<u>**Sheet 5**</u>

NAME: ARNAB SEN
EN.NO: 510519006 (Gx)
SUBJECT: DBMS Lab

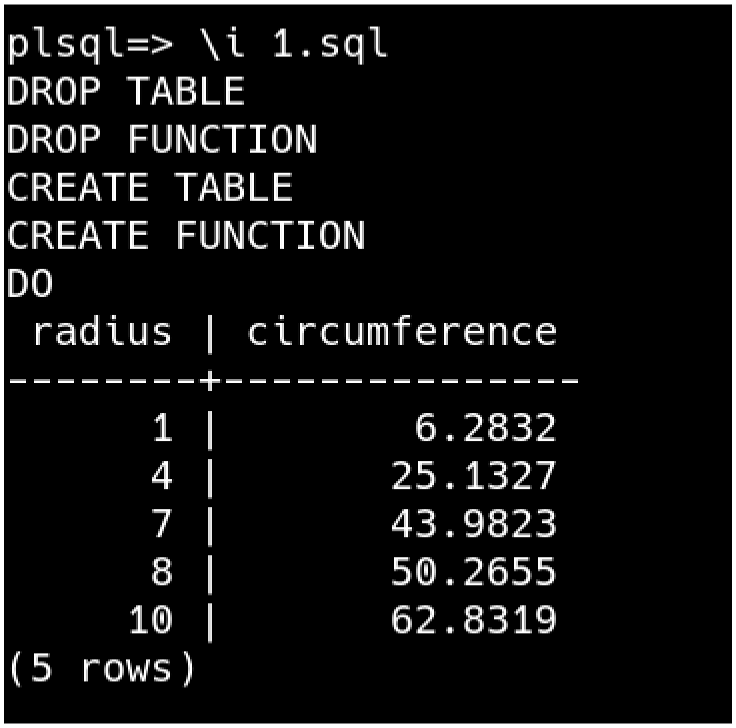
Assignment No: 10

Insert data into a table containing two attributes namely radius & circumference of circles. You may get different values of radius either from the keyboard or you may generate different values.

query:

```
DROP TABLE IF EXISTS circle;

DROP FUNCTION IF EXISTS insertIntoCircle;

CREATE TABLE circle (
    radius NUMERIC PRIMARY KEY,
    circumference NUMERIC(10, 4)
);

CREATE FUNCTION insertIntoCircle(radius int) RETURNS real AS $$
DECLARE
    circumference DOUBLE PRECISION := 2 * pi() * radius;
BEGIN
    INSERT INTO circle VALUES (radius, circumference);
    RETURN circumference;
END;
$$ LANGUAGE plpgsql;
```

```
DO $$
    BEGIN
        perform insertIntoCircle(1);
        perform insertIntoCircle(4);
        perform insertIntoCircle(7);
        perform insertIntoCircle(8);
        perform insertIntoCircle(10);
    END;
$$;

SELECT * FROM circle;
```

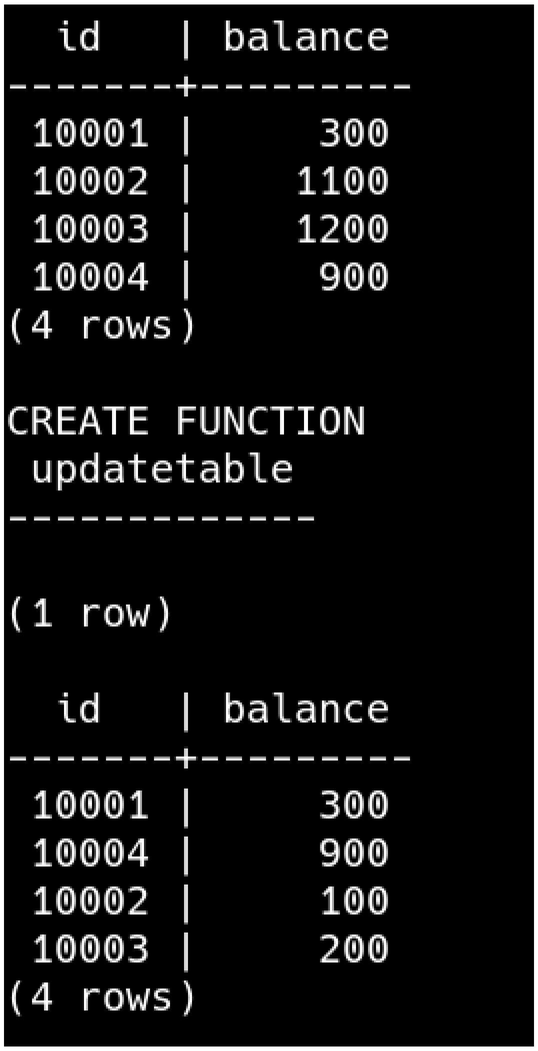


```
plsql=> \i 1.sql
DROP TABLE
DROP FUNCTION
CREATE TABLE
CREATE FUNCTION
DO
 radius | circumference
--------+---------------
      1 |        6.2832
      4 |       25.1327
      7 |       43.9823
      8 |       50.2655
     10 |       62.8319
(5 rows)
```

Update the balance of each customer from a cust_acct table showing withdrawal of Rs.1000/- as service charge provided that the customer balance shows at least Rs.1000/-.

query:

```
drop table if exists cust_acct;
drop function if exists updateTable;

create table cust_acct(
    id numeric(10) primary key,
    balance numeric(6) default 0
);

insert into cust_acct values (10001, 300);
insert into cust_acct values (10002, 1100);
insert into cust_acct values (10003, 1200);
insert into cust_acct values (10004, 900);

select * from cust_acct;

create function updateTable() returns void as $$
begin
    update cust_acct
    set balance = balance - 1000
    where balance >= 1000;
end;
$$ language plpgsql;

select updateTable();
select * from cust_acct;
```

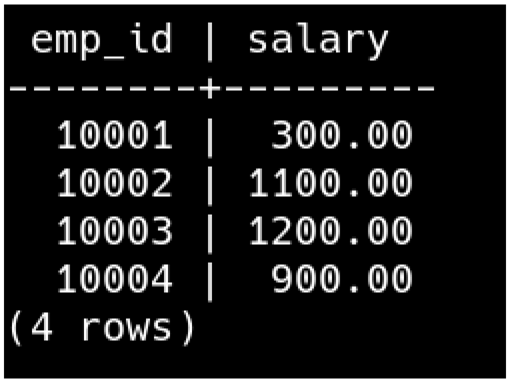


```
  id   | balance
-------+---------
 10001 |     300
 10002 |    1100
 10003 |    1200
 10004 |     900
(4 rows)

CREATE FUNCTION
 updatetable
------------

(1 row)

  id   | balance
-------+---------
 10001 |     300
 10004 |     900
 10002 |     100
 10003 |     200
(4 rows)
```

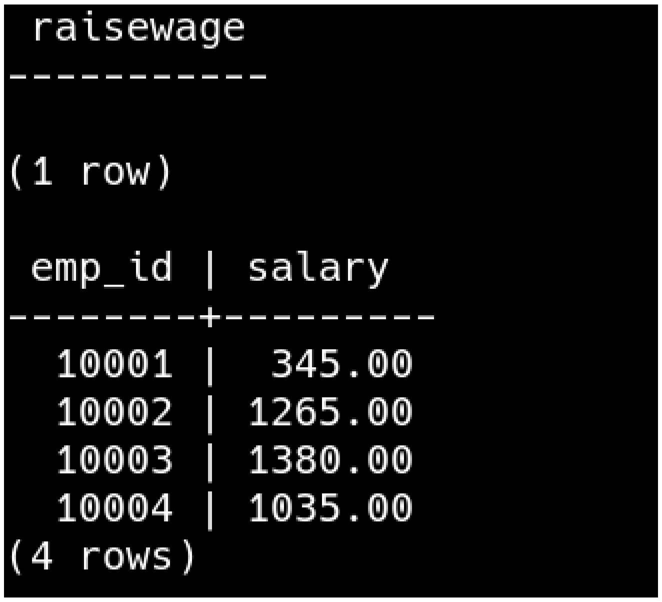
Update the salary of each employee from the EMP table by 15% using the cursor.

query:

```
drop table if exists emp;
drop function if exists raiseWage;

create table emp(
    emp_id numeric(10) primary key,
    salary numeric(10, 2)
);

insert into emp values (10001, 300);
insert into emp values (10002, 1100);
insert into emp values (10003, 1200);
insert into emp values (10004, 900);

select * from emp order by emp;

create function raiseWage() returns void as $$
    declare
        emp_rec record;
        emp_cursor cursor for select * from emp;
    begin
        open emp_cursor;

        loop
            fetch emp_cursor into emp_rec;
            exit when not found;

            update emp
            set salary = salary + salary * 15 /100
            where emp_id = emp_rec.emp_id;
        end loop;

        close emp_cursor;
    end;
-- $$ language plpgsql;

select raiseWage();

select * from emp order by emp;
```



```
 emp_id | salary
--------+---------
  10001 |  300.00
  10002 | 1100.00
  10003 | 1200.00
  10004 |  900.00
(4 rows)
```



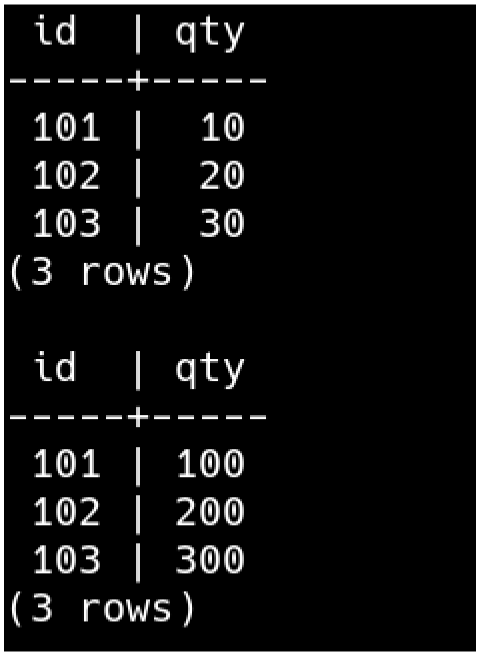
```
 raisewage
-----------

(1 row)

 emp_id | salary
--------+---------
  10001 |  345.00
  10002 | 1265.00
  10003 | 1380.00
  10004 | 1035.00
(4 rows)
```

Update the balance in the ITEM_MSTR table each time a transaction takes place in the ITEM_TR table. If this item_id is already present in the ITEM_MSTR table an update is performed to decrease the balance by the quantity specified in the ITEM_TR table. If the item_id is not present in the ITEM_MSTR table, the tuple is to be inserted.

query:

```
drop table if exists item_mstr;
drop table if exists item_tr;
drop function if exists item_trans_fn;

create table item_tr(
    id numeric(10),
    qty numeric(5)
);

create table item_mstr(
    id numeric(10) primary key,
    qty numeric(5)
);

insert into item_tr values (101, 10);
insert into item_tr values (102, 20);
insert into item_tr values (103, 30);

insert into item_mstr values (101, 100);
insert into item_mstr values (102, 200);
insert into item_mstr values (103, 300);

select * from item_tr;
select * from item_mstr;

create or replace function item_trans_fn() returns trigger language plpgsql
as $$
begin
    update item_mstr set qty = qty - new.qty where id = new.id;
    if not found then
        insert into item_mstr values(new.id, 500-new.qty);
    end if;

    return null;
end;
$$;
```

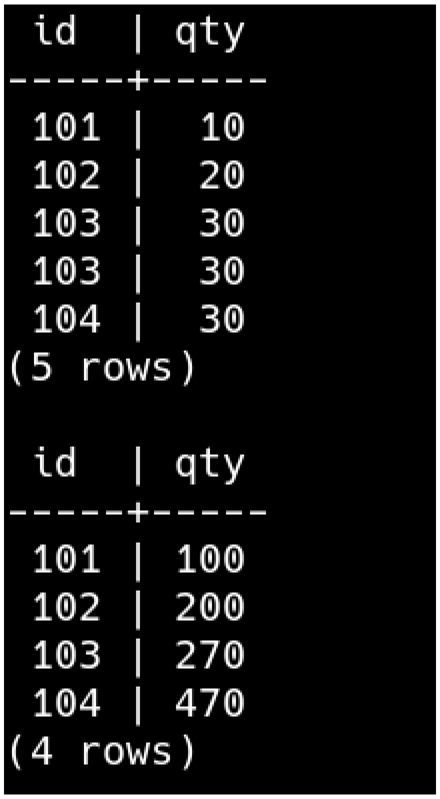
```
create trigger item_trans_trigger
after insert on item_tr
for each row
execute function item_trans_fn();

insert into item_tr values (103, 30);
insert into item_tr values (104, 30);

select * from item_tr order by id;
select * from item_mstr order by id;
```



```
 id  | qty
-----+-----
 101 |  10
 102 |  20
 103 |  30
(3 rows)

 id  | qty
-----+-----
 101 | 100
 102 | 200
 103 | 300
(3 rows)
```

After

```
insert into item_tr values (103, 30);
insert into item_tr values (104, 30);
```



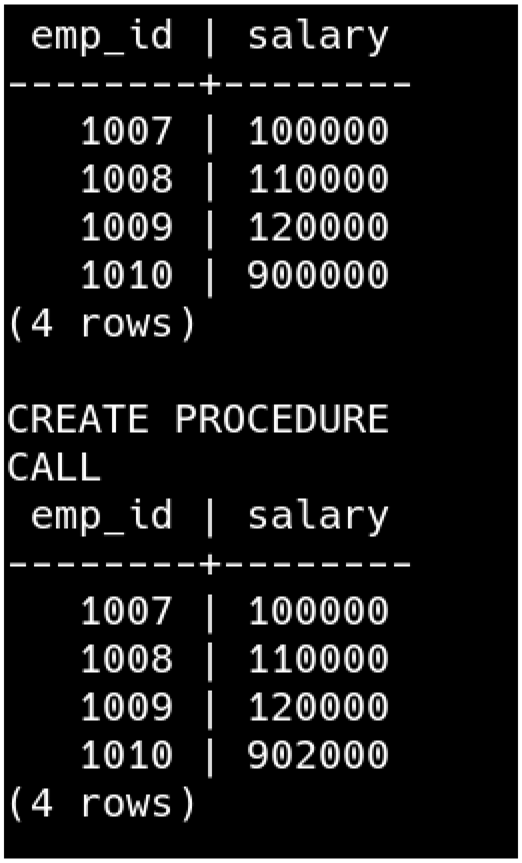
```
 id  | qty
-----+-----
 101 |  10
 102 |  20
 103 |  30
 103 |  30
 104 |  30
(5 rows)

 id  | qty
-----+-----
 101 | 100
 102 | 200
 103 | 270
 104 | 470
(4 rows)
```

Write a PROCEDURE for raising the salary of some employees by some amount. The PROCEDURE to be written may carry two parameters emp_id and amt to be raised. Include two exceptions that will be raised when either emp_id is not present or salary is NULL.

query:

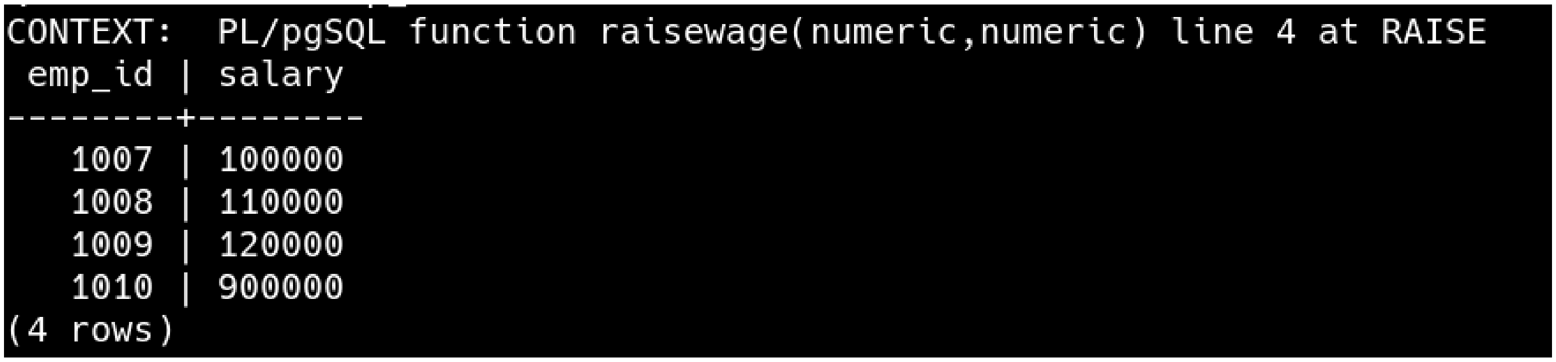
```
drop table if exists emp;
drop procedure if exists raiseWage(id numeric(10), amount numeric(10));

create table emp(
    emp_id numeric(10) primary key,
    salary numeric(10)
);
```

```
insert into emp values (1007, 100000);
insert into emp values (1008, 110000);
insert into emp values (1009, 120000);
insert into emp values (1010, 900000);

select * from emp order by emp_id;

create procedure raiseWage(id numeric(10), amount numeric(10))
    language plpgsql as $$
        begin
            if(id not in (select E.emp_id from emp E)) then
                raise exception 'nonexistent ID --> %', emp_id
                    using hint= 'Please check your emp_id';
            elsif (amount is null) then
              raise exception 'amount can''t be null'
                    using hint= 'Please check your amount';
            end if;

            update emp
               set salary=salary + amount
             where emp_id = id;
        end;
    $$;

call raiseWage(510719010,2000);

select * from emp order by emp_id;
```



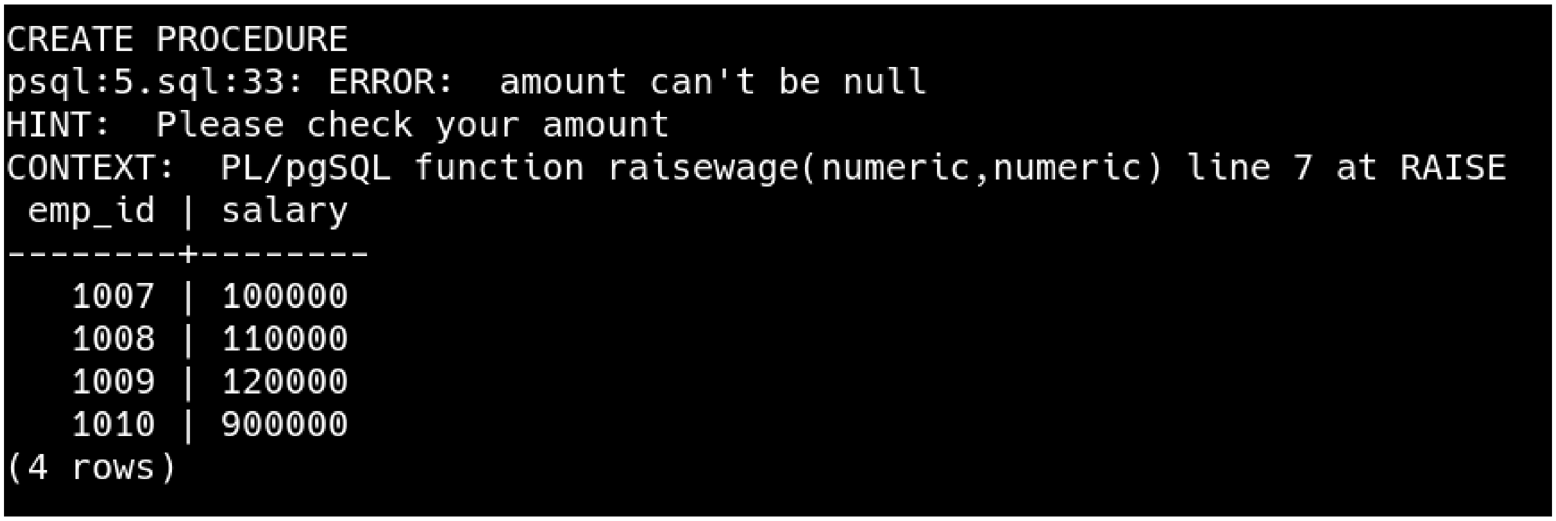
```
 emp_id | salary
--------+--------
   1007 | 100000
   1008 | 110000
   1009 | 120000
   1010 | 900000
(4 rows)

CREATE PROCEDURE
CALL
 emp_id | salary
--------+--------
   1007 | 100000
   1008 | 110000
   1009 | 120000
   1010 | 902000
(4 rows)
```

If the id doesn't exist then an exception is raised



```
CONTEXT:  PL/pgSQL function raisewage(numeric,numeric) line 4 at RAISE
 emp_id | salary
--------+--------
   1007 | 100000
   1008 | 110000
   1009 | 120000
   1010 | 900000
(4 rows)
```

Similarly, if the salary is NULL another exception is raised



```
CREATE PROCEDURE
psql:5.sql:33: ERROR:  amount can't be null
HINT:  Please check your amount
CONTEXT:  PL/pgSQL function raisewage(numeric,numeric) line 7 at RAISE
 emp_id | salary
--------+--------
   1007 | 100000
   1008 | 110000
   1009 | 120000
   1010 | 900000
(4 rows)
```