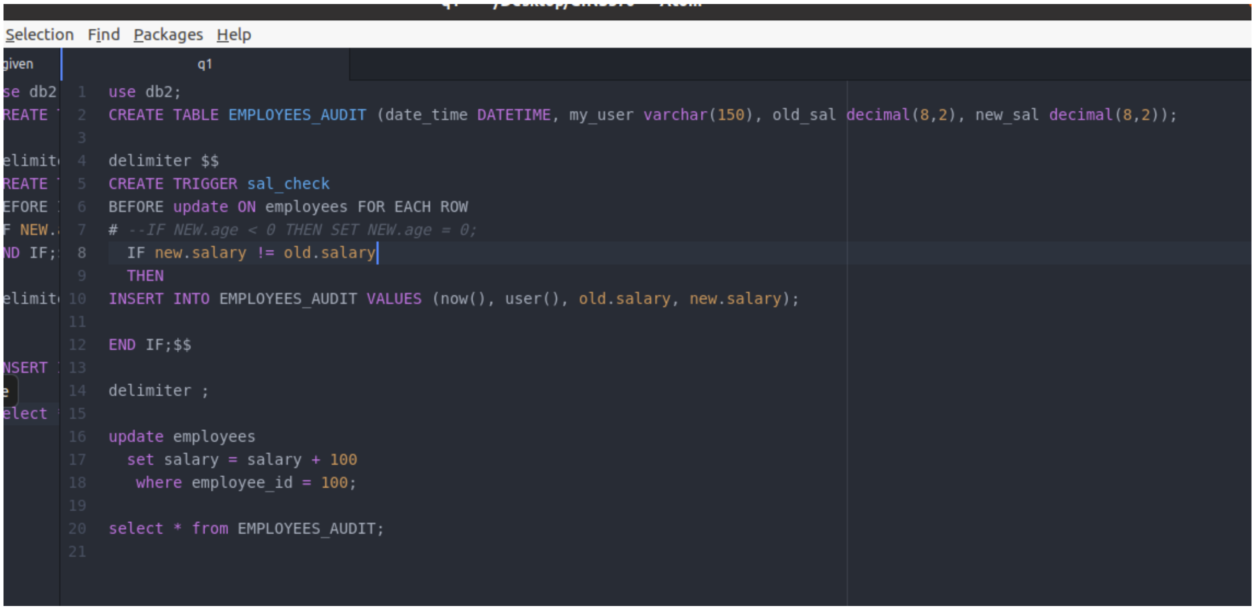
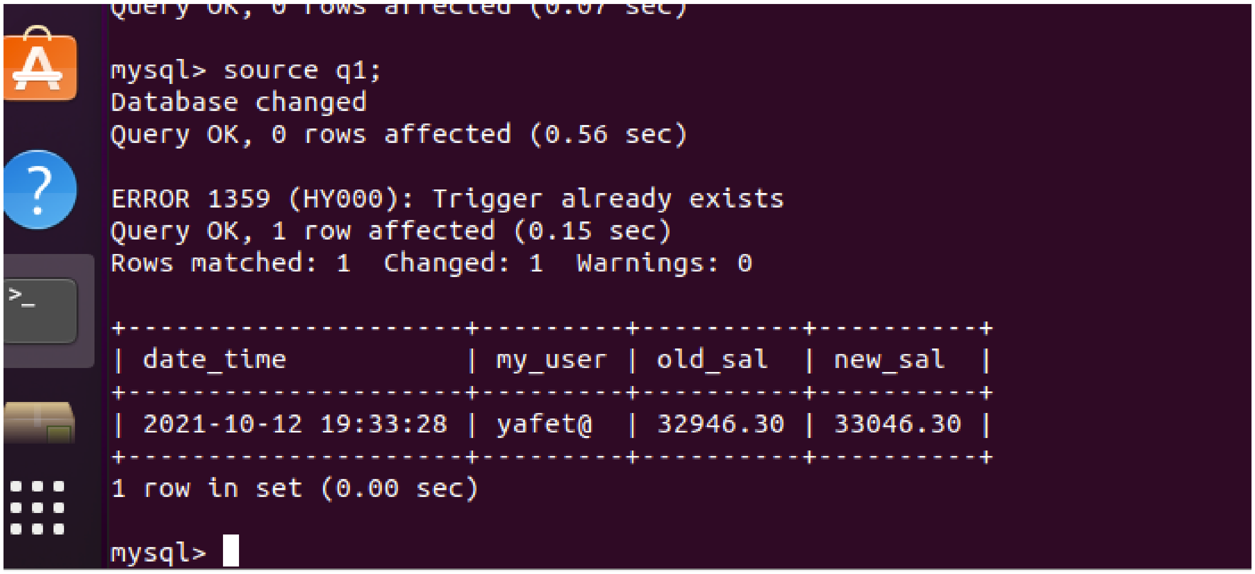
1. 8 pts. Implement a logging trigger. Create the EMPLOYEES_AUDIT table that logs activity on all salary updates done on the EMPLOYEES table (from our HR DB), using a TRIGGER. The activity should be the full system date and time, the user who performed the update, a copy of the old and new values. Make sure to test and show the contents of the EMPLOYEES_AUDIT.

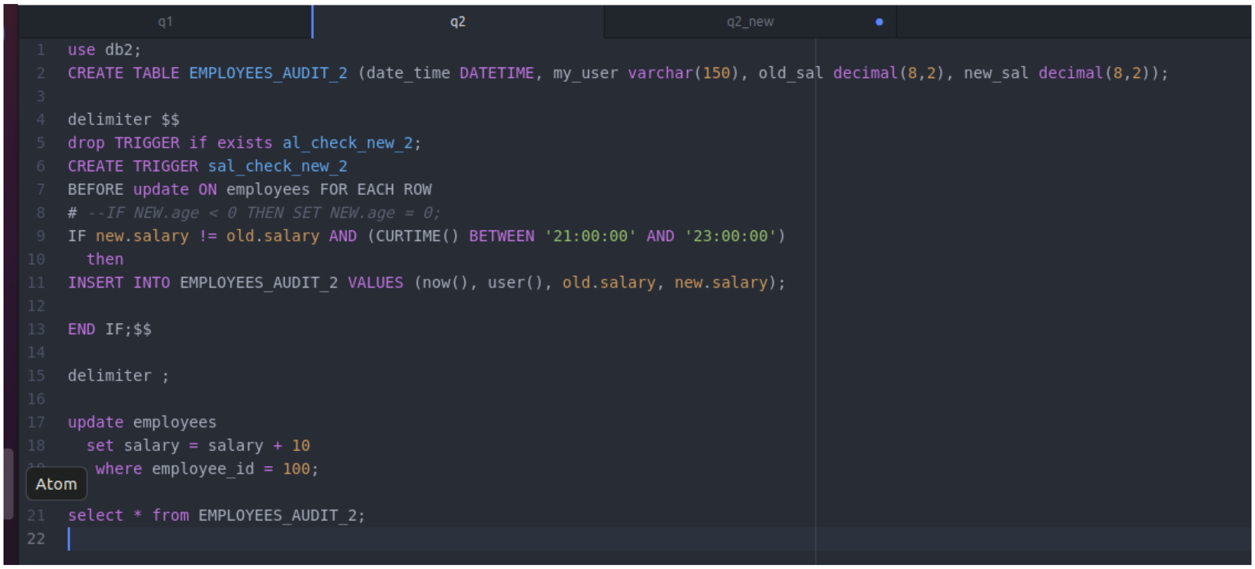


```
use db2;
CREATE TABLE EMPLOYEES_AUDIT (date_time DATETIME, my_user varchar(150), old_sal decimal(8,2), new_sal decimal(8,2));

delimiter $$
CREATE TRIGGER sal_check
BEFORE update ON employees FOR EACH ROW
# --IF NEW.age < 0 THEN SET NEW.age = 0;
  IF new.salary != old.salary
  THEN
INSERT INTO EMPLOYEES_AUDIT VALUES (now(), user(), old.salary, new.salary);

END IF;$$

delimiter ;

update employees
  set salary = salary + 100
   where employee_id = 100;

select * from EMPLOYEES_AUDIT;
```

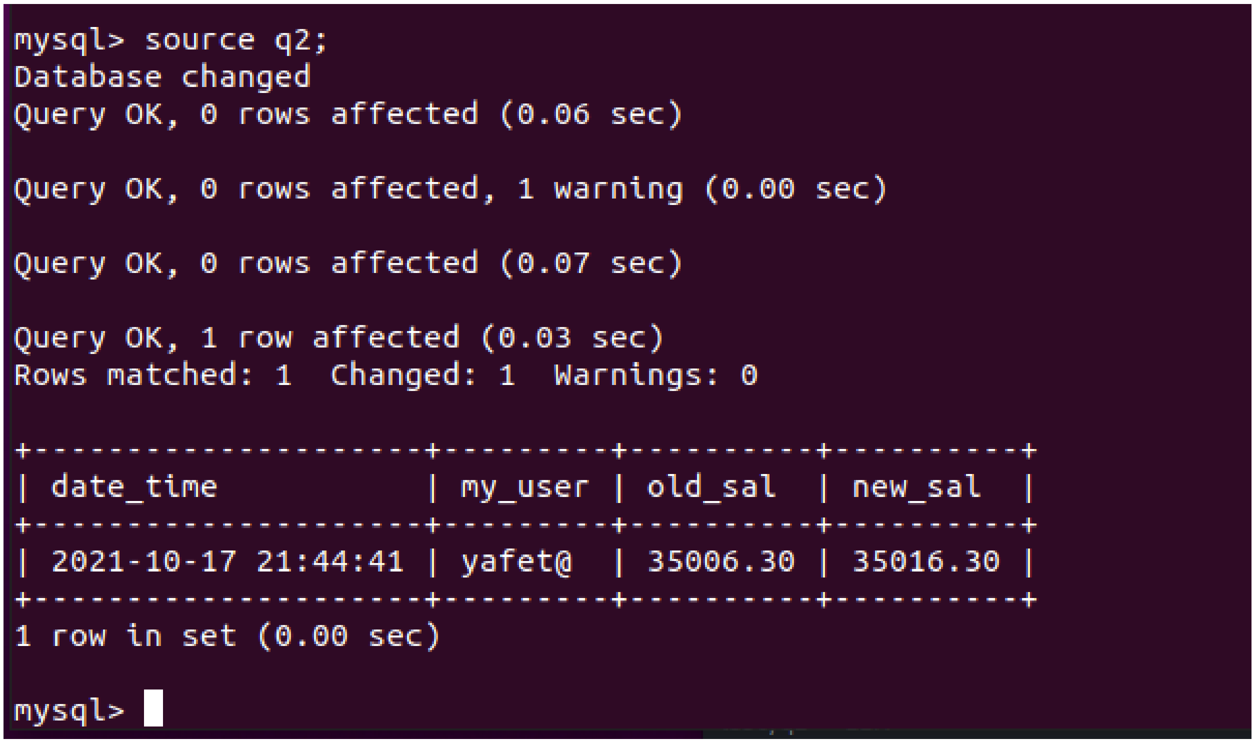


```
Query OK, 0 rows affected (0.07 sec)

mysql> source q1;
Database changed
Query OK, 0 rows affected (0.56 sec)

ERROR 1359 (HY000): Trigger already exists
Query OK, 1 row affected (0.15 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+---------------------+---------+----------+----------+
| date_time           | my_user | old_sal  | new_sal  |
+---------------------+---------+----------+----------+
| 2021-10-12 19:33:28 | yafet@  | 32946.30 | 33046.30 |
+---------------------+---------+----------+----------+
1 row in set (0.00 sec)

mysql> 
```

2. 8 pts. Implementing a timed-based trigger. Create a trigger that is executed if an update is being done on the salary field of the EMPLOYEES table between the hours of 2AM and 6AM. To test your code, it is OK to either change the system time of your computer or modify the trigger time period (i.e. 2PM to 4PM if you are testing at 3PM). The log activity should be the full system date and time, and the user who performed the update. Make sure to test your code by including an example of the trigger executing.

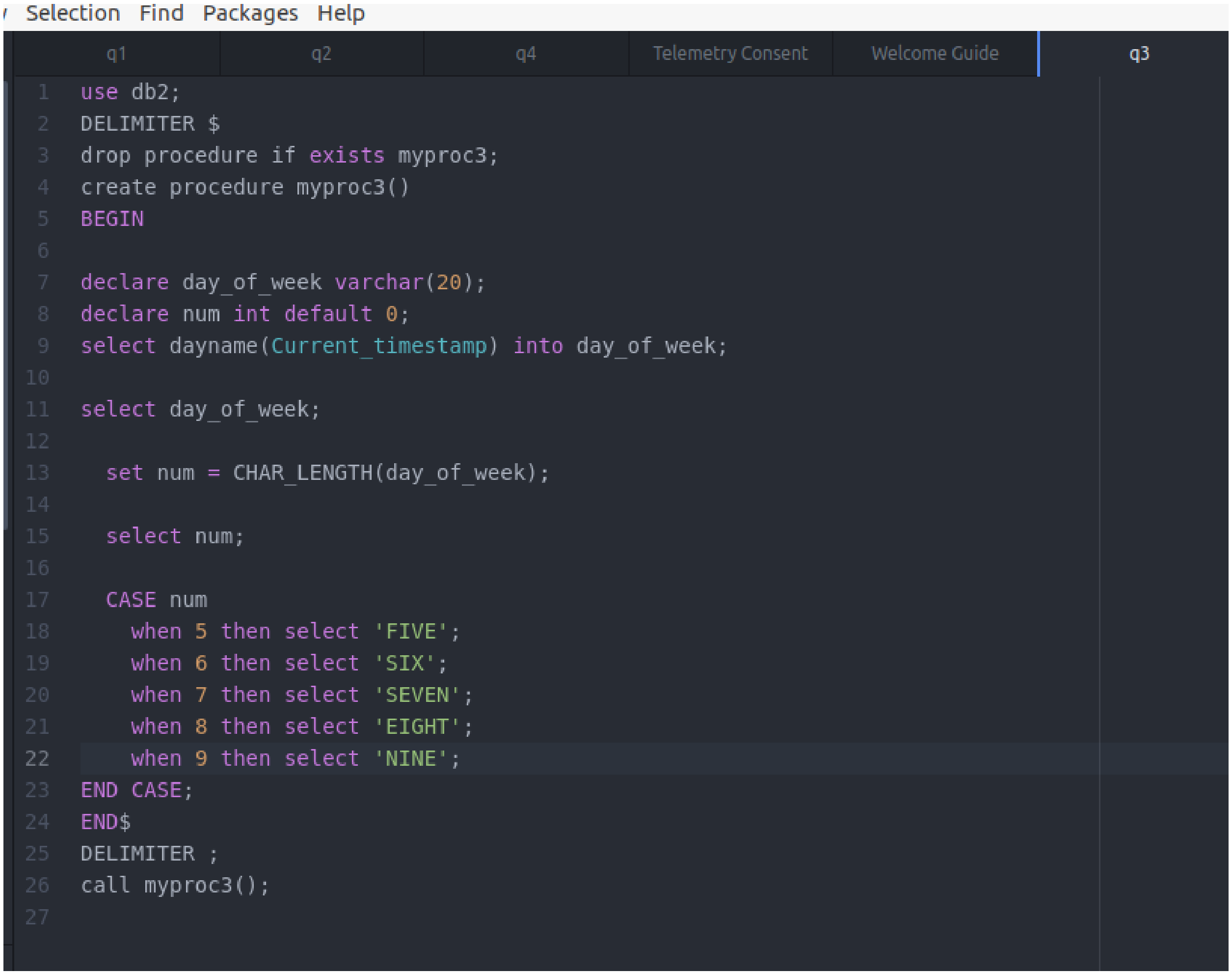


```
use db2;
CREATE TABLE EMPLOYEES_AUDIT_2 (date_time DATETIME, my_user varchar(150), old_sal decimal(8,2), new_sal decimal(8,2));

delimiter $$
drop TRIGGER if exists al_check_new_2;
CREATE TRIGGER sal_check_new_2
BEFORE update ON employees FOR EACH ROW
# --IF NEW.age < 0 THEN SET NEW.age = 0;
IF new.salary != old.salary AND (CURTIME() BETWEEN '21:00:00' AND '23:00:00')
  then
INSERT INTO EMPLOYEES_AUDIT_2 VALUES (now(), user(), old.salary, new.salary);

END IF;$$

delimiter ;

update employees
  set salary = salary + 10
   where employee_id = 100;

select * from EMPLOYEES_AUDIT_2;

```

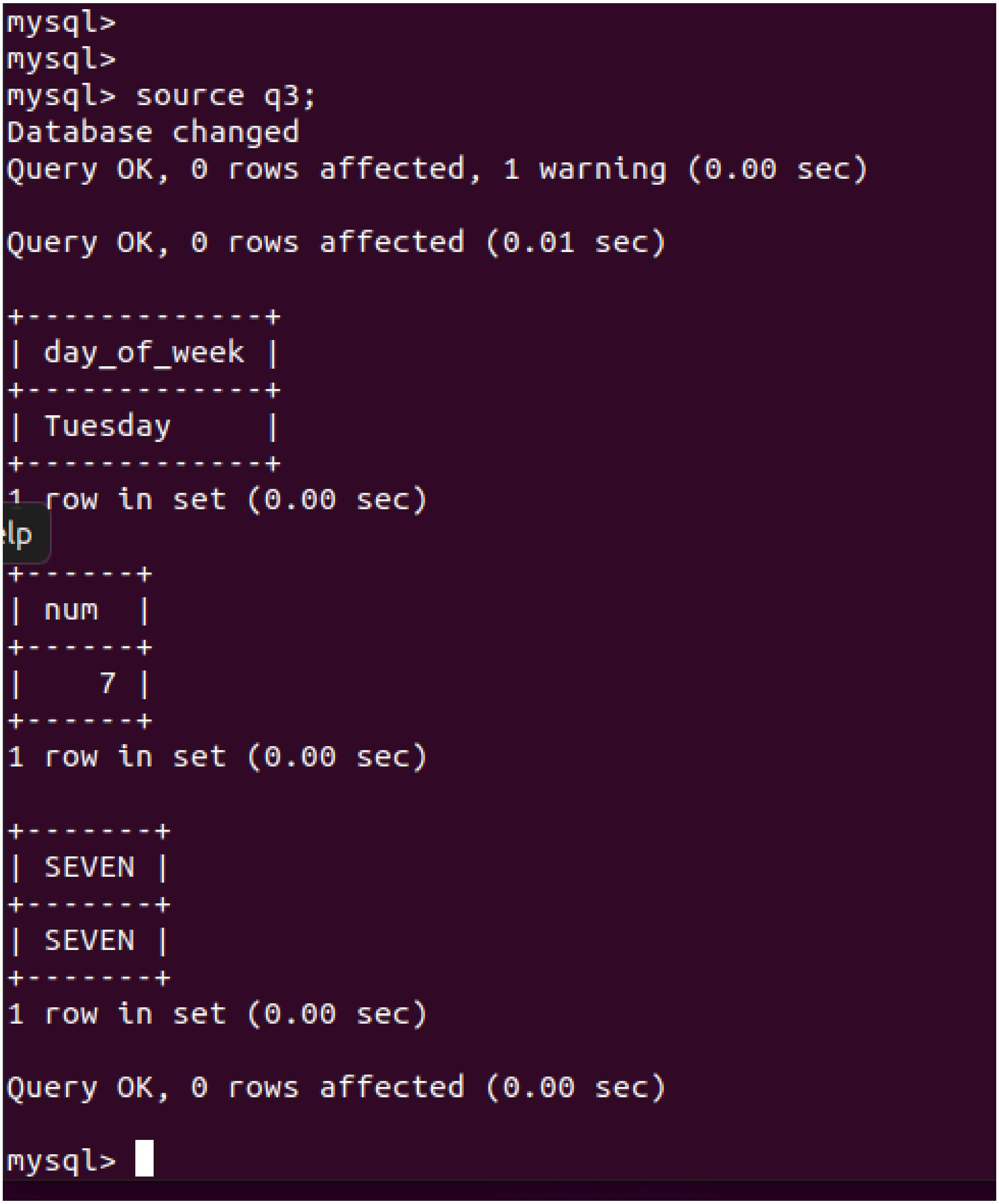


```
mysql> source q2;
Database changed
Query OK, 0 rows affected (0.06 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.07 sec)

Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+---------------------+---------+----------+----------+
| date_time           | my_user | old_sal  | new_sal  |
+---------------------+---------+----------+----------+
| 2021-10-17 21:44:41 | yafet@  | 35006.30 | 35016.30 |
+---------------------+---------+----------+----------+
1 row in set (0.00 sec)

mysql> 
```

3, Create a procedure that includes a CASE statement. The code should extract the current Day of the Week from the system time (i.e. Sunday, Monday, etc). Determine the number of characters the day of the week has (i.e. Monday = 6). The CASE statement should then use this variable (i.e. number of characters) to display the mnemonic name (i.e. 6 should display ‘SIX’).

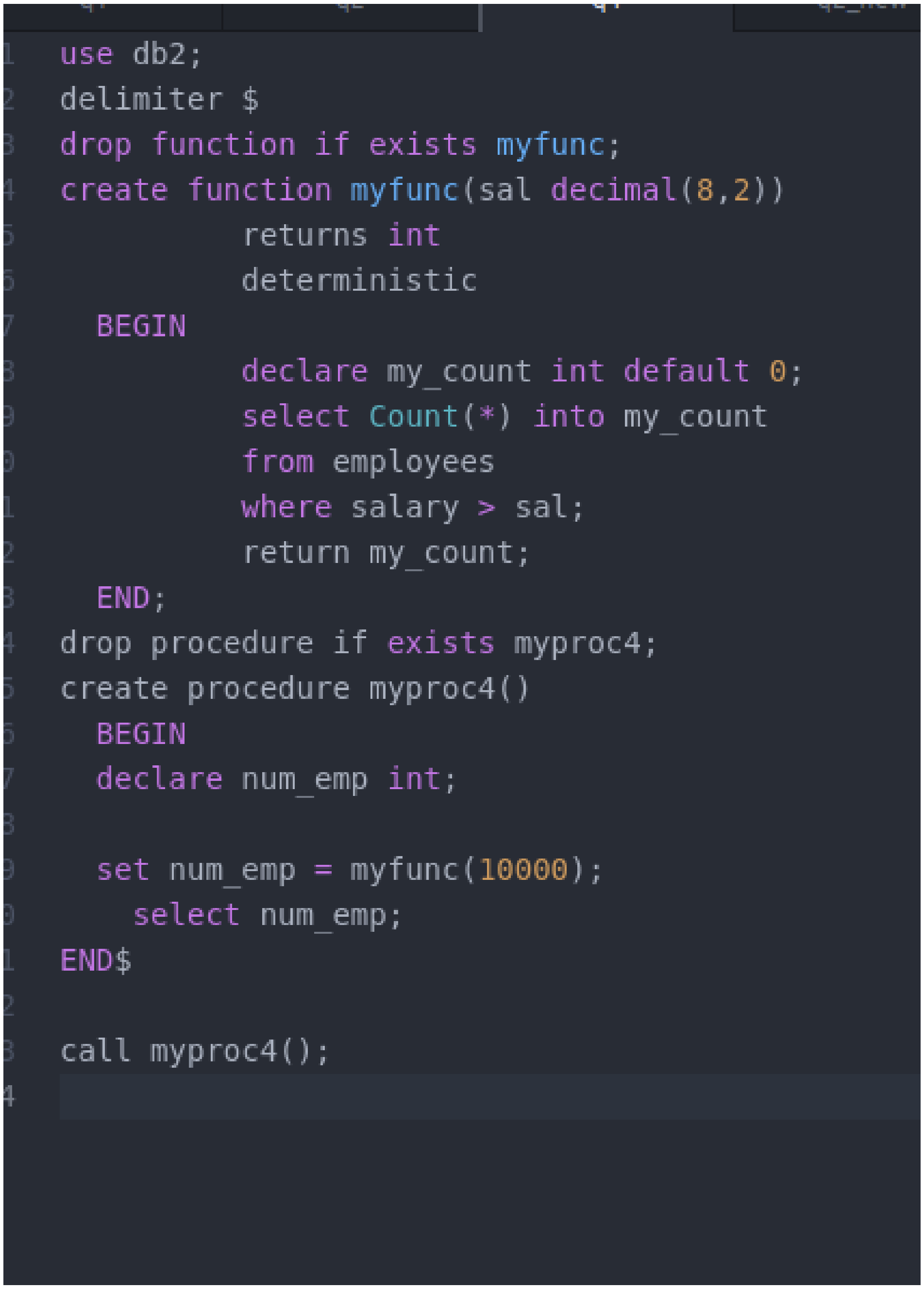


```
use db2;
DELIMITER $
drop procedure if exists myproc3;
create procedure myproc3()
BEGIN

declare day_of_week varchar(20);
declare num int default 0;
select dayname(Current_timestamp) into day_of_week;

select day_of_week;

  set num = CHAR_LENGTH(day_of_week);

  select num;

  CASE num
    when 5 then select 'FIVE';
    when 6 then select 'SIX';
    when 7 then select 'SEVEN';
    when 8 then select 'EIGHT';
    when 9 then select 'NINE';
END CASE;
END$
DELIMITER ;
call myproc3();
```

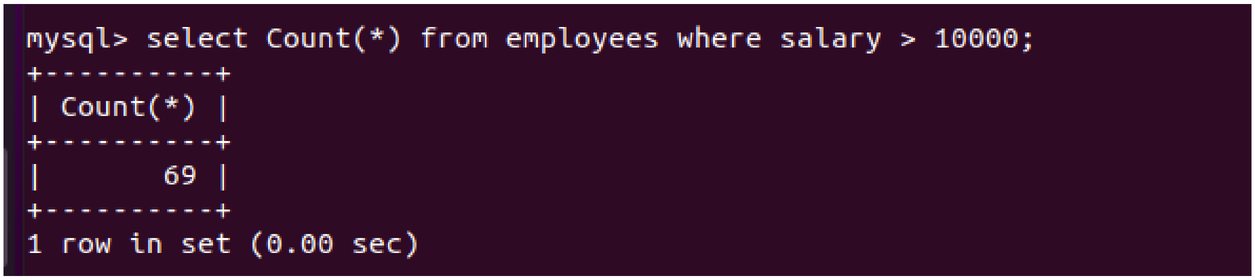


```
mysql>
mysql>
mysql> source q3;
Database changed
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

+-------------+
| day_of_week |
+-------------+
| Tuesday     |
+-------------+
1 row in set (0.00 sec)

+------+
| num  |
+------+
|    7 |
+------+
1 row in set (0.00 sec)

+-------+
| SEVEN |
+-------+
| SEVEN |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

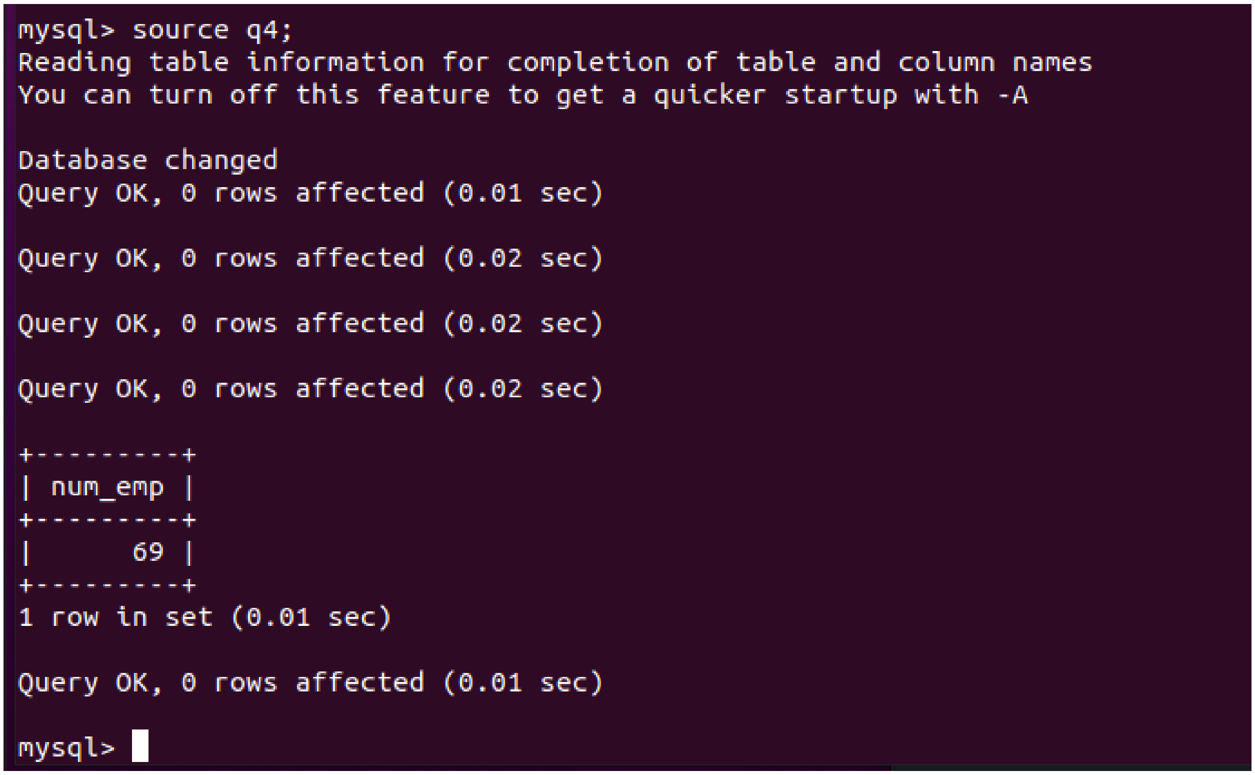
4, Create a procedure that calls a function. The function should have an argument value of the same type as the salary attribute from the EMPLOYEES table. The function should count the number of employees earning more than the salary amount being passed in. The function should return this count which is then displayed from the calling procedure.



```
use db2;
delimiter $
drop function if exists myfunc;
create function myfunc(sal decimal(8,2))
          returns int
          deterministic
  BEGIN
          declare my_count int default 0;
          select Count(*) into my_count
          from employees
          where salary > sal;
          return my_count;
  END;
drop procedure if exists myproc4;
create procedure myproc4()
  BEGIN
  declare num_emp int;

  set num_emp = myfunc(10000);
    select num_emp;
END$

call myproc4();

```



```
mysql> select Count(*) from employees where salary > 10000;
+----------+
| Count(*) |
+----------+
|       69 |
+----------+
1 row in set (0.00 sec)
```



```
mysql> source q4;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

+---------+
| num_emp |
+---------+
|      69 |
+---------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> 
```