION(NUM1,NUM2);
        DBMS_OUTPUT.PUT_LINE(RES);
END;
```



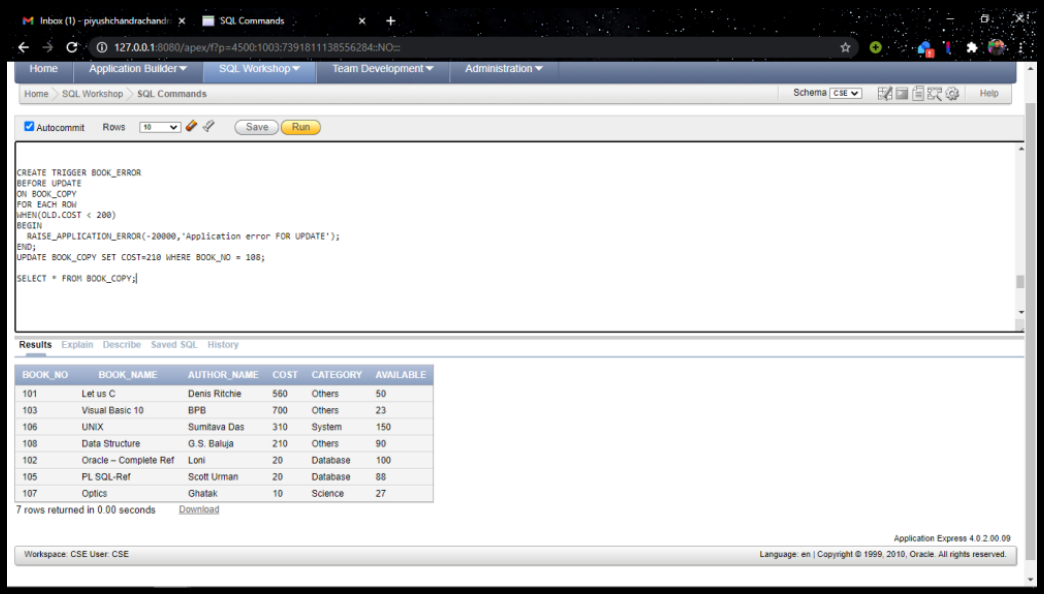
6) Create a PL/SQL Trigger which can fire on MEMBER_COPY table and raise an application error when delete a row from this table.

```
➢ CREATE TRIGGER FIRE_ERROR
  BEFORE DELETE
  ON MEMBER_COPY
  FOR EACH ROW
  BEGIN
    RAISE_APPLICATION_ERROR(-20000,'Application error');
  END;
  DELETE FROM MEMBER_COPY WHERE MEMBER_ID = 2;
```



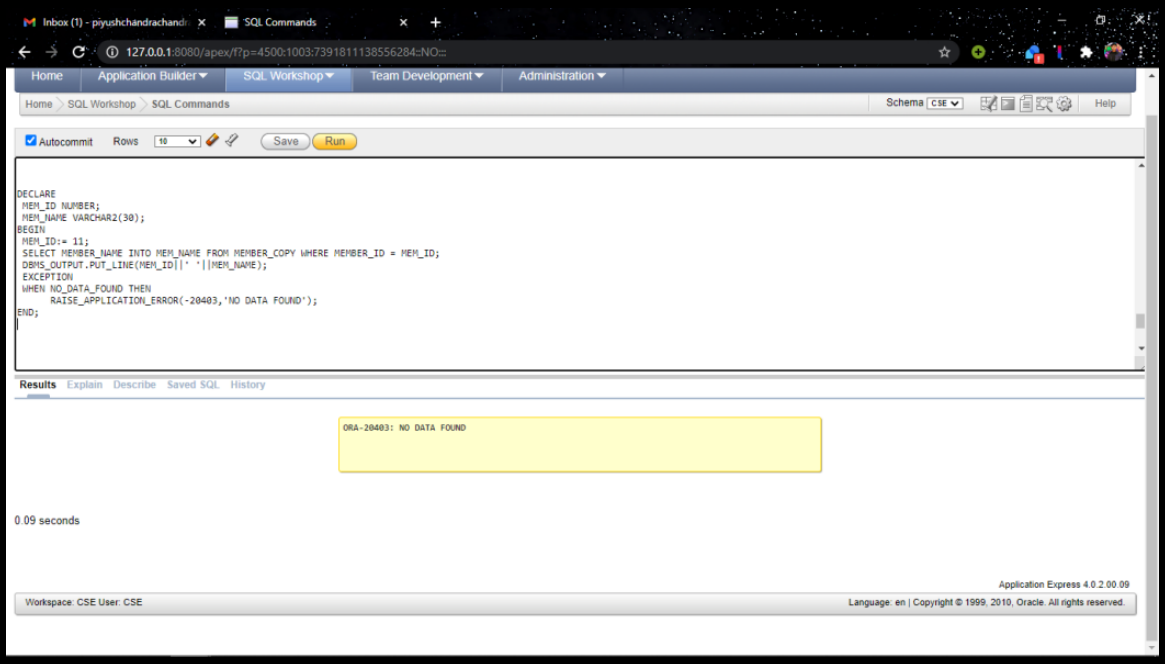
7) Create a PL/SQL Trigger which can fire on BOOK_COPY table and raise an application error when you are going to update cost less than 200 against any book.

```
CREATE TRIGGER BOOK_ERROR
BEFORE UPDATE
ON BOOK_COPY
FOR EACH ROW
WHEN(OLD.COST < 200)
BEGIN
  RAISE_APPLICATION_ERROR(-20000,'Application error FOR
UPDATE');
END;
UPDATE BOOK_COPY SET COST=210 WHERE BOOK_NO = 108;

SELECT * FROM BOOK_COPY;
```



8) Write a PL/SQL program which a show the Member_Name against a Member_ID and also raise an application error is no data found. (Using NO_DATA_FOUND Exception)

```
➢ DECLARE
   MEM_ID NUMBER;
   MEM_NAME VARCHAR2(30);
  BEGIN
   MEM_ID:= 11;
   SELECT MEMBER_NAME INTO MEM_NAME FROM MEMBER_COPY
  WHERE MEMBER_ID = MEM_ID;
   DBMS_OUTPUT.PUT_LINE(MEM_ID||' '||MEM_NAME);
   EXCEPTION
   WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20403,'NO DATA FOUND');
  END;
```



9) Create a table named CHANGE with three attribute Book_ID number(5), Change_Date date, Change_type varchar(10); Then Write a PL/SQL Trigger which can fire on BOOK_COPY table when you update or delete any particular Book and automatically insert a row into CHANGE table with the value of changed Book_NO, sysdate and Type of change (Update/Delete).

- CREATE TABLE CHANGE(
  BOOK_ID NUMBER(5),
  CHANGE_DATE DATE,
  CHANGE_TYPE VARCHAR(10));
  CREATE TRIGGER update_changes
  BEFORE UPDATE OR DELETE
  ON BOOK_COPY
  FOR EACH ROW
  BEGIN
  IF UPDATING THEN
  INSERT INTO CHANGE VALUES
  (:OLD.BOOK_NO,SYSDATE,'UPDATE');
  ELSE
  INSERT INTO CHANGE VALUES
  (:OLD.BOOK_NO,SYSDATE,'DELETE');
  END IF;
  END;

  UPDATE BOOK_COPY SET COST = 360 WHERE BOOK_NO = 104;

  SELECT * FROM CHANGE;

  DELETE BOOK_COPY WHERE BOOK_NO = 104;