EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING
AND INFORMATION TECHNOLOGY
□□□

Practice Assignment – Quarter 4, 2022-2023
Course Name: Database
Course Code: CSE 301
Student's Full Name: Phan Ngoc Hanh Nhi
Student ID: 2131209002

Practice Assignment 8

***Instruction*:**

** Students are allowing to write their answers (like SQL queries, Screen shot of outputs, etc.) in word file (Answer sheet) provided by instructor. After finishing the assignment, students must convert the word file (Answer sheet) into a PDF file. Finally, students upload the file in Moodle.*

a) TRIGGER (when an event happens, do something ex (insert, update, delete) checks data, handles errors, auditing tables)

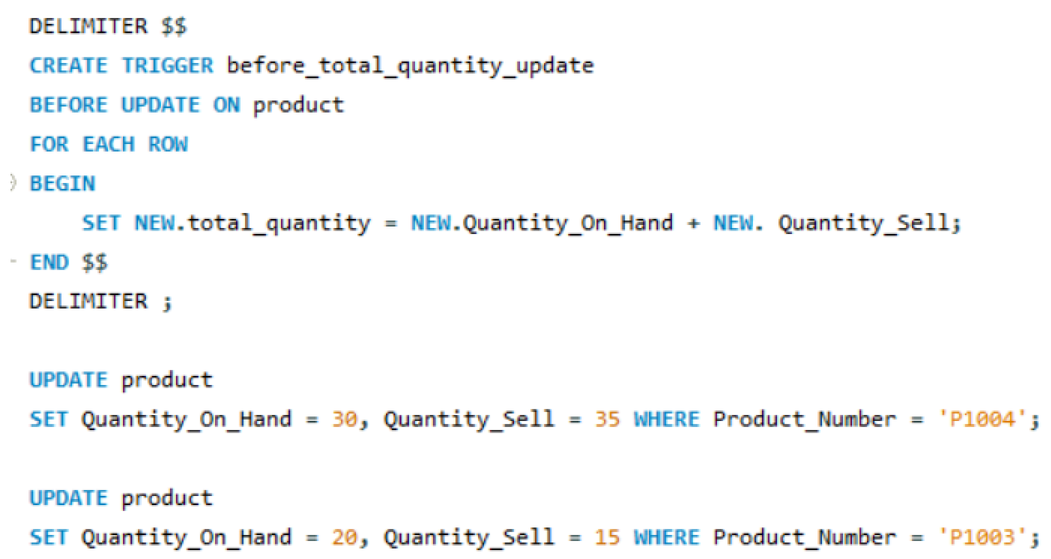
There are six types of row-level triggers in MySQL. They are:

- Before Insert Trigger
- After Insert Trigger
- Before Update Trigger
- After Update Trigger
- Before Delete Trigger
- After Delete Trigger

Syntax:

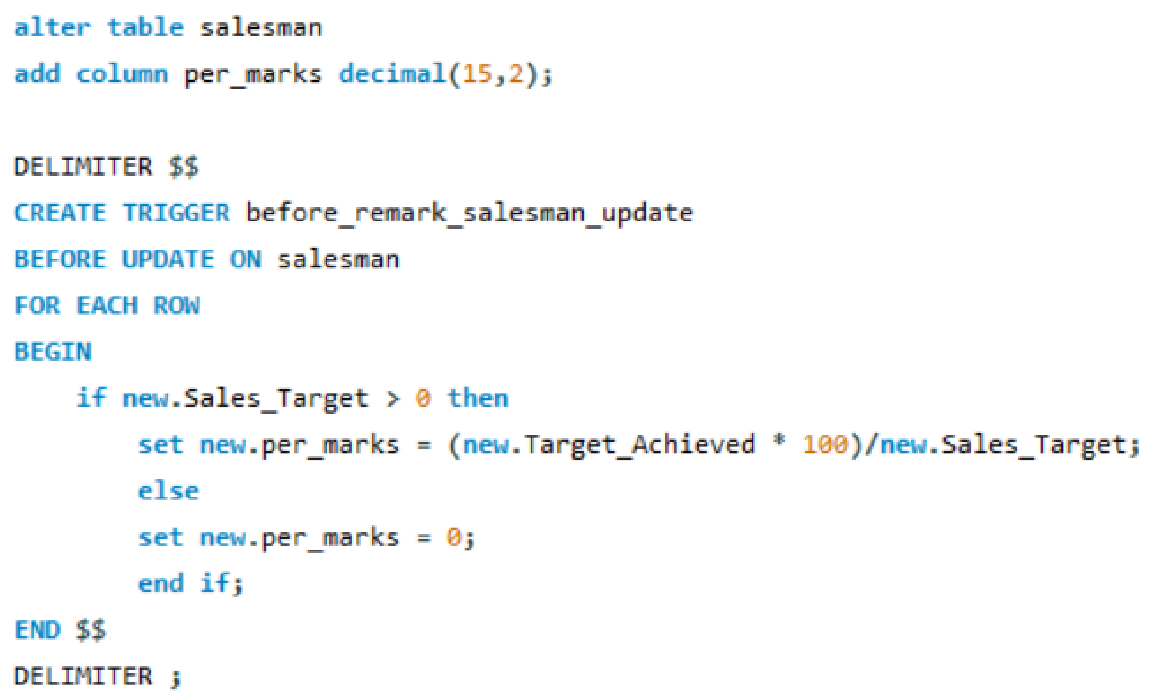
```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
   -- Trigger body (SQL statements)
END;


DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name;
```

1. Create a trigger before_total_quantity_update to update total quantity of product when Quantity_On_Hand and Quantity_sell change values. Then Update total quantity when Product P1004 have Quantity_On_Hand = 30, quantity_sell =35.

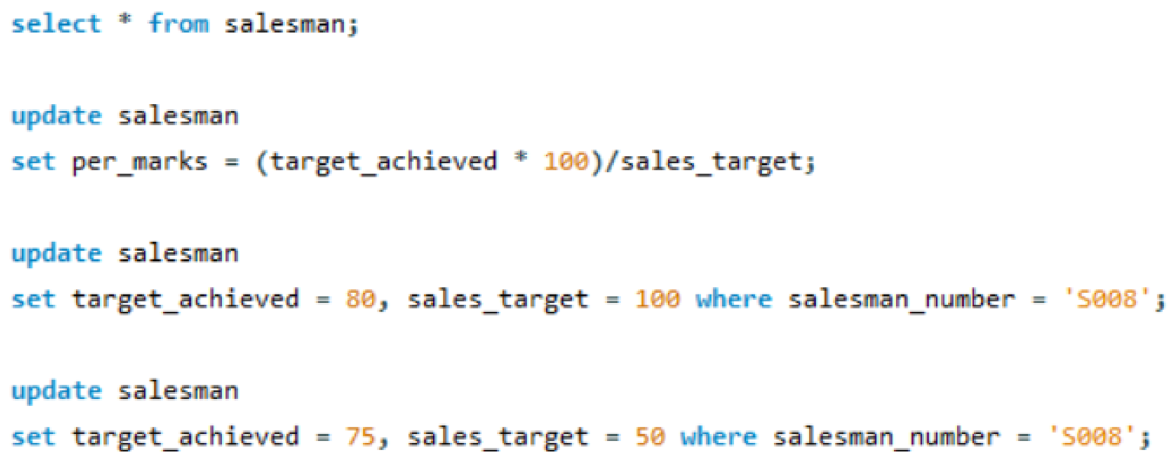


```
DELIMITER $$
CREATE TRIGGER before_total_quantity_update
BEFORE UPDATE ON product
FOR EACH ROW
BEGIN
    SET NEW.total_quantity = NEW.Quantity_On_Hand + NEW. Quantity_Sell;
END $$
DELIMITER ;

UPDATE product
SET Quantity_On_Hand = 30, Quantity_Sell = 35 WHERE Product_Number = 'P1004';

UPDATE product
SET Quantity_On_Hand = 20, Quantity_Sell = 15 WHERE Product_Number = 'P1003';
```

2. Create a trigger before_remark_salesman_update to update Percentage of per_remarks in a salesman table (will be stored in PER_MARKS column) : per_remarks = target_achieved*100/sales_target.



```
alter table salesman
add column per_marks decimal(15,2);

DELIMITER $$
CREATE TRIGGER before_remark_salesman_update
BEFORE UPDATE ON salesman
FOR EACH ROW
BEGIN
    if new.Sales_Target > 0 then
        set new.per_marks = (new.Target_Achieved * 100)/new.Sales_Target;
        else
        set new.per_marks = 0;
        end if;
END $$
DELIMITER ;
```



```
select * from salesman;

update salesman
set per_marks = (target_achieved * 100)/sales_target;

update salesman
set target_achieved = 80, sales_target = 100 where salesman_number = 'S008';

update salesman
set target_achieved = 75, sales_target = 50 where salesman_number = 'S008';
```

3. Create a trigger before_product_insert to insert a product in product table.

```
delimiter $$
create trigger before_product_insert
before insert on product
for each row
begin
    declare profit_percent decimal (5,2);
    set profit_percent = (new.Sell_Price-new.Cost_Price)*100/new.Cost_Price;
    if profit_percent < 30 then
        signal SQLSTATE '45000'
        set message_text = 'Profit percentage must be more than and equal 30%';
    end if;
end $$
delimiter ;

select * from product;
insert into product (Product_Number, Product_Name, Quantity_On_Hand, Quantity_Sell, Sell_Price, Cost_Price)
values ('P1009', 'Printer', 20, 5, 130, 100);

insert into product (Product_Number, Product_Name, Quantity_On_Hand, Quantity_Sell, Sell_Price, Cost_Price)
values ('P1012', 'Scanner', 15, 3, 110, 100);
```

4. Create a trigger to update the delivery status to "Delivered" when an order is marked as "Shipped".

```
delimiter $$
create trigger after_order_shipper
after update on salesorder
for each row
begin
    if new.order_status = 'Successful' then
        update salesorder
        set delivery_status = 'Delivered'
        where order_number = new.order_number;
    end if;
end $$
delimiter ;
```

5. Create a trigger to update the remarks for all salesmen to "Good" when a new salesman is inserted.

```
delimiter $$
create trigger update_remarks
after insert on salesman
for each row
begin
    update salesman
    set remark = 'Good';
end $$
delimiter ;
```

6. Create a trigger to enforce that the first digit of the pin code in the "Clients" table must be 7.

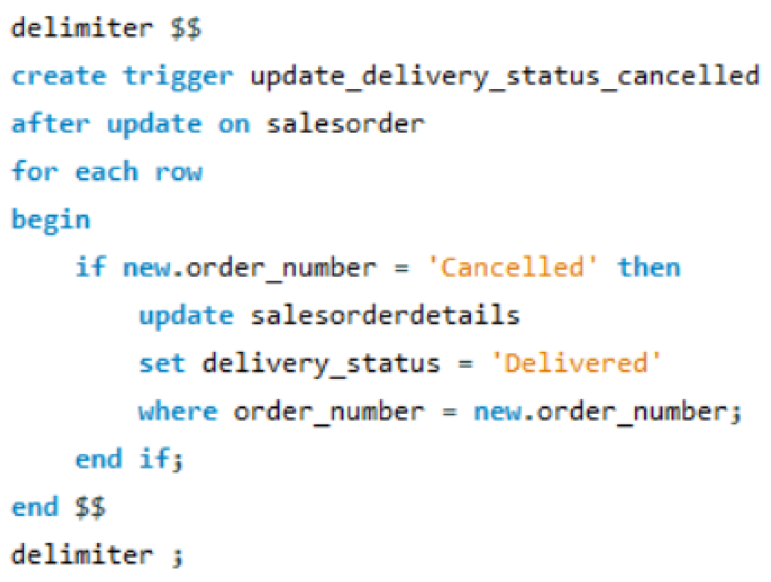
```
delimiter $$
create trigger enforce_pincode
before insert on clients
for each row
begin
    if new.Pincode not like '7%' then
        signal sqlstate '45000'
        set message_text = 'Pincode must be begin with the 7';
    end if;
end $$
delimiter ;
```

7. Create a trigger to update the city for a specific client to "Unknown" when the client is deleted.

```
create table delete_clients(
    Client_Number varchar(30) not null,
    Client_Name varchar(30) not null,
    City varchar(30) default null
);

create table clone_clients(
    Client_Number varchar(10) not null,
    Client_Name varchar(25) not null,
    City varchar(30) default null
);
```

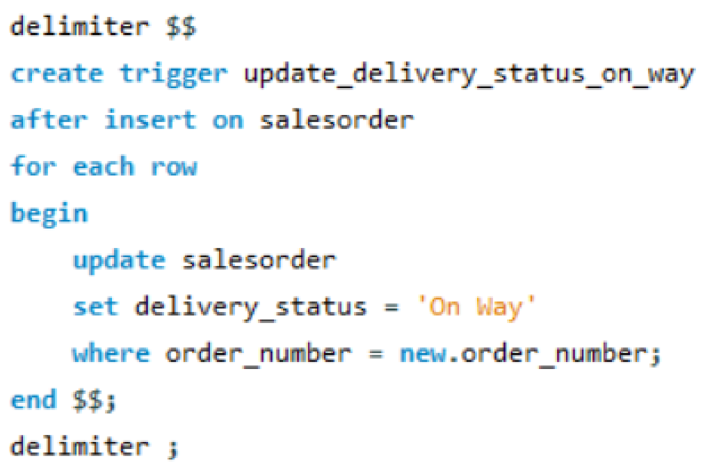
```
delimiter $$
create trigger update_unknown_city
before delete on clone_clients
for each row
begin
    insert into delete_clients
    select clone_clients.Client_Number, clone_clients.Client_Name, clone_clients.City
    from clone_clients;
    update delete_clients
    set city = 'Unknown' where city not like 'Unknown';
end $$
delimiter ;
```

8. Create a trigger to update the delivery status to "Canceled" for corresponding order details when an order is canceled.



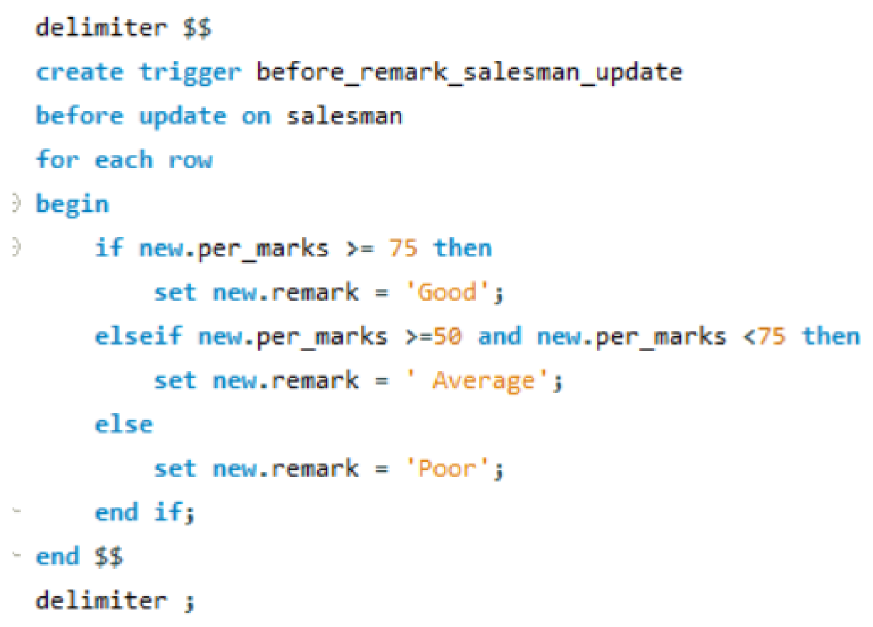
```
delimiter $$
create trigger update_delivery_status_cancelled
after update on salesorder
for each row
begin
    if new.order_number = 'Cancelled' then
        update salesorderdetails
        set delivery_status = 'Delivered'
        where order_number = new.order_number;
    end if;
end $$
delimiter ;
```

9. Create a trigger to update the delivery status to "Pending" for a specific order when an order is inserted



```
delimiter $$
create trigger update_delivery_status_on_way
after insert on salesorder
for each row
begin
    update salesorder
    set delivery_status = 'On Way'
    where order_number = new.order_number;
end $$;
delimiter ;
```

10. Create a trigger before_remark_salesman_update to update Percentage of per_remarks in a salesman table (will be stored in PER_MARKS column) If per_remarks >= 75%, his remarks should be ‘Good’. If 50% <= per_remarks < 75%, he is labeled as 'Average'. If per_remarks <50%, he is considered 'Poor'.

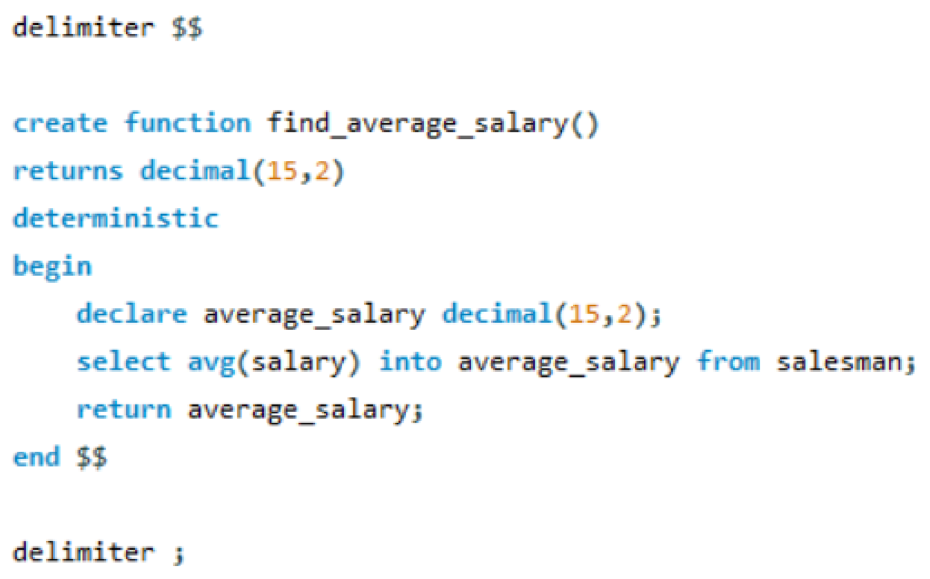

```
delimiter $$
create trigger before_remark_salesman_update
before update on salesman
for each row
begin
    if new.per_marks >= 75 then
        set new.remark = 'Good';
    elseif new.per_marks >=50 and new.per_marks <75 then
        set new.remark = ' Average';
    else
        set new.remark = 'Poor';
    end if;
end $$
delimiter ;
```

1. Writing Function Stored the following tables in a new database ‘Assignment3’:

Hàm đơn trị

CREATE FUNCTION name_function (Var)
RETURNS data_type
DETERMINISTIC
BEGIN
  Statement SQL
  RETURN value
END

1. Find the average salesman’s salary.


```
delimiter $$

create function find_average_salary()
returns decimal(15,2)
deterministic
begin
    declare average_salary decimal(15,2);
    select avg(salary) into average_salary from salesman;
    return average_salary;
end $$

delimiter ;
```

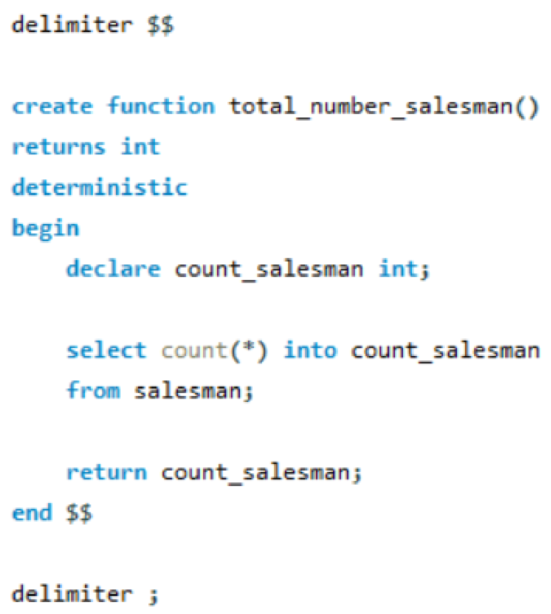
2. Find the name of the highest paid salesman.

```
delimiter $$

create function find_name_of_highest_paid_salesman()
returns varchar(40)
deterministic
begin
    declare highest_paid_salesman varchar(40);

    select salesman_name into highest_paid_salesman
    from salesman
    where salary = (select max(salary) from salesman);

    return highest_paid_salesman;
end $$

delimiter ;
```

3. Find the name of the salesman who is paid the lowest salary.

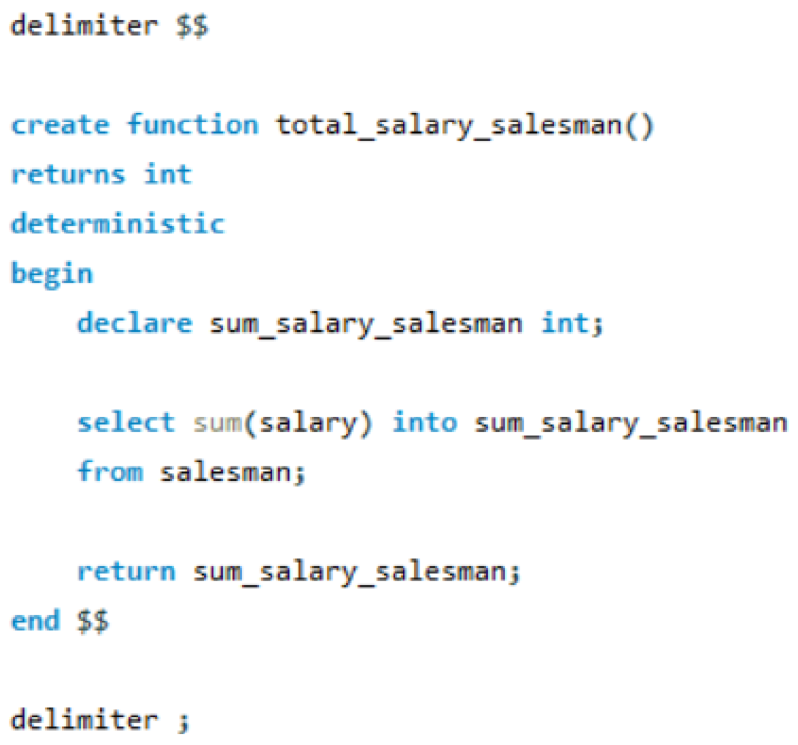
```
delimiter $$

create function find_name_of_lowest_paid_salesman()
returns varchar(40)
deterministic
begin
    declare lowest_paid_salesman varchar(40);

    select salesman_name into lowest_paid_salesman
    from salesman
    where salary = (select min(salary) from salesman);

    return lowest_paid_salesman;
end $$

delimiter ;
```

4. Determine the total number of salespeople employed by the company.



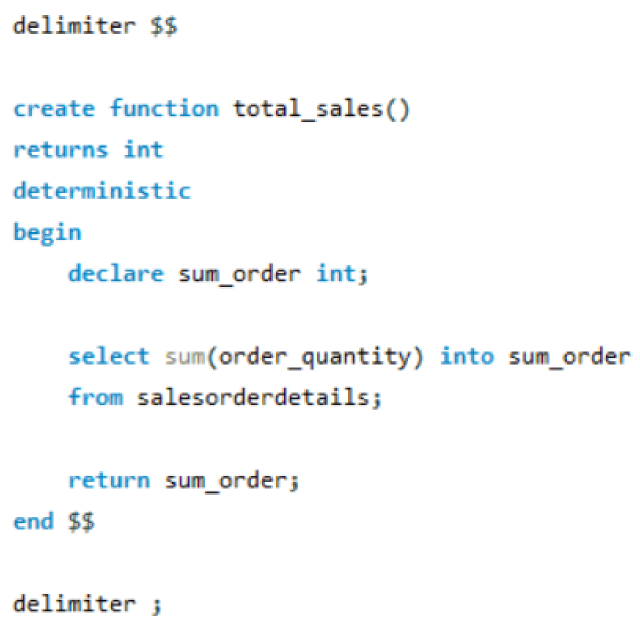
```
delimiter $$

create function total_number_salesman()
returns int
deterministic
begin
    declare count_salesman int;

    select count(*) into count_salesman
    from salesman;

    return count_salesman;
end $$

delimiter ;
```

5. Compute the total salary paid to the company's salesman.



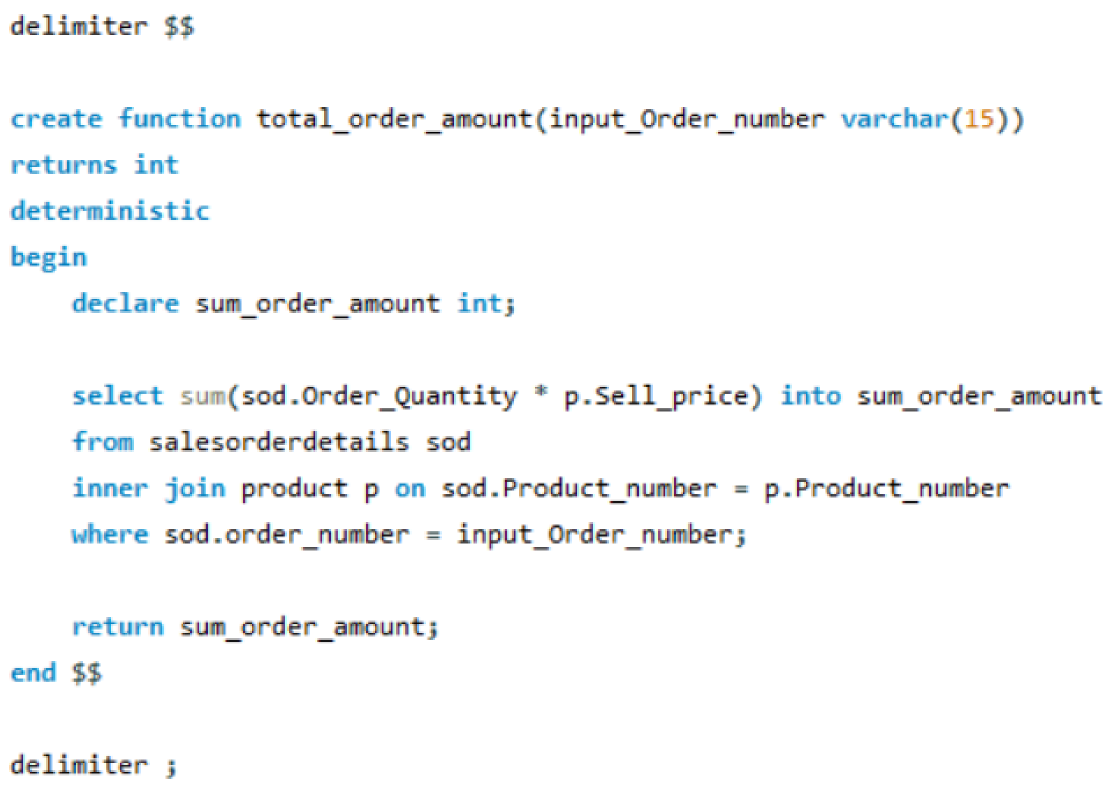
```
delimiter $$

create function total_salary_salesman()
returns int
deterministic
begin
    declare sum_salary_salesman int;

    select sum(salary) into sum_salary_salesman
    from salesman;

    return sum_salary_salesman;
end $$

delimiter ;
```

6. Find Clients in a Province
7. Calculate Total Sales



```
delimiter $$

create function total_sales()
returns int
deterministic
begin
    declare sum_order int;

    select sum(order_quantity) into sum_order
    from salesorderdetails;

    return sum_order;
end $$

delimiter ;
```

8. Calculate Total Order Amount



```
delimiter $$

create function total_order_amount(input_Order_number varchar(15))
returns int
deterministic
begin
    declare sum_order_amount int;

    select sum(sod.Order_Quantity * p.Sell_price) into sum_order_amount
    from salesorderdetails sod
    inner join product p on sod.Product_number = p.Product_number
    where sod.order_number = input_Order_number;

    return sum_order_amount;
end $$

delimiter ;
```