KOKI YAMANAKA COMP 3610 : Database Systems
Class Activity #7

Due date : 14th march time : Midnight (30 marks)

For this activity use the tables workers and branch created for activity#1. You need to add 2 more tables to the schema. One is bonus table and another is pastManagers table (with 4 attributes – brancid, staffid, managerfrom and managerTo).

pastManagers will get data inserted when a new manager joins a branch. (Q1)

Now create and test the following procedures/functions.

Q1. Create a trigger that will run on update on the branch table when a manager is changed. Update branch set managerCode = ‘B1001’, managersince=sysdate; [5 marks]

On update, the trigger should run to:

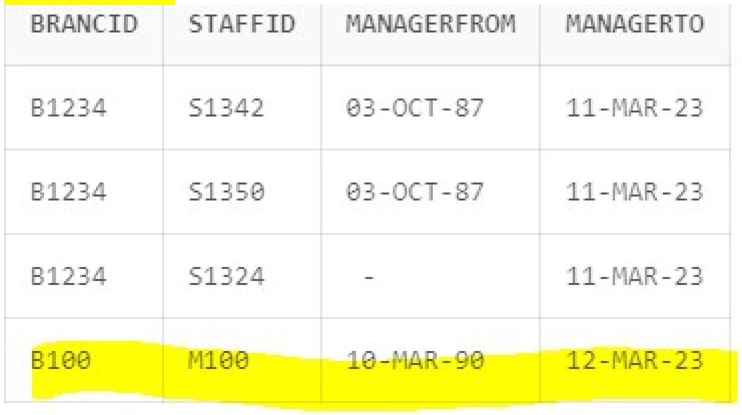
1. Insert the information on the pastManager table as managerCode, BranchCode, dateFrom, dateTo

CODE Q1

```
-- <QUESTION 1> - Q1. Create a trigger that will run on update on the branch table when a
manager is changed.
CREATE OR REPLACE TRIGGER update_branch_manager_changed
AFTER UPDATE OF managerCode ON branch001
FOR EACH ROW
BEGIN
  -- Insert the information on the pastManagers table as managerCode, BranchCode,
dateFrom, dateTo
  INSERT INTO pastManagers (brancid, staffid, managerfrom, managerTo)
  VALUES (:new.branchCode, :old.managerCode, :old.managersince, sysdate);
END;

-- <QUESTION 1: TEST CASE>
-- Insert a record into the branch001 table
INSERT INTO branch001 (branchCode, address, city, phone, yearOfOpening, managerCode,
managersince)
VALUES ('B100', '123 Main St', 'Seattle', '555-555-1212', 1990, 'M100', to_date('03-10-1990',
'MM-DD-YYYY'));

-- Update the managerCode in the branch001 table
UPDATE branch001
SET managerCode = 'M100', managersince = sysdate
WHERE branchCode = 'B100';

-- Check the pastManagers table to see if the information was inserted
```

```
SELECT * FROM pastManagers;
```

OUTPUT Q1 :



BRANCID	STAFFID	MANAGERFROM	MANAGERTO
B1234	S1342	03-OCT-87	11-MAR-23
B1234	S1350	03-OCT-87	11-MAR-23
B1234	S1324	-	11-MAR-23
B100	M100	10-MAR-90	12-MAR-23

To create a bonus table, run the command as:
Update staff set bonus = 2000, bonus_date=sysdate where staffId='S100';
Q2. Create a trigger to add bonus to bonus table, and every time a bonus is given to staff, it should be added to the bonus table.[5 marks]

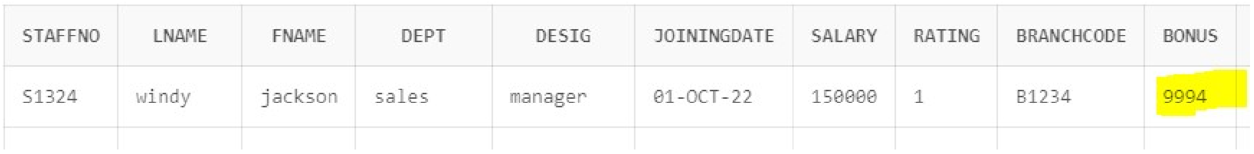
code for Q2

```
CREATE OR REPLACE TRIGGER add_bonus
AFTER UPDATE ON staff035 -- update bonus on staff035
FOR EACH ROW
BEGIN
  -- add bonus amount to bonus table
    INSERT INTO bonus (BONUS_ID, BONUS_AMOUNT, BONUS_ADDED_DATE,
BONUS_ADDED_TO_ID)
           VALUES (bonus_sequence.NEXTVAL, :NEW.BONUS, SYSDATE, :NEW.STAFFNO);
END;
```

TEST CASE

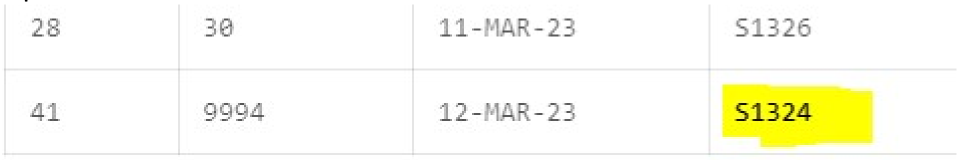
```
-- <QUESTION 2:  TEST CASE>
UPDATE staff035 SET BONUS = 9994 WHERE STAFFNO = 'S1324';
SELECT * FROM staff035;
SELECT * FROM bonus;
```

OUTPUT Q2
Update on staff table



STAFFNO	LNAME	FNAME	DEPT	DESIG	JOININGDATE	SALARY	RATING	BRANCHCODE	BONUS
S1324	windy	jackson	sales	manager	01-OCT-22	150000	1	B1234	9994

Update on staff bonus table



28	30	11-MAR-23	S1326
41	9994	12-MAR-23	S1324

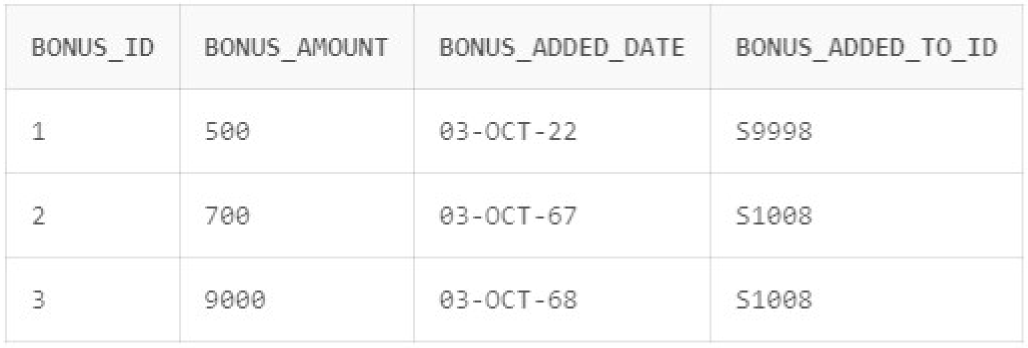
Q3. Write a function to pass worker_id and return the number of bonuses received by the employee. (Not the total amount, # only) [5 marks]

CODE Q3

```
CREATE OR REPLACE FUNCTION number_of_times_bonuses_received (
        p_staffId IN staff035.STAFFNO%TYPE
)
RETURN NUMBER
AS
   num_times_bonuses_received NUMBER;
BEGIN
        SELECT COUNT(BONUS_ID) INTO num_times_bonuses_received FROM bonus WHERE
BONUS_ADDED_TO_ID =  p_staffId ;
        RETURN num_times_bonuses_received;
END;
```

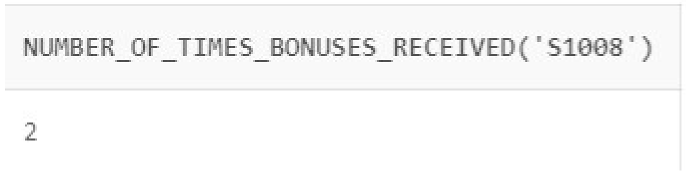
OUTPUT Q3

Actual bonus table



BONUS_ID	BONUS_AMOUNT	BONUS_ADDED_DATE	BONUS_ADDED_TO_ID
1	500	03-OCT-22	S9998
2	700	03-OCT-67	S1008
3	9000	03-OCT-68	S1008

Executed function



NUMBER_OF_TIMES_BONUSES_RECEIVED('S1008')
2

Q4. Write a procedure that will take the worker_id as the parameter. The procedure should first print the first_name, Last_name, department and date of joining.
Followed by this must be the list of all the bonuses the employee has obtained listed date wise.

At the end, it must list the total bonus received by the worker. Also Use the function defined in Q1 to display the total number of bonus transactions for the employee at the end. [6 marks]

Sample output:

```
Bonus Details of Amitabh Singh from Admin, since 14-02-20
          16-02-20.           4500
          5-03-21             5000
         ----------------------------------
          Total               9500
         ----------------------------------
Total transactions : ## (this should come from Q1 function)
```

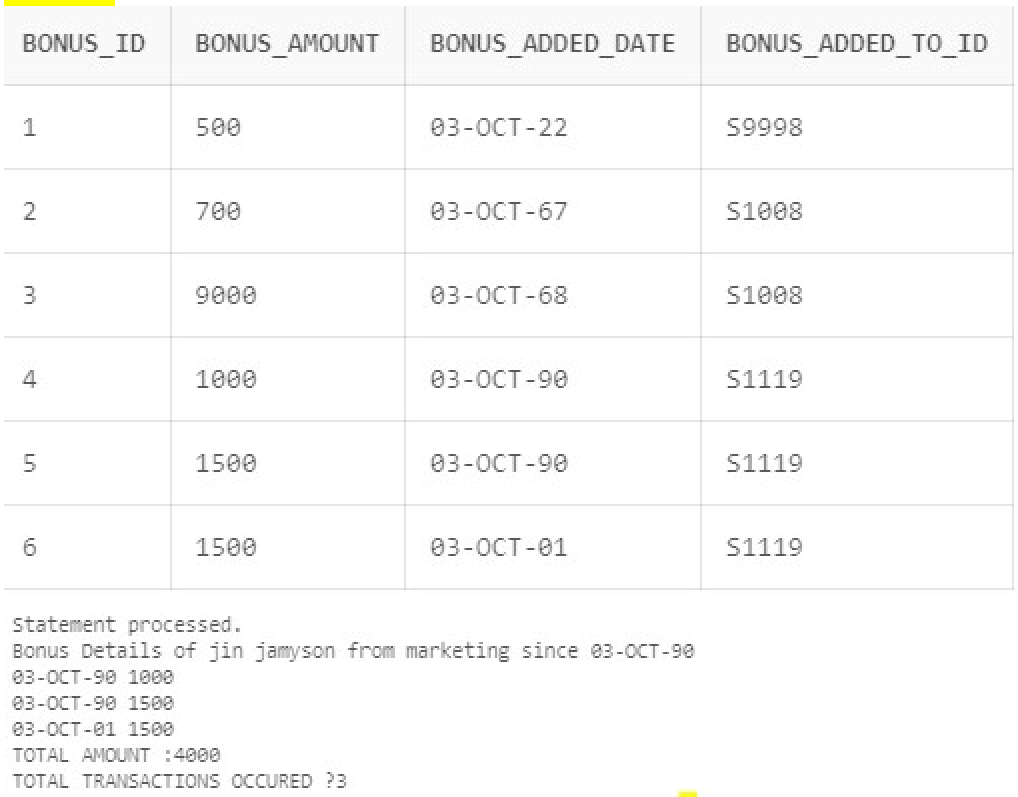
** please add a few more records in bonus table to have this output

CODE Q4

```
-- <QUESTION 4> - procedure to display bonuses received by an employee
CREATE OR REPLACE PROCEDURE display_bonus_details  (
          p_staffId IN staff035.STAFFNO%TYPE
) AS
   first_name staff035.fname%TYPE;
   last_name staff035.lname%TYPE;
   dept staff035.dept%TYPE;
   doj staff035.joiningDate%TYPE;
   bonus_amount bonus.bonus_amount%TYPE;
   bonus_date bonus.bonus_added_date%TYPE;
   total_bonus NUMBER;
BEGIN
    -- 1) The first name, last name, department, and date of joining of the employee.
          SELECT FNAME,LNAME,DEPT ,JOININGDATE INTO first_name,last_name,dept,doj FROM
staff035 WHERE STAFFNO = p_staffId ;
          DBMS_OUTPUT.PUT_LINE('Bonus Details of ' || first_name || ' ' || last_name || ' from '
|| dept || ' since ' || doj);

          -- 2) List of all bonuses the employee has received, listed date wise.
          DECLARE
   CURSOR c_bonus_details IS
   SELECT BONUS_ADDED_DATE, BONUS_AMOUNT FROM bonus WHERE
BONUS_ADDED_TO_ID = p_staffId ;
   BEGIN
   OPEN c_bonus_details;
   FETCH c_bonus_details INTO bonus_date, bonus_amount;
```

```
    WHILE c_bonus_details%FOUND
    LOOP
    DBMS_OUTPUT.PUT_LINE(bonus_date || ' ' || bonus_amount);
    FETCH c_bonus_details INTO bonus_date, bonus_amount;
    END LOOP;
    CLOSE c_bonus_details;
    END;

            -- 3) Total amount of bonus received
            SELECT SUM(bonus_amount) INTO total_bonus FROM bonus WHERE
bonus_added_to_ID = p_staffId;
            DBMS_OUTPUT.PUT_LINE('TOTAL AMOUNT :' || total_bonus);
            -- 4) The total number of bonus transactions
            DBMS_OUTPUT.PUT_LINE('TOTAL TRANSACTIONS OCCURED
?'||number_of_times_bonuses_received(p_staffId));

END;
```

OUTPUT Q4



BONUS_ID	BONUS_AMOUNT	BONUS_ADDED_DATE	BONUS_ADDED_TO_ID
1	500	03-OCT-22	S9998
2	700	03-OCT-67	S1008
3	9000	03-OCT-68	S1008
4	1000	03-OCT-90	S1119
5	1500	03-OCT-90	S1119
6	1500	03-OCT-01	S1119

```
Statement processed.
Bonus Details of jin jamyson from marketing since 03-OCT-90
03-OCT-90 1000
03-OCT-90 1500
03-OCT-01 1500
TOTAL AMOUNT :4000
TOTAL TRANSACTIONS OCCURED ?3
```

Q5. Write a function that should pass the worker_id and bonus amount. The procedure will check if the worker has got the bonus before. It should add the new bonus into bonus table and also print an appropriate message depending upon the past bonus amounts [4 marks]

Congratulations of your first bonus (if it's the first time)
OR
Wow! You got the biggest bonus till yet (if the past bonus were less than todays)
OR
Enjoy the bonus!!! (else)

Code for Q5:

```
CREATE SEQUENCE bonus_sequence
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE  check_has_bonus (
  -- worker id and bonus amount as parameter
  p_worker_id staff035.staffNo%TYPE,
  p_bonus  VARCHAR2
)
AS
   prev_bonus_amount NUMBER; -- store bonus amount
   new_bonus_id NUMBER; -- store bonus new bonus id for bonus transaction
BEGIN
    SELECT BONUS INTO prev_bonus_amount FROM staff035 WHERE staffNo = p_worker_id;
            SELECT bonus_sequence.NEXTVAL INTO new_bonus_id FROM dual; -- get a new bonus
id

            IF prev_bonus_amount IS NULL THEN -- first bonus
        INSERT INTO bonus (bonus_id , bonus_amount , bonus_added_date,bonus_added_to_ID  )
                        VALUES (new_bonus_id, TO_NUMBER(p_bonus), sysdate, p_worker_id);
                        DBMS_OUTPUT.PUT_LINE('Congratulations on your first bonus!');

    ELSIF prev_bonus_amount < TO_NUMBER(p_bonus) THEN
                        INSERT INTO bonus (bonus_id , bonus_amount ,
bonus_added_date,bonus_added_to_ID  )
                        VALUES (new_bonus_id, TO_NUMBER(p_bonus), sysdate, p_worker_id);
                        DBMS_OUTPUT.PUT_LINE('Wow! You got the biggest bonus till yet!');
            ELSE
                        INSERT INTO bonus (bonus_id , bonus_amount ,
bonus_added_date,bonus_added_to_ID  )
                        VALUES (new_bonus_id, TO_NUMBER(p_bonus), sysdate, p_worker_id);
                        DBMS_OUTPUT.PUT_LINE('Enjoy the bonus!!!');
            END IF;
END;
```

Other way of adding data to pastManager is using the following procedure:

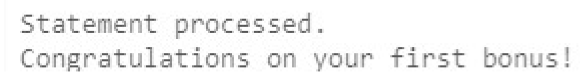
OUTPUT Q5

-- UPDATE sample data check the conditions defined in Q5 procedure

```
UPDATE staff035
SET BONUS = 30
WHERE STAFFNO = 'S1326';

UPDATE staff035
SET BONUS = 56
WHERE STAFFNO = 'S1326';
```

-- test case 1, 1st condition

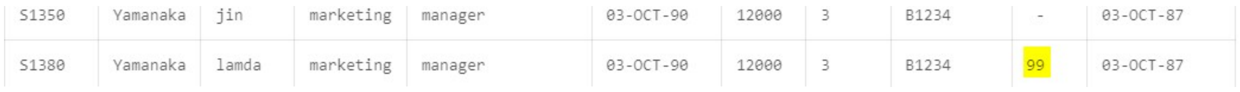
```
EXEC check_has_bonus('S1327',100);
```



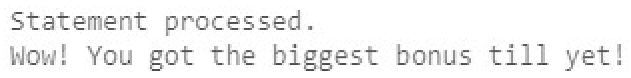
```
Statement processed.
Congratulations on your first bonus!
```

-- test case 1, 2nd condition

```
EXEC check_has_bonus('S1380',1000);
```



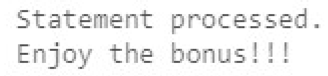
S1350	Yamanaka	jin	marketing	manager	03-OCT-90	12000	3	B1234	-	03-OCT-87
S1380	Yamanaka	lamda	marketing	manager	03-OCT-90	12000	3	B1234	99	03-OCT-87



```
Statement processed.
Wow! You got the biggest bonus till yet!
```

-- test case 2, 3rd condition check

```
EXEC check_has_bonus('S1326',30);
```



```
Statement processed.
Enjoy the bonus!!!
```

Q6. Write a procedure newBranchManager that passes staffid and branch id and do the following:

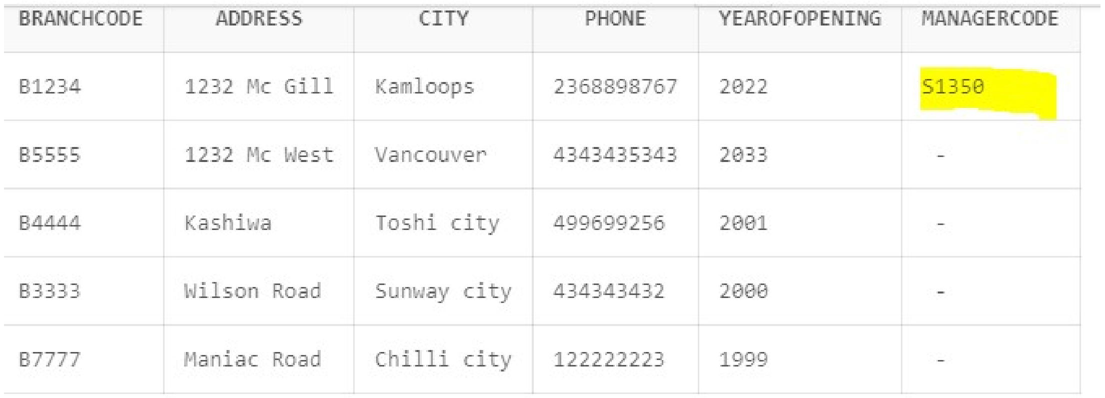
Code Q6:

```
-- <><> Q6 <><> ----  purpose : record past managers for each branch

CREATE OR REPLACE PROCEDURE newBranchManager (p_staffid IN VARCHAR2, p_branchid IN
VARCHAR2)
AS
        retrieved_branchCode varchar(5);
        retrieved_desig varchar(25);
        retrieved_working_manager_since DATE;
BEGIN
  -- Check if the staff works at the same branch as the p_branchid
   SELECT BRANCHCODE INTO retrieved_branchCode FROM staff035 WHERE staffNo =
p_staffid;
        IF retrieved_branchCode != p_branchid THEN
     RAISE_APPLICATION_ERROR(-20001, 'The staff does not work at the same branch.');
   END IF;

  -- Check if the staff's designation is not "manager"
   SELECT DESIG INTO retrieved_desig FROM staff035 WHERE staffNo = p_staffid;
        IF retrieved_desig = 'manager' THEN
     RAISE_APPLICATION_ERROR(-20002, 'The staff is already a manager.');
   END IF;

        -- change staff desig to manager
        UPDATE staff035 SET DESIG = 'manager' WHERE STAFFNO = p_staffid;


  -- Insert a record into the pastManagers table
   SELECT WORKING_SINCE INTO retrieved_working_manager_since FROM staff035 WHERE
staffNo = p_staffid;
   INSERT INTO pastManagers (brancid, staffid, managerfrom, managerTo)
        VALUES (p_branchid, p_staffid, retrieved_working_manager_since, SYSDATE);

  --  Update the branch001 table
   UPDATE branch001 SET managercode = p_staffid WHERE branchCode = p_branchid;


  COMMIT;
END;
```

OUTPUT Q6

a. Set the managercode of the branch to staffId.

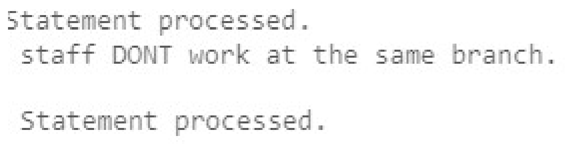
-- TEST CASE 3 : set manager code on branch table
EXEC newBranchManager('S1350','B1234');



BRANCHCODE	ADDRESS	CITY	PHONE	YEAROFOPENING	MANAGERCODE
B1234	1232 Mc Gill	Kamloops	2368898767	2022	S1350
B5555	1232 Mc West	Vancouver	4343435343	2033	-
B4444	Kashiwa	Toshi city	499699256	2001	-
B3333	Wilson Road	Sunway city	434343432	2000	-
B7777	Maniac Road	Chilli city	122222223	1999	-

b. Make sure staff works at the same branch.

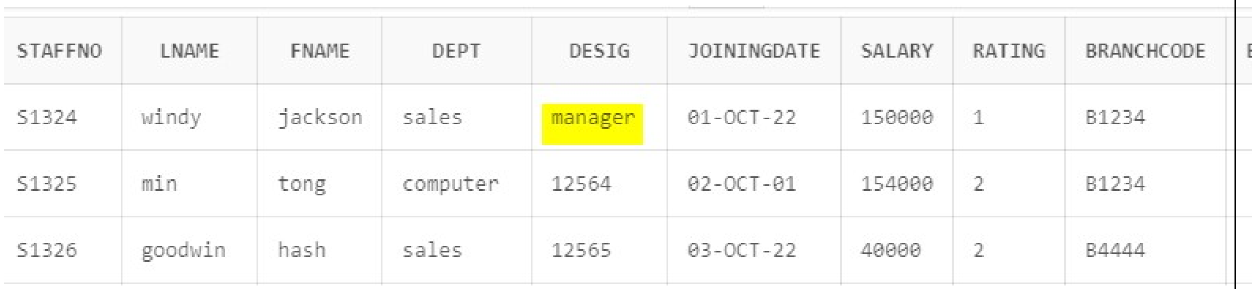
-- TEST CASE 1 : manager is not working at same branch
UPDATE staff035 -- alter a worker branch code
SET BRANCHCODE = 'B4444'
WHERE STAFFNO = 'S1326';
EXEC newBranchManager('S1326','B1234');
EXEC newBranchManager('S1326','B123');



```
Statement processed.
 staff DONT work at the same branch.

 Statement processed.
```

c. Make sure staff designation is changed to manager.

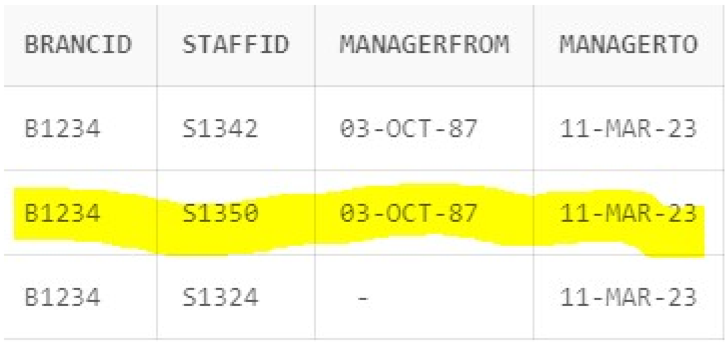
EXEC newBranchManager('S1324','B1234');



STAFFNO	LNAME	FNAME	DEPT	DESIG	JOININGDATE	SALARY	RATING	BRANCHCODE	E
S1324	windy	jackson	sales	manager	01-OCT-22	150000	1	B1234	
S1325	min	tong	computer	12564	02-OCT-01	154000	2	B1234	
S1326	goodwin	hash	sales	12565	03-OCT-22	40000	2	B4444	

d. Before updating branch table, make sure to insert a record in pastManagers table with the values branchCode, staffId, managerSince, sysdate

```
-- TEST CASE 3 : set manager code on branch table
EXEC newBranchManager('S1350','B1234');;
```



BRANCID	STAFFID	MANAGERFROM	MANAGERTO
B1234	S1342	03-OCT-87	11-MAR-23
B1234	S1350	03-OCT-87	11-MAR-23
B1234	S1324	-	11-MAR-23

with 4 attributes – brancid, staffid, managerfrom and managerTo. [5 marks]