IT252
Database Management System

Assignment VII

NIRAJ NANDISH

191IT234 | IT252 | 19/03/2021

1. Create stored procedure usp_get_employees_salary_above that accept a number as parameter and return all employees' first and last names whose salary is above or equal to the given number. The result should be sorted by first_name then by last_name alphabetically.

```
> CREATE TABLE Employee(Emp_number INT NOT NULL PRIMARY KEY,
first_name VARCHAR(30), last_name VARCHAR(30), Salary INT);
>
> INSERT INTO Employee VALUES
> (1, 'William', 'Hartnell', 10000),
> (2, 'Patrick', 'Throughton', 20000),
> (3, 'Jon', 'Pertwee', 30000),
> (4, 'Tom', 'Baker', 40000),
> (5, 'Peter', 'Davison', 50000),
> (6, 'Colin', 'Baker', 60000),
> (7, 'Sylvester', 'McCoy', 70000),
> (8, 'Paul', 'McGann', 80000),
> (9, 'Christopher', 'Eccleston', 90000),
> (10, 'David', 'Tennant', 100000),
> (11, 'Matt', 'Smith', 110000),
> (12, 'Peter', 'Capaldi', 120000),
> (13, 'Jodie', 'Whittaker', 130000);
>
> DELIMITER $$
> CREATE PROCEDURE usp_get_employees_salary_above(IN sal INT)
> BEGIN
> SELECT first_name, last_name
> FROM Employee
> WHERE Salary >= sal
> ORDER BY first_name, last_name;
> END $$
> DELIMITER ;
> CALL usp_get_employees_salary_above(90000);
```

```
mysql> delimiter $$
mysql> create procedure usp_get_employees_salary_above(in sal int)
    -> begin
    -> select first_name, last_name
    -> from Employee
    -> where Salary >= sal
    -> order by first_name, last_name;
    -> end $$
Query OK, 0 rows affected (0.77 sec)

mysql> delimiter ;
mysql> call usp_get_employees_salary_above(90000);
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Christopher | Eccelston |
| David      | Tennant   |
| Jodie      | Whittaker |
| Matt       | Smith     |
| Peter      | Capaldi   |
+------------+-----------+
5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

2. Write a stored procedure usp_get_towns_starting_with that accept string as parameter and returns all town names starting with that string. The result should be sorted by town name alphabetically.

```
> CREATE TABLE pgm2(Name VARCHAR(20));
>
> INSERT INTO pgm2 VALUES
> ('Bangalore'),
> ('Delhi'),
> ('Mysore'),
> ('Mangalore');
>
> DELIMITER $$
> CREATE PROCEDURE usp_get_towns_starting_with(prefix VARCHAR(20))
> BEGIN
> SELECT Name AS "town"
> FROM pgm2
> WHERE Name LIKE concat(prefix, "%")
> ORDER BY Name;
> END $$
> DELIMITER ;
> CALL usp_get_towns_starting_with("M");
```

```
mysql> delimiter $$
mysql> create procedure usp_get_towns_starting_with(prefix varchar(20))
    -> begin
    -> select Name as "town"
    -> from pgm2
    -> where Name like concat(prefix, "%")
    -> order by Name;
    -> end $$
Query OK, 0 rows affected (1.94 sec)

mysql> delimiter ;
mysql> call usp_get_towns_starting_with('M');
+-----------+
| town      |
+-----------+
| Mangalore |
| Mysore    |
+-----------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

3. Write a function ufn_get_salary_level that receives salary of an employee and returns the level of the salary. If salary is < 30000 return “Low”, if salary is between 30000 and 50000 (inclusive) return “Average” or If salary is > 50000 return “High”.

```
> DELIMITER $$
> CREATE FUNCTION ufn_get_salary_level(sal INT)
> RETURNS VARCHAR(10)
> DETERMINISTIC
> BEGIN
> DECLARE lvl VARCHAR(10);
> IF sal < 30000 THEN
> SET lvl = 'Low';
> ELSEIF (sal >= 30000 AND sal <= 50000) THEN
> SET lvl = 'Average';
> ELSEIF sal > 50000 THEN
> SET lvl = 'High';
> END IF;
> RETURN (lvl);
> END $$
> DELIMITER ;
> SELECT ufn_get_salary_level(20000);
> SELECT ufn_get_salary_level(40000);
> SELECT ufn_get_salary_level(60000);
```

```
mysql> create function ufn_get_salary_level(sal int)
    -> returns varchar(10)
    -> deterministic
    -> begin
    -> declare lvl varchar(10);
    -> if sal < 30000 then
    -> set lvl = 'Low';
    -> elseif (sal >= 30000 and sal <= 50000) then
    -> set lvl = 'Average';
    -> elseif sal > 50000 then
    -> set lvl = 'High';
    -> end if;
    -> return (lvl);
    -> end $$
Query OK, 0 rows affected (1.24 sec)

mysql> delimiter ;
mysql> select ufn_get_salary_level(20000);
+-----------------------------+
| ufn_get_salary_level(20000) |
+-----------------------------+
| Low                         |
+-----------------------------+
1 row in set (0.00 sec)
```

4. Create a function ufn_calculate_future_value that accepts as parameters – sum, yearly interest rate and number of years. It should calculate and return the future value of the initial sum. Using the following formula: FV= I×((1+R)T), I – Initial sum, R – Yearly interest rate, T – Number of years .

```
> DELIMITER $$
> CREATE FUNCTION ufn_calculate_future_value(p DECIMAL(8,2), r
DECIMAL(5,2), t DECIMAL(5,2))
> RETURNS DECIMAL(8,2)
> DETERMINISTIC
> BEGIN
> DECLARE interest DECIMAL(8,2);
> SELECT POW(1+r, t) INTO interest;
> SET interest = p*interest;
> RETURN (interest);
> END $$
> SELECT ufn_calculate_future_value(1000, 0.1, 5);
```

```
mysql> delimiter $$
mysql> create function ufn_calculate_future_value(p decimal(8,2), r decimal(5,2), t decimal(5,2))
    -> returns decimal(8,2)
    -> deterministic
    -> begin
    -> declare interest decimal(8,2);
    -> select pow(1+r, t) into interest;
    -> set interest = p*interest;
    -> return (interest);
    -> end $$
Query OK, 0 rows affected (2.45 sec)

mysql> delimiter ;
mysql> select ufn_calculate_future_value(1000, 0.1, 5);
+-----------------------------------------+
| ufn_calculate_future_value(1000, 0.1, 5) |
+-----------------------------------------+
|                                 1610.00 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

5. Create stored procedure Student_Marks that accept a Roll_number as parameter and return all students Name with their total marks whose total is above or equal to the given Roll number student Marks. The result should be reversely sorted by Total.

```
> CREATE TABLE Student(Roll_number INTEGER NOT NULL PRIMARY KEY, Name
VARCHAR(50), Total_Marks INTEGER);
>
> INSERT INTO Student Values
> (34, 'Prasad', 460),
> (25, 'Gaurang', 465),
> (3, 'Binod', 450),
> (4, 'Pranav', 485),
> (30, 'Vijay', 490),
> (12, 'Kumar', 453),
> (8, 'Sanjay', 480),
> (6, 'Mihir', 440),
> (5, 'Nitya', 495),
> (15, 'Pushpa', 495),
> (21, 'Radhika', 455),
> (18, 'Mahika', 470),
> (11, 'Bhavika', 465);
>
> DELIMITER $$
> CREATE PROCEDURE Student_Marks(IN rn INT)
> BEGIN
> SELECT Name, Total_Marks
> FROM Student
> WHERE Total_Marks >= (SELECT Total_Marks FROM Student WHERE
Roll_number = rn)
> ORDER BY Total_Marks DESC;
> END $$
> DELIMITER ;
> CALL Student_Marks(8);
```

```
mysql> delimiter $$
mysql> create procedure Student_Marks(in rn int)
    -> begin
    -> select Name, Total_Marks
    -> from Student
    -> where Total_Marks >= (select Total_Marks from Student where Roll_number = rn)
    -> order by Total_Marks desc;
    -> end $$
Query OK, 0 rows affected (2.65 sec)

mysql> delimiter ;
mysql> call Student_Marks(8);
+--------+-------------+
| Name   | Total_Marks |
+--------+-------------+
| Nitya  |         495 |
| Pushpa |         495 |
| Vijay  |         490 |
| Pranav |         485 |
| Sanjay |         480 |
+--------+-------------+
5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

6. Write a function Get_Grade_level that receives Roll_number of a student and returns the level of the marks of him / her. If Total_Marks is < 300 return “C”, If Total_Marks is between 300 and 400 (inclusive) return “B”, if Total_Marks is between 401 and 450 (inclusive) > return “A”, if Total_Marks is between 451 and 475 (inclusive)> return “A+” or if Total_Marks is > 475 return “S”.

```
> DELIMITER $$
> CREATE FUNCTION Get_Grade_level(rn INT)
> RETURNS CHAR(2)
> DETERMINISTIC
> BEGIN
> DECLARE t_marks INT;
> DECLARE lvl CHAR(2);
> SELECT Total_Marks INTO t_marks FROM Student WHERE Roll_number = rn;
> IF t_marks < 300 THEN
> SET lvl = "C";
> ELSEIF (t_marks >= 300 AND t_marks <= 400) THEN
> SET lvl = "B";
> ELSEIF (t_marks > 400 AND t_marks <= 450) THEN
> SET lvl = "A";
> ELSEIF (t_marks > 450 AND t_marks <= 475) THEN
> SET lvl = "A+";
> ELSEIF t_marks > 475 THEN
> SET lvl = "S";
> END IF;
> RETURN (lvl);
> END $$
> DELIMITER ;
> SELECT Get_Grade_level(8);
```

```
mysql> delimiter $$
mysql> create function Get_Grade_level(rn int)
    -> returns char(2)
    -> deterministic
    -> begin
    -> declare t_marks int;
    -> declare lvl char(2);
    -> select Total_Marks into t_marks from Student where Roll_number = rn;
    -> if t_marks < 300 then
    -> set lvl = "C";
    -> elseif (t_marks >= 300 and t_marks <= 400) then
    -> set lvl = "B";
    -> elseif (t_marks > 400 and t_marks <= 450) then
    -> set lvl = "A";
    -> elseif (t_marks > 450 and t_marks <= 475) then
    -> set lvl = "A+";
    -> elseif t_marks > 475 then
    -> set lvl = "S";
    -> end if;
    -> return (lvl);
    -> end $$
Query OK, 0 rows affected (2.77 sec)

mysql> delimiter ;
mysql> select Get_Grade_level(8)
    -> ;
+--------------------+
| Get_Grade_level(8) |
+--------------------+
| S                  |
+--------------------+
1 row in set (0.00 sec)
```

7. Write a function Get_Sum that receives Limit as a parameter and returns sum of Natural numbers up to Limit.

```
> DELIMITER $$
> CREATE FUNCTION Get_Sum(num INT)
> RETURNS INT
> DETERMINISTIC
> BEGIN
> RETURN (num*(num+1)/2);
> END $$
> DELIMITER ;
> SELECT Get_Sum(5);
```

```
mysql> delimiter $$
mysql> create function Get_Sum(num int)
    -> returns int
    -> deterministic
    -> begin
    -> return (num*(num+1)/2);
    -> end $$
Query OK, 0 rows affected (3.45 sec)

mysql> delimiter ;
mysql> select Get_Sum(5);
+------------+
| Get_Sum(5) |
+------------+
|         15 |
+------------+
1 row in set (0.00 sec)

mysql> select Get_Sum(100);
+--------------+
| Get_Sum(100) |
+--------------+
|         5050 |
+--------------+
1 row in set (0.00 sec)
```

8. Write a function IT_Rate_employee that accepts a salary as input and returns Income-tax amount to the calling function as output. IT_Rate Calculated based on the Salary is given below. If Basicpay is > 100000, IT = 20%, if Basicpay is between 100000 and 50000 (inclusive), IT = 15% or if Basicpay is < 50000, IT = 10% .

```
> DELIMITER $$
> CREATE FUNCTION IT_Rate_employee(sal INT)
> RETURNS DECIMAL(8,2)
> DETERMINISTIC
> BEGIN
> DECLARE it_rate INT;
> IF sal > 100000 THEN
> SET it_rate = 20;
> ELSEIF (sal <= 100000 AND sal >= 50000) THEN
> SET it_rate = 15;
> ELSEIF sal < 50000 THEN
> SET it_rate = 10;
> END IF;
> RETURN (sal * it_rate/100);
> END $$
> DELIMITER ;
> SELECT IT_Rate_employee(60000);
```

```
mysql> delimiter $$
mysql> create function IT_Rate_employee(sal int)
    -> returns decimal(8,2)
    -> deterministic
    -> begin
    -> declare it_rate int;
    -> if sal > 100000 then
    -> set it_rate = 20;
    -> elseif (sal <= 100000 and sal >= 50000) then
    -> set it_rate = 15;
    -> elseif sal < 50000 then
    -> set it_rate = 10;
    -> end if;
    -> return (sal * it_rate/100);
    -> end $$
Query OK, 0 rows affected (2.91 sec)

mysql> delimiter ;
mysql> select IT_Rate_employee(60000);
+-------------------------+
| IT_Rate_employee(60000) |
+-------------------------+
|                 9000.00 |
+-------------------------+
1 row in set (0.00 sec)
```

9. Create a Stored Procedure Net_Pay_employee that accepts a Emp_no. as input parameter and returns Gross_pay and Net_Pay of all employees. The result should be sorted based on the Net_Pay value (HRA= 12% of BP and DA=10% of BP).

ASSUMPTION: I used the same table as 1st Q, only modified it to the required schema.

```
> ALTER TABLE Employee RENAME COLUMN Salary TO Basicpay;
> ALTER TABLE Employee ADD Gross_pay DECIMAL(8,2);
> ALTER TABLE Employee ADD Net_pay DECIMAL(8,2);
>
> DELIMITER $$
> CREATE PROCEDURE Net_Pay_employee(IN emp_no INT)
> BEGIN
> DECLARE sal INT;
> DECLARE it INT;
> DECLARE gp INT;
> DECLARE np INT;
> SELECT Basicpay INTO sal FROM Employee WHERE Emp_number = emp_no;
> SET it = IT_Rate_employee(sal);
> SET gp = sal + (sal*0.12) + (sal*0.1);
> SET np = gp - it;
> UPDATE Employee SET Gross_pay = gp WHERE Emp_number = emp_no;
> UPDATE Employee SET Net_pay = np WHERE Emp_number = emp_no;
> SELECT Gross_pay, Net_pay
> FROM Employee
> WHERE Gross_pay IS NOT NULL AND Net_pay IS NOT NULL
> ORDER BY Net_pay DESC;
> END $$
> DELIMITER ;
> CALL Net_Pay_employee(1);
> CALL Net_Pay_employee(8);
> CALL Net_Pay_employee(12);
```

```
mysql> delimiter $$
mysql> create procedure Net_Pay_employee(in emp_no int)
    -> begin
    -> declare sal int;
    -> declare it int;
    -> declare gp int;
    -> declare np int;
    -> select Basicpay into sal from Employee where Emp_number = emp_no;
    -> set it = IT_Rate_employee(sal);
    -> set gp = sal + (sal*0.12) + (sal*0.1);
    -> set np = gp - it;
    -> update Employee set Gross_pay = gp where Emp_number = emp_no;
    -> update Employee set Net_pay = np where Emp_number = emp_no;
    -> select Gross_pay, Net_pay
    -> from Employee
    -> where Gross_pay is not null and Net_pay is not null
    -> order by Net_pay desc;
    -> end $$
Query OK, 0 rows affected (3.04 sec)

mysql> delimiter ;
mysql> call Net_Pay_employee(1);
+-----------+----------+
| Gross_pay | Net_pay  |
+-----------+----------+
|  12200.00 | 11200.00 |
+-----------+----------+
1 row in set (3.51 sec)

Query OK, 0 rows affected (3.51 sec)

mysql> call Net_Pay_employee(8);
+-----------+----------+
| Gross_pay | Net_pay  |
+----------+----------+
|  97600.00 | 85600.00 |
|  12200.00 | 11200.00 |
+-----------+----------+
2 rows in set (3.22 sec)

Query OK, 0 rows affected (3.22 sec)

mysql> call Net_Pay_employee(12);
+-----------+-----------+
| Gross_pay | Net_pay   |
+-----------+-----------+
| 146400.00 | 122400.00 |
|  97600.00 |  85600.00 |
|  12200.00 |  11200.00 |
+-----------+-----------+
3 rows in set (1.90 sec)
```

10. Create a Stored Procedure for to implement anyone error/exception handler in Mysql.

```
> CREATE TABLE Books(BookID INT PRIMARY KEY, Title VARCHAR(50));
>
> DELIMITER $$
> CREATE PROCEDURE InsertBooks(IN inBookID INT, IN inTitle VARCHAR(50))
> BEGIN
> DECLARE EXIT HANDLER FOR 1062
> BEGIN
> SELECT CONCAT('Duplicate key: BookID ', inBookID, ' exists') AS
message;
> END;
> DECLARE EXIT HANDLER FOR 1146
> BEGIN
> SELECT 'Table Books missing. Please create table first.' AS Message;
> END;
> INSERT INTO Books(BookID, Title) VALUES (inBookID, inTitle);
> SELECT count(*) FROM Books WHERE BookID = inBookID;
> END $$
> DELIMITER ;
> CALL InsertBooks(1, 'Thrawn');
> CALL InsertBooks(1, 'Chaos Rising');
> DROP TABLE Books;
> CALL InsertBooks(2, 'Hello There');
```

```
mysql> create table Books(BookID int primary key, Title varchar(50));
Query OK, 0 rows affected (8.66 sec)

mysql> delimiter $$
mysql> create procedure InsertBooks(in inBookID int, in inTitle varchar(50))
    -> begin
    -> declare exit handler for 1062
    -> begin
    -> select concat('Duplicate key: BookID ', inBookID, ' exists') as message;
    -> end;
    -> declare exit handler for 1146
    -> begin
    -> select 'Table Books missing. Please create table first.' as Message;
    -> end;
    -> insert into Books(BookID, Title) values (inBookID, inTitle);
    -> select count(*) from Books where BookID = inBookID;
    -> end $$
Query OK, 0 rows affected (1.78 sec)

mysql> delimiter ;
mysql> call InsertBooks(1, 'Thrawn');
+----------+
| count(*) |
+----------+
|        1 |
+----------+
1 row in set (0.54 sec)

Query OK, 0 rows affected (0.54 sec)

mysql> call InsertBooks(1, 'Chaos Rising');
+--------------------------------+
| message                        |
+--------------------------------+
| Duplicate key: BookID 1 exists |
+--------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> drop table Books;
Query OK, 0 rows affected (2.99 sec)

mysql> call InsertBooks(2, 'Hello There');
+-----------------------------------------------+
| Message                                       |
+-----------------------------------------------+
| Table Books missing. Please create table first. |
+-----------------------------------------------+
1 row in set (0.00 sec)
```