BSCCS2001: Practice Assignment with Solutions
Week 3

1. Consider the relational schema given in Figure 1.

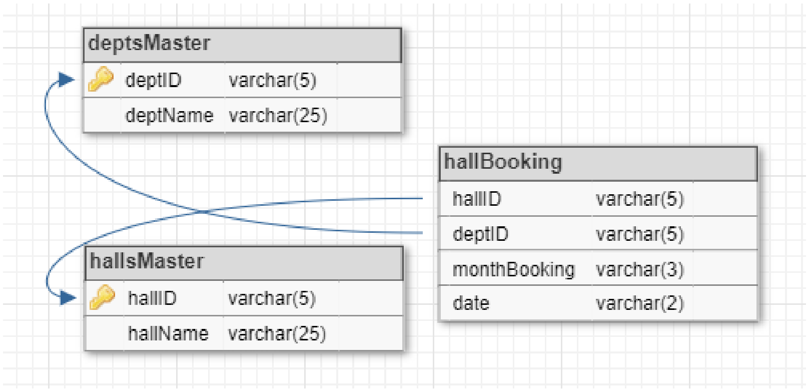


Figure 1: Hall Booking Relational Schema

Find the names of departments that have booked all the halls at least once in the month of January. [MCQ: 2 points]

○
```
SELECT deptName FROM deptsMaster
WHERE deptID IN
        (SELECT DISTINCT deptID FROM hallBooking AS hb1
        WHERE NOT EXISTS
                (SELECT hm.hallID FROM hallsMaster AS hm
                INTERSECT
                SELECT hb2.hallID FROM hallBooking AS hb2
                WHERE hb1.deptID = hb2.deptID
                AND monthBooking = 'Jan'));
```

✓
```
SELECT deptName FROM deptsMaster
WHERE deptID IN
        (SELECT DISTINCT deptID FROM hallBooking AS hb1
        WHERE NOT EXISTS
                (SELECT hm.hallID FROM hallsMaster AS hm
                EXCEPT
                SELECT hb2.hallID FROM hallBooking AS hb2
                WHERE hb1.deptID = hb2.deptID
                AND monthBooking = 'Jan'));
```

○
```
SELECT deptName FROM deptsMaster
WHERE deptID IN
        (SELECT DISTINCT deptID FROM hallBooking AS hb1
        WHERE EXISTS
```

```
                (SELECT hm.hallID FROM hallsMaster AS hm
                EXCEPT
                SELECT hb2.hallID FROM hallBooking AS hb2
                WHERE hb1.deptID = hb2.deptID
                AND monthBooking = 'Jan'));
```

◯
```
SELECT deptName FROM deptsMaster
WHERE deptID IN
        (SELECT DISTINCT deptID FROM hallBooking AS hb1
        WHERE EXISTS
                (SELECT hm.hallID FROM hallsMaster AS hm
                INTERSECT
                SELECT hb2.hallID FROM hallBooking AS hb2
                WHERE hb1.deptID = hb2.deptID
                AND monthBooking = 'Jan'));
```

Solution:

```
SELECT hm.hallID FROM hallsMaster AS hm
                EXCEPT
                SELECT hb2.hallID FROM hallBooking AS hb2
                WHERE hb1.deptID = hb2.deptID
                AND monthBooking = 'Jan'
```

The above query fetches all hallIDs that have not been booked in January.

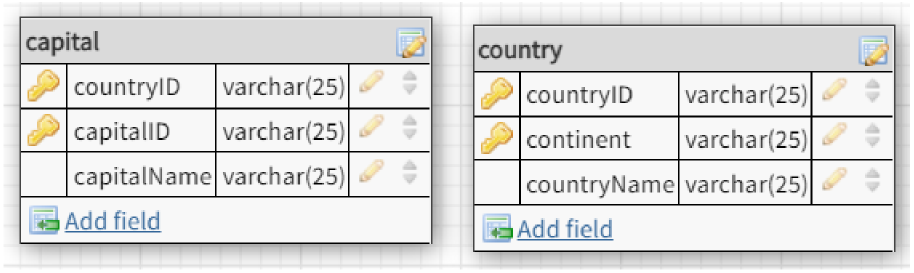
```
SELECT DISTINCT deptID FROM hallBooking AS hb1
        WHERE NOT EXISTS (hallIDs that have not been booked in January)
```

The above query will retrieve all department IDs that have not booked any halls that have not been booked in January.

```
SELECT deptName FROM deptsMaster
WHERE deptID IN (all department IDs that have not booked any halls
that have not been booked in January)
```

The above query fetches the names of departments that have not booked any halls that have not been booked in January.
Note that if we execute the nested queries directly on the sql prompt, they will give errors due to the aliases used.

2. Consider the relational schema given in Figure 2.



capital

🔑	countryID	varchar(25)
🔑	capitalID	varchar(25)
	capitalName	varchar(25)

country

🔑	countryID	varchar(25)
🔑	continent	varchar(25)
	countryName	varchar(25)

Figure 2: Country Capitals Relational Schema

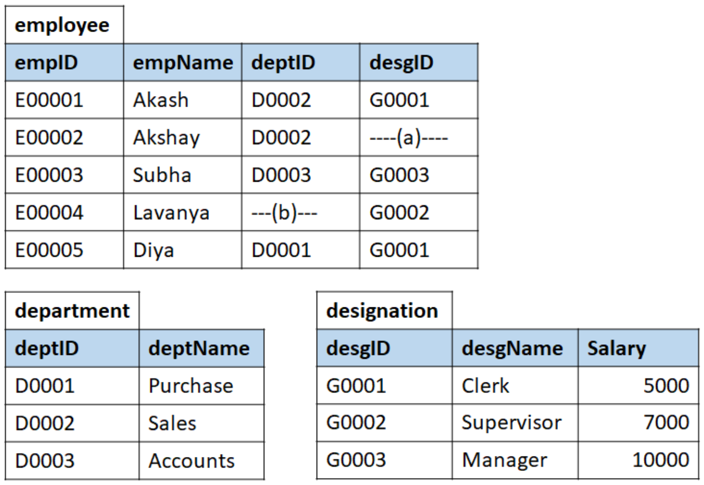
What should be filled in the blank so that the following query will return the capitals of all countries that belong to Asia but not Europe? (Write the answer as a single word in all CAPS) [NAT: 2 points]

```
SELECT capitalName FROM capital
WHERE countryID IN (SELECT countryID FROM country
                    WHERE continent =‘Asia’
                    -----------------
                    SELECT countryID FROM country
                    WHERE continent =‘Europe’);
```

Answer: EXCEPT

Solution: The first part of the inner query returns all countries that belong to Asia. If we need to find countries that belong to Asia but not Europe, then from the rows returned by the first part of the inner query, we have to remove those that contain countries that belong to Europe as well. Hence, to remove those rows, we use EXCEPT.

3. Based on the relations given in Figure 3 answer the question that follows.



employee

empID	empName	deptID	desgID
E00001	Akash	D0002	G0001
E00002	Akshay	D0002	----(a)----
E00003	Subha	D0003	G0003
E00004	Lavanya	---(b)---	G0002
E00005	Diya	D0001	G0001

department

deptID	deptName
D0001	Purchase
D0002	Sales
D0003	Accounts

designation

desgID	desgName	Salary
G0001	Clerk	5000
G0002	Supervisor	7000
G0003	Manager	10000

Figure 3: Employee instance

What should be filled in blank (a) in the table `employee` in Figure 3, if the query given below returns the value: Akshay? [NAT: 2 points]

```
SELECT empName FROM employee
WHERE desgID LIKE '%2' AND deptID LIKE '%2';
```

Answer: G0002

Solution: Since the value returned is Akshay, it corresponds to second row of table `employee`. desgID in employee table is a foreign key that references department table, hence the desgID can only be any one value from {G0001, G0002, G0003}. The `WHERE` condition specifies `desgID LIKE '%2'`. It follows from all three reasons that the only possible value that can be filled in blank (a) is G0002.

4. Consider the following SQL statement: [MSQ: 2 points]

```
CREATE TABLE boats(
    boatID VARCHAR (8),
    boatName VARCHAR (20),
    boatColour VARCHAR (8),
    yearOfPurchase INTEGER,
    weight INTEGER,
    PRIMARY KEY (boatID),
    CHECK (boatColour IN ('Black', 'White', 'Red', 'Yellow')));
```

Which among the following will cause an integrity constraint violation in the `boats` table?

- ◯ `INSERT INTO boats('B1', 'Liberty', 'Red', 2003, 500);`
- ✓ `INSERT INTO boats('B1', 'Liberty', 'Blue', 2003, 500);`
- ✓ `UPDATE boats SET boatColour = 'Green' WHERE boatID = 'B1';`
- ◯ `DELETE FROM boats;`

Solution: In option 1, there is no constraint violation.
In option 2, since the permitted colors do not include blue, it will cause a violation.
In option 3, the permitted colors do not include green and hence it will cause a violation.
In option 4, there is no constraint violation.

5. Consider the relational schema given in Figure 4.

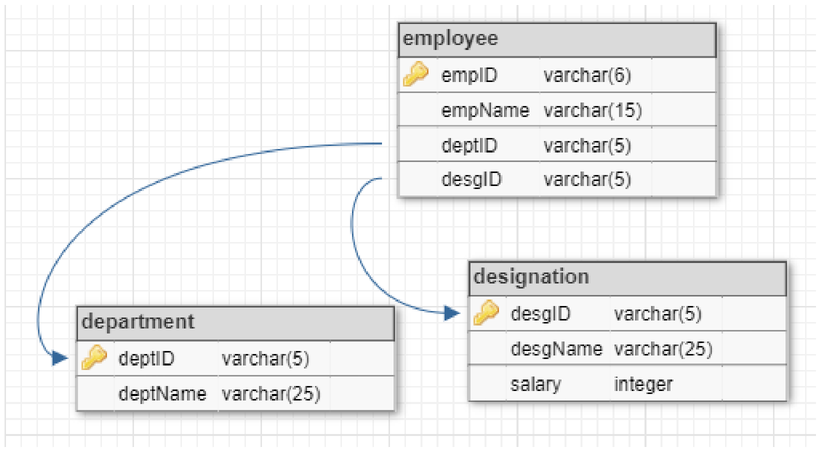


Figure 4: Employee Schema

If the relations **employee, designation** and **department** have 100, 6, 5 rows respectively, what is the difference between the maximum and the minimum number of rows returned by the following query? [NAT: 2 points]

```
SELECT * FROM employee LEFT OUTER JOIN designation
ON employee.desgID = designation.desgID;
```

Answer: 0

Solution: Left outer join (also known as Left join) returns all tuples returned by natural join along with those tuples in the left table (here, `employee`) that does not have matching entry in the right table. In the given question, however, `desgID` is the foreign key in Table `employee` that references Table `designation`. Therefore, there will not be any tuple in the left table that does not have a matching entry in the right table. Thus, the maximum number of rows returned by the left join in the given example is 100.
The case when `employee` table has no rows is the case when left outer join will have the minimum number of rows. In this case, however, the `employee` table has 100 rows. So, there will be at least 100 rows returned by the left join.
The answer is $100 - 100 = 0$.

6. Choose the appropriate query/queries to find the names of batsmen who scored the second-highest runs. [MSQ: 2 points]

√
```
SELECT name, MAX(runs) AS runs
FROM batsman WHERE runs < (SELECT MAX(runs) FROM batsman);
```

√
```
SELECT name, MAX(runs) AS runs
FROM batsman WHERE runs IN
(SELECT runs FROM batsman MINUS (SELECT MAX(runs) FROM batsman));
```

√
```
SELECT name, runs AS runs
FROM batsman WHERE runs = (SELECT runs FROM batsman
ORDER BY runs LIMIT 1,1);
```

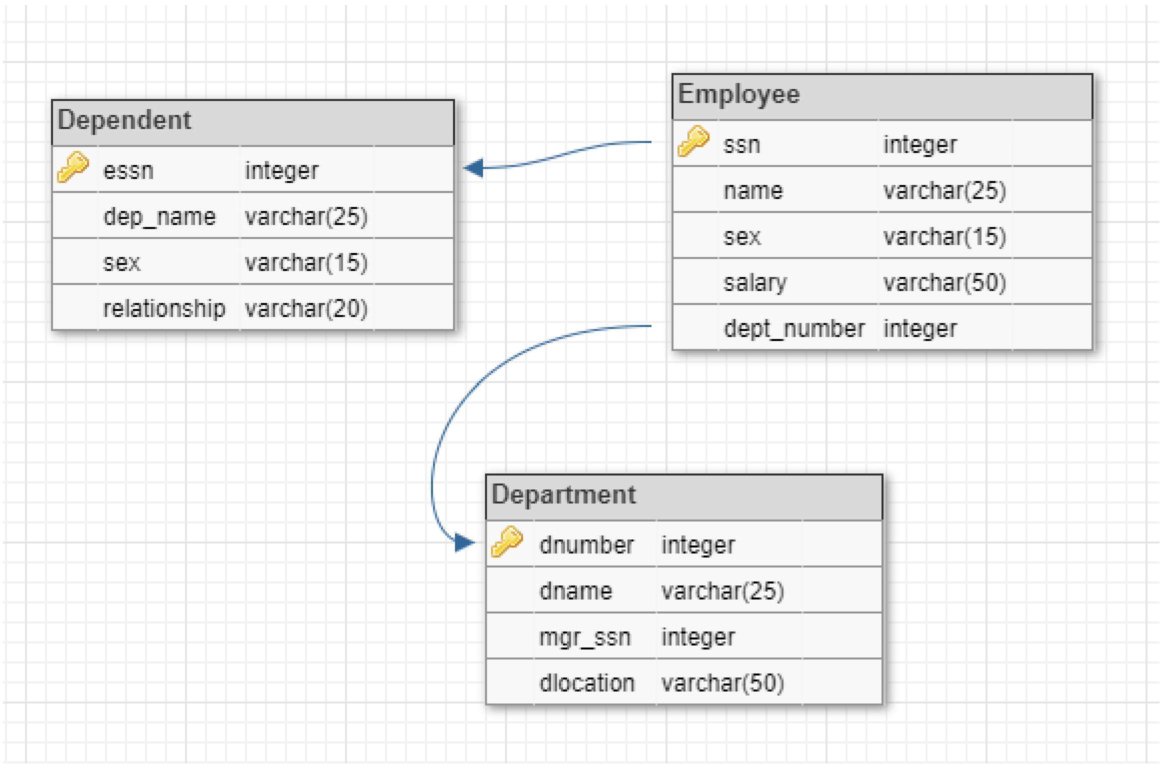
○
```
SELECT name, MAX(runs) AS runs FROM batsman
WHERE runs > (SELECT MIN(runs) FROM batsman);
```

Solution:

- MAX, MIN functions are used to find out the record with maximum and minimum values respectively among a record set.
- The SQL MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement.
- The LIMIT statement is used to limit the number of records returned based on a limit value.
- The ORDER BY keyword is used to sort the result-set in ascending or descending order.

Option 1 - The inner query will fetch the maximum runs and then the outer query will return the runs value which is maximum among all and lesser than the value retrieved by the inner query. Hence, the second-highest value of runs is fetched.
Option 2 - The inner query will fetch all the runs values other than the maximum runs and from this set, the IN operator will retrieve the maximum value. Hence, the second-highest value of runs is fetched.
Option 3 - The inner query will fetch the second-highest value of runs using the Limit operator, then using the '=' the outer query will retrieve it.
Option 4 - The inner query will return the minimum runs and the outer query will fetch the maximum runs greater than the runs value of the inner query. Hence it is incorrect.

Consider the table **Employee**, table **Department** and table **Dependent**, and answer the questions 7 and 8.



Dependent	
essn	integer
dep_name	varchar(25)
sex	varchar(15)
relationship	varchar(20)

Employee	
ssn	integer
name	varchar(25)
sex	varchar(15)
salary	varchar(50)
dept_number	integer

Department	
dnumber	integer
dname	varchar(25)
mgr_ssn	integer
dlocation	varchar(50)

7. Select the suitable query to retrieve the names of employees who have no dependents. [MCQ: 2 points]

○
```
SELECT name FROM Employee
WHERE NOT EXISTS (SELECT * FROM Dependent AS D WHERE D.ssn = essn);
```

○
```
SELECT name FROM Dependent
WHERE NOT EXISTS (SELECT * FROM Employee WHERE ssn = essn);
```

[✓
```
SELECT name
FROM Employee
WHERE NOT EXISTS (SELECT * FROM Dependent WHERE ssn = essn);
```
]

○
```
SELECT name FROM Employee
WHERE IN (SELECT * FROM Employee WHERE ssn = essn);
```

Solution: The EXISTS/NOT EXISTS condition in SQL is used to check whether the result of a correlated nested query is empty (contains no tuples) or not.
As per the question, to retrieve the names of employees who have no dependents, the outer query needs to fetch data from the table Employee and the inner query needs

to fetch data from the table Dependent. Hence, options 2 and 4 are incorrect.

In option 1- After aliasing Dependent as D, the condition must be ssn = D.essn. Hence, option 1 is incorrect.

In option 3 - Inner query will fetch all the dependents where attribute ssn of Employee is matched with essn from Dependent. Hence, only if there is no matched value, NOT EXISTS will be true and the names of the employees who have no dependents will be retrieved.

8. Select the suitable query to retrieve the names of employees who have some dependent(s) whose name ends with 'KUMAR'. [MCQ: 2 points]

◯
```
SELECT name FROM Dependent
WHERE ssn IN (SELECT essn FROM Employee
WHERE dep_name LIKE '%KUMAR');
```

◯
```
SELECT name FROM Employee
WHERE essn IN (SELECT ssn FROM Dependent
WHERE dep_name LIKE '%KUMAR%');
```

✓
```
SELECT name FROM Employee
WHERE ssn IN (SELECT essn FROM Dependent
WHERE dep_name LIKE '%KUMAR');
```

◯
```
SELECT name FROM Employee
WHERE ssn IN (SELECT essn FROM Dependent
WHERE dep_name LIKE 'KUMAR');
```

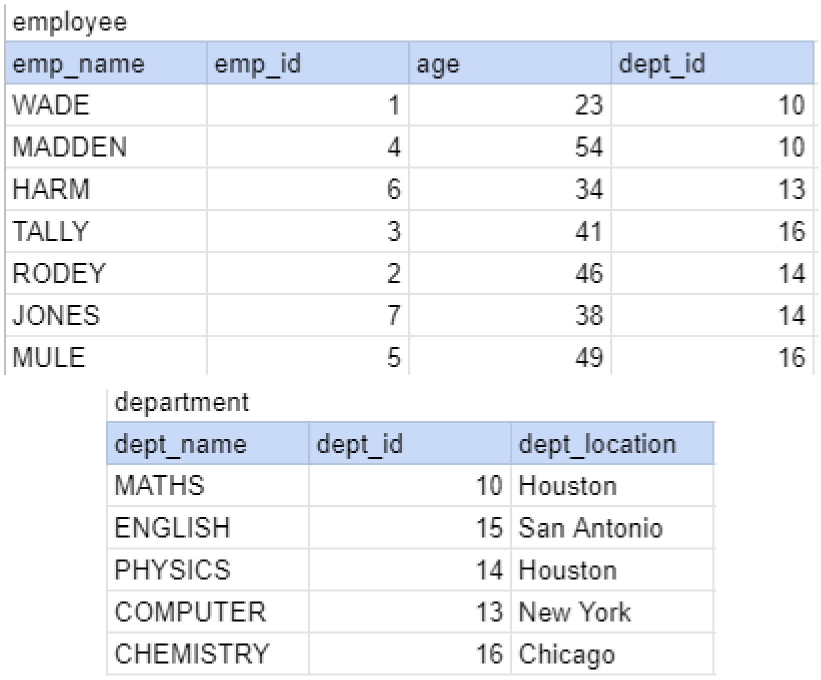
Solution: The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. The percent sign (%) represents zero, one, or multiple characters and the underscore sign (_) represents one, single character.

So to retrieve all the names ending with KUMAR, it has to match '%KUMAR'. Hence, options 2 and 4 are incorrect.

In option 1, the inner query needs to fetch from table Dependent and outer query from table Employee. Hence, incorrect.

In option 3, the inner query will fetch the Dependent(s) whose name ends with KUMAR and using IN keyword, the outer query will retrieve the names of the corresponding employees. Hence, correct.

9. Consider the table **employee** and table **department** as shown in Figure 5, and answer the question that follows. [MCQ: 2 points]



employee

emp_name	emp_id	age	dept_id
WADE	1	23	10
MADDEN	4	54	10
HARM	6	34	13
TALLY	3	41	16
RODEY	2	46	14
JONES	7	38	14
MULE	5	49	16

department

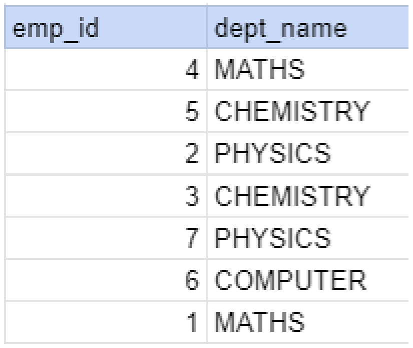
dept_name	dept_id	dept_location
MATHS	10	Houston
ENGLISH	15	San Antonio
PHYSICS	14	Houston
COMPUTER	13	New York
CHEMISTRY	16	Chicago

Figure 5: employee & department

What will be the output of the following query?

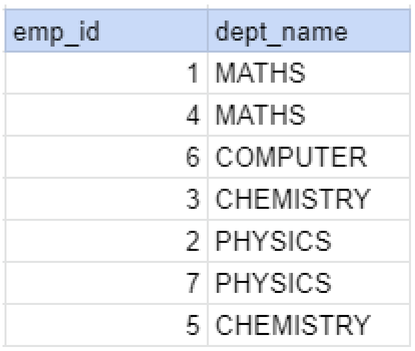
```
SELECT emp_id, dept_name
FROM employee NATURAL JOIN department
ORDER BY age desc;
```

[✓ Output:]



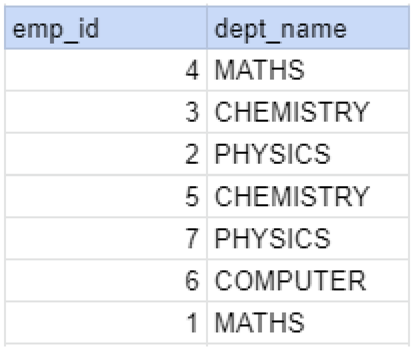
emp_id	dept_name
4	MATHS
5	CHEMISTRY
2	PHYSICS
3	CHEMISTRY
7	PHYSICS
6	COMPUTER
1	MATHS

◯ Output:



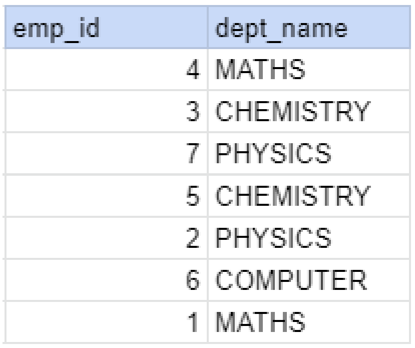
emp_id	dept_name
1	MATHS
4	MATHS
6	COMPUTER
3	CHEMISTRY
2	PHYSICS
7	PHYSICS
5	CHEMISTRY

○ Output:



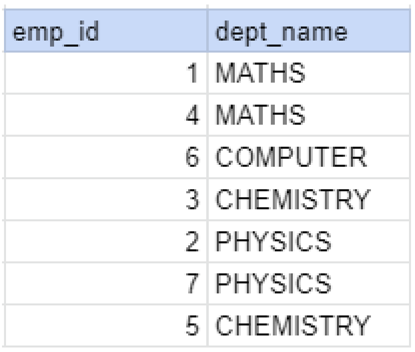
emp_id	dept_name
4	MATHS
3	CHEMISTRY
2	PHYSICS
5	CHEMISTRY
7	PHYSICS
6	COMPUTER
1	MATHS

○ Output:



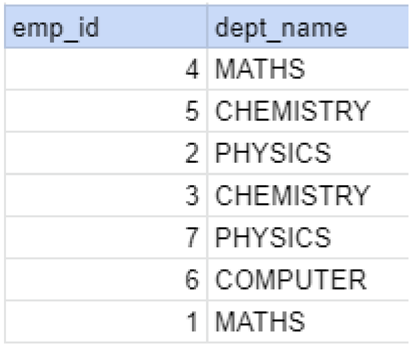
emp_id	dept_name
4	MATHS
3	CHEMISTRY
7	PHYSICS
5	CHEMISTRY
2	PHYSICS
6	COMPUTER
1	MATHS

Solution: As per the query, after NATURAL JOIN on employee table and department table, the resultant table will be -



emp_id	dept_name
1	MATHS
4	MATHS
6	COMPUTER
3	CHEMISTRY
2	PHYSICS
7	PHYSICS
5	CHEMISTRY

And as and when we put ORDER BY age in descending order, we will fetch the following resultant table -



emp_id	dept_name
4	MATHS
5	CHEMISTRY
2	PHYSICS
3	CHEMISTRY
7	PHYSICS
6	COMPUTER
1	MATHS

10. Consider a table **Employee**(*eid, edept, ename, esalary, ebonus*). The table has no records initially.

[MCQ:2 points]

```
CREATE OR REPLACE FUNCTION bonus_fun() RETURNS TRIGGER AS $$
    BEGIN
         IF NEW.edept = 'R/D' THEN
               NEW.ebonus = NEW.esalary * .75;
         END IF;
         RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER bonus_trig
    BEFORE INSERT ON Employee
    EXECUTE PROCEDURE bonus_fun();

INSERT INTO Employee VALUES (4,'R/D','Diksha',30000);
INSERT INTO Employee VALUES (2,'Accounts','Raj',40000);
SELECT ebonus FROM Employee;
```

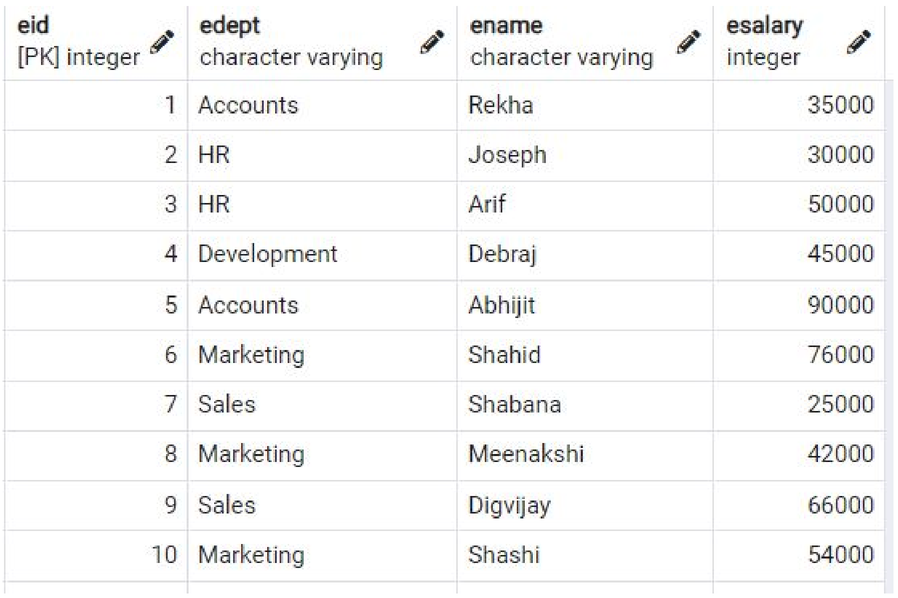
If the given code is executed, then what will be the output?

- ◯ 22500
  0
- ◯ 22500
  NULL
- ◯ 22500
  30000
- √ The code has errors.

Solution: The code is erroneous because the trigger definition does not explicitly mention its granularity (*for each row* or *for each statement*). This trigger checks each insertion and modifies the value of an attribute (*ebonus*) when the described condition satisfies, therefore it should work as a row level trigger.

11. Consider an instance of the table **Employee** given below.

[MCQ:2 points]



eid [PK] integer	edept character varying	ename character varying	esalary integer
1	Accounts	Rekha	35000
2	HR	Joseph	30000
3	HR	Arif	50000
4	Development	Debraj	45000
5	Accounts	Abhijit	90000
6	Marketing	Shahid	76000
7	Sales	Shabana	25000
8	Marketing	Meenakshi	42000
9	Sales	Digvijay	66000
10	Marketing	Shashi	54000

Figure 6: Table: Employee

If the given code is executed on this instance, then what will be the output/error?

[MCQ:2 points]

```
CREATE OR REPLACE FUNCTION salary_fun() RETURNS TRIGGER AS $$
DECLARE
    counter INT := 0;
BEGIN
    IF  NEW.esalary > 75000 THEN
            counter = counter + 1;
            RAISE NOTICE 'Number of affected rows : %', counter;
            --//This statement prints => NOTICE: <whatever follows>
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER salary_trig
    AFTER UPDATE ON Employee
    FOR EACH ROW
    EXECUTE PROCEDURE salary_fun();

UPDATE Employee SET esalary = esalary * 1.5;
```

- ◯ NOTICE: Number of affected rows : 4
- ◯ NOTICE: Number of affected rows : 10
- ◯ NOTICE: Number of affected rows : 4
  NOTICE: Number of affected rows : 4
  NOTICE: Number of affected rows : 4
  NOTICE: Number of affected rows : 4
- ✓ None of the above

Solution: The trigger will be executed on every row affected by the `UPDATE` statement. Only 4 rows in the given instance will have their new salary more than 75000. The `RAISE NOTICE` statement is inside the `IF` clause, thus it will be executed 4 times. Observe the fact that the trigger fires for each updated row, and hence every time the variable *counter* is reinitialized with 0. It will be incremented to 1 with respect to that specific row, and thus we will get the output as:
NOTICE: Number of affected rows : 1
NOTICE: Number of affected rows : 1
NOTICE: Number of affected rows : 1
NOTICE: Number of affected rows : 1

12. If we want to store/print the number of affected rows when an update or delete statement is executed, then which type of trigger should we use to count?

[MCQ:2 points]

- ✓ Statement level trigger
- ◯ Row level trigger
- ◯ Both are equally efficient
- ◯ Table level trigger

Solution: If we want to count the number of affected rows then we need not execute a trigger every time for each row. After all the modifications are over, a single execution of a trigger to count the affected rows should be done. Running a row level trigger will simply do the same job again and again for all the affected rows. Hence, Statement level triggers should be used here.