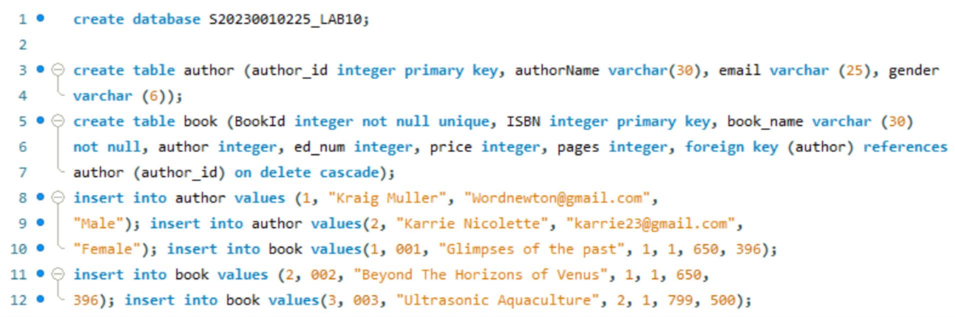
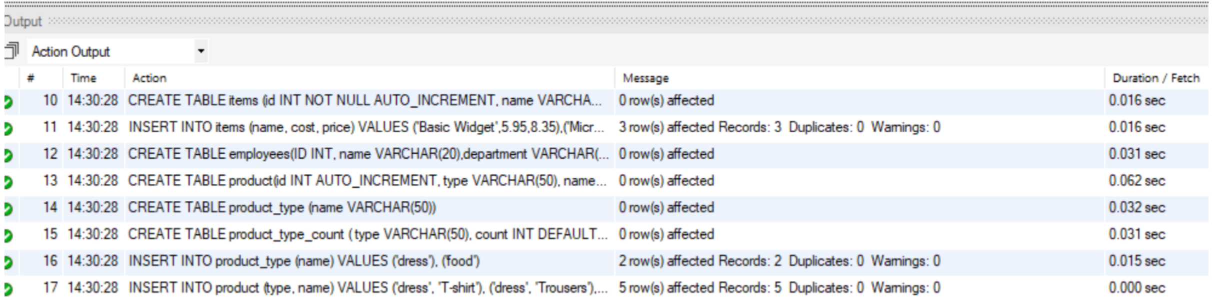
S20230010225

DBMS LAB 10 Screenshots and code

Creating database and tables with data



```
create database S20230010225_LAB10;

create table author (author_id integer primary key, authorName varchar(30), email varchar (25), gender
varchar (6));
create table book (BookId integer not null unique, ISBN integer primary key, book_name varchar (30)
not null, author integer, ed_num integer, price integer, pages integer, foreign key (author) references
author (author_id) on delete cascade);
insert into author values (1, "Kraig Muller", "Wordnewton@gmail.com",
"Male"); insert into author values(2, "Karrie Nicolette", "karrie23@gmail.com",
"Female"); insert into book values(1, 001, "Glimpses of the past", 1, 1, 650, 396);
insert into book values (2, 002, "Beyond The Horizons of Venus", 1, 1, 650,
396); insert into book values(3, 003, "Ultrasonic Aquaculture", 2, 1, 799, 500);
```

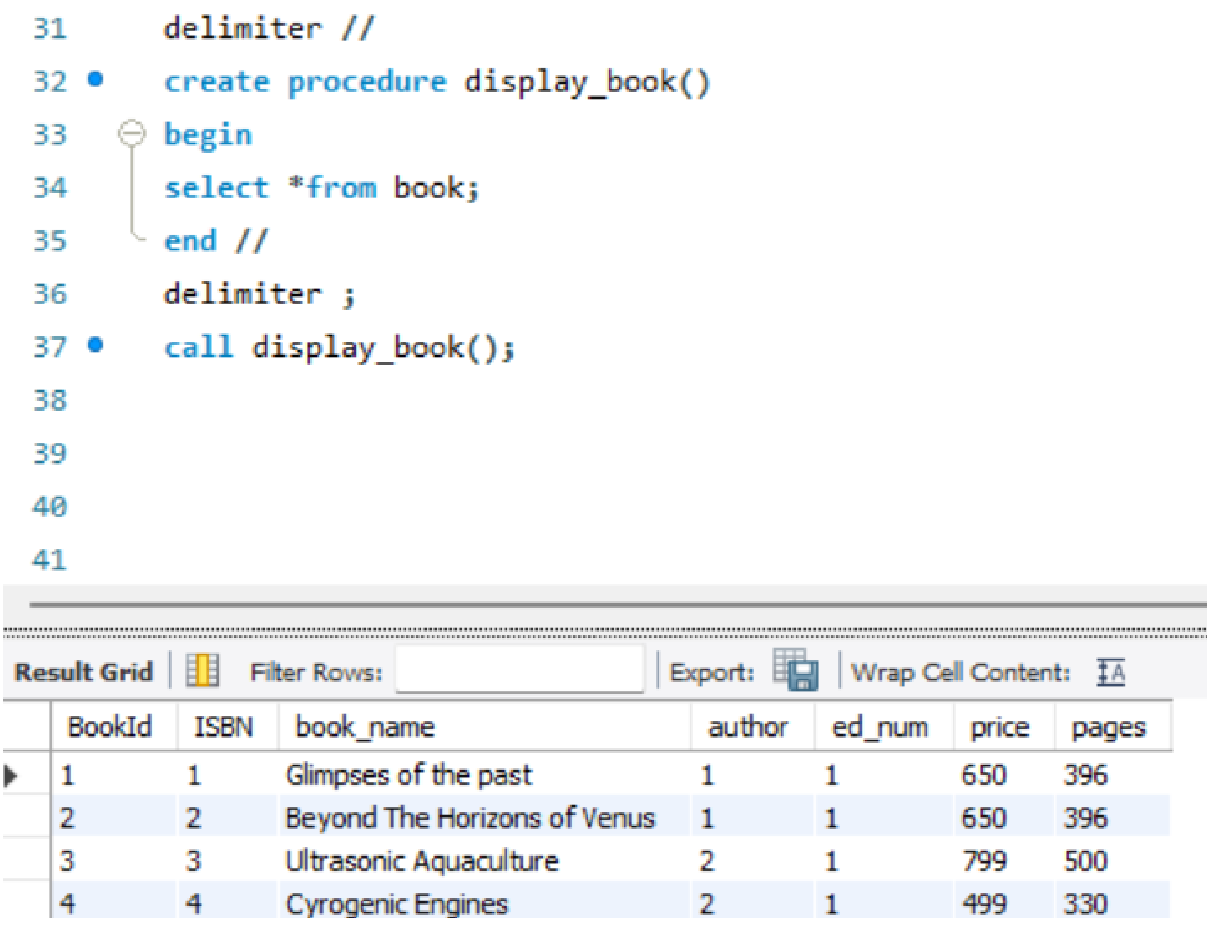


Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	14:30:28	CREATE TABLE items (id INT NOT NULL AUTO_INCREMENT, name VARCHA...	0 row(s) affected	0.016 sec
11	14:30:28	INSERT INTO items (name, cost, price) VALUES ('Basic Widget',5.95,8.35),('Micr...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
12	14:30:28	CREATE TABLE employees(ID INT, name VARCHAR(20),department VARCHAR(...	0 row(s) affected	0.031 sec
13	14:30:28	CREATE TABLE product(id INT AUTO_INCREMENT, type VARCHAR(50), name...	0 row(s) affected	0.062 sec
14	14:30:28	CREATE TABLE product_type (name VARCHAR(50))	0 row(s) affected	0.032 sec
15	14:30:28	CREATE TABLE product_type_count (type VARCHAR(50), count INT DEFAULT...	0 row(s) affected	0.031 sec
16	14:30:28	INSERT INTO product_type (name) VALUES ('dress'), ('food')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.015 sec
17	14:30:28	INSERT INTO product (type, name) VALUES ('dress', 'T-shirt'), ('dress', 'Trousers'),...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec

Example 1:

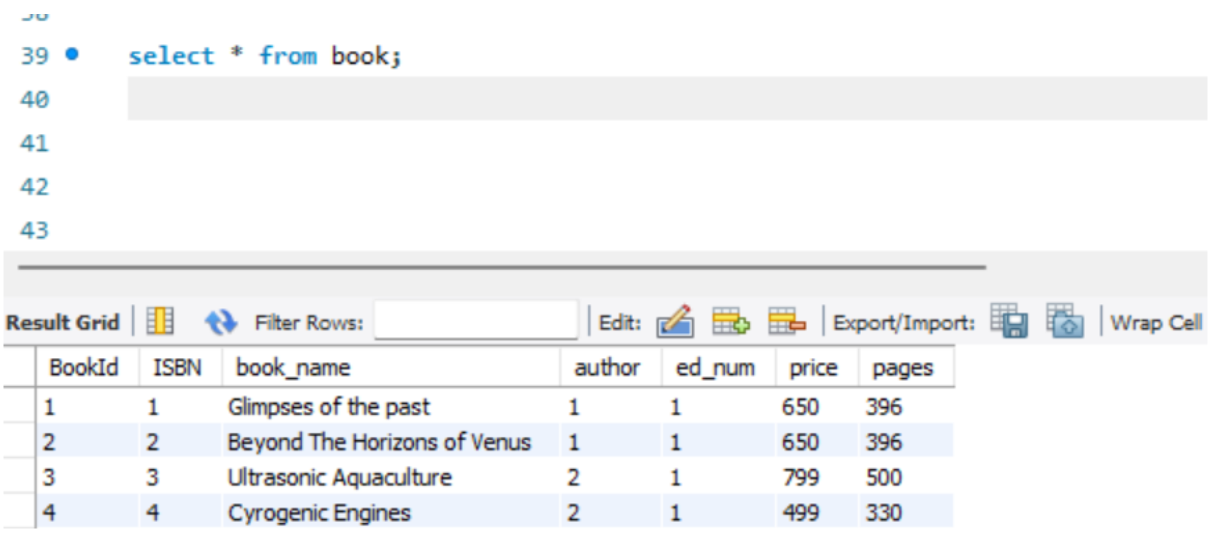


```
delimiter //
create procedure display_book()
begin
select *from book;
end //
delimiter ;
call display_book();
```

BookId	ISBN	book_name	author	ed_num	price	pages
1	1	Glimpses of the past	1	1	650	396
2	2	Beyond The Horizons of Venus	1	1	650	396
3	3	Ultrasonic Aquaculture	2	1	799	500
4	4	Cyrogenic Engines	2	1	499	330

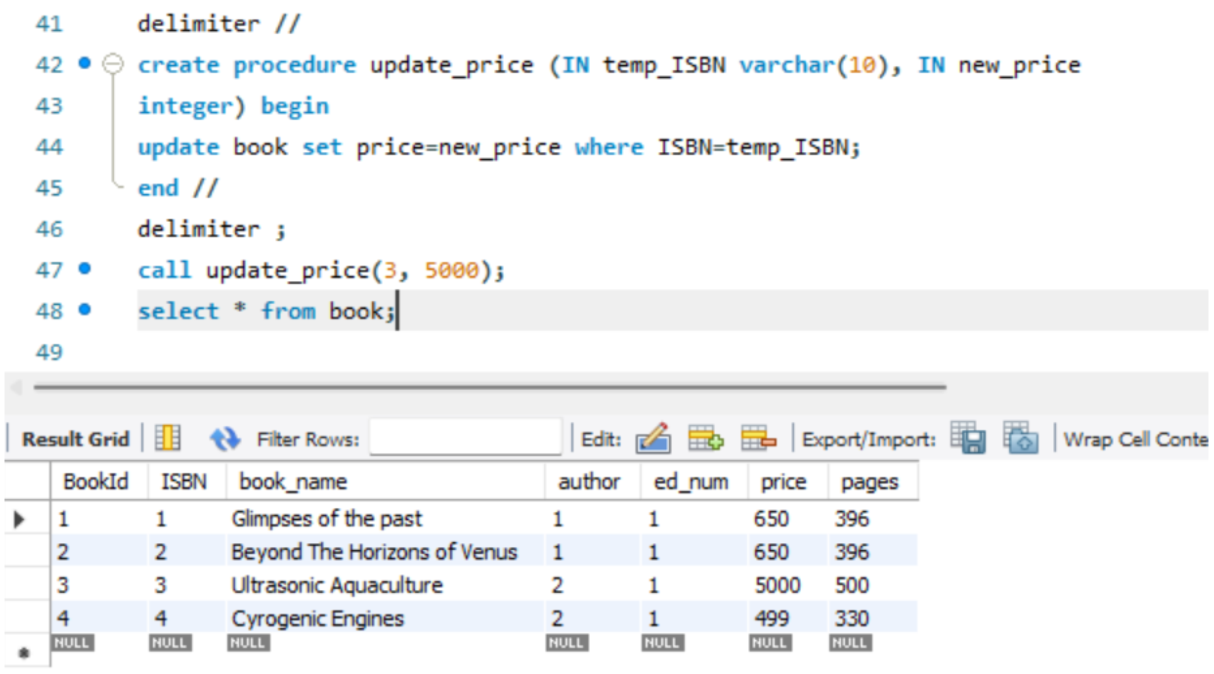
Example 2:

Selecting all from book to see the isbns



```
select * from book;
```

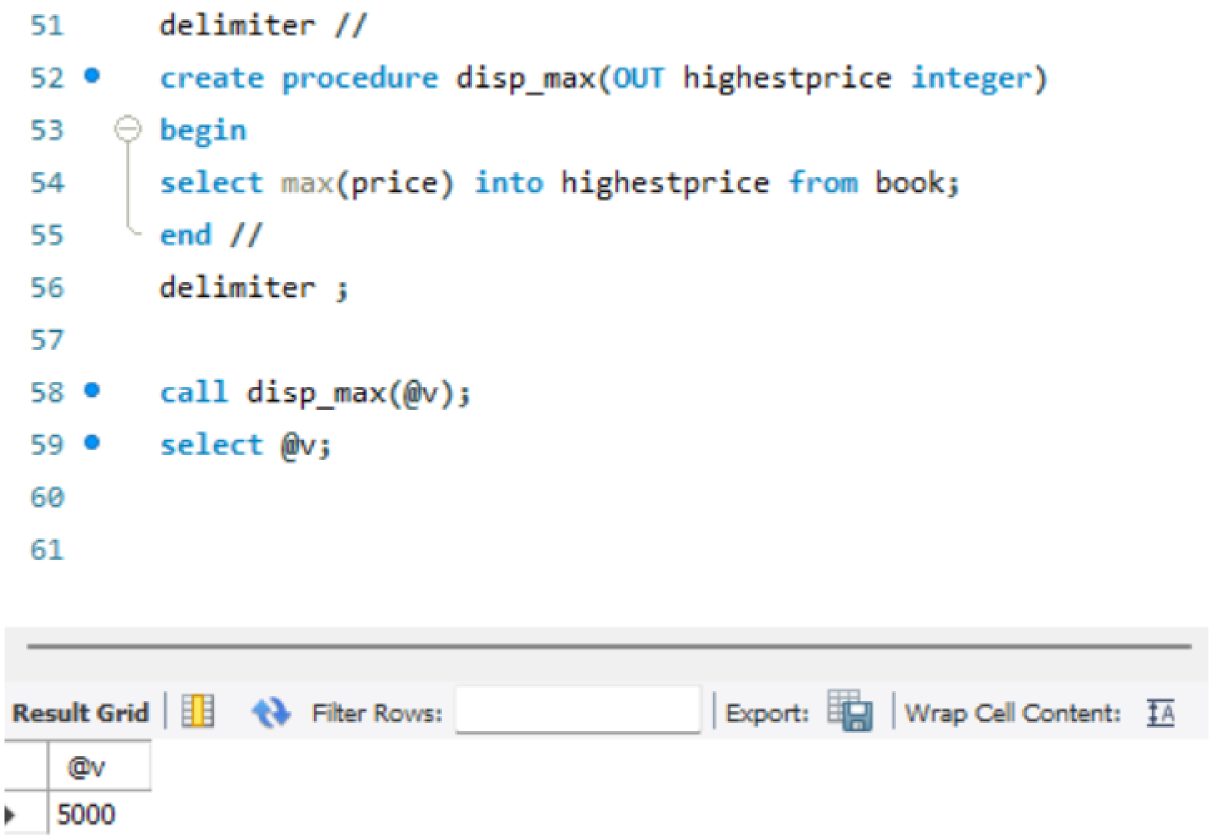
BookId	ISBN	book_name	author	ed_num	price	pages
1	1	Glimpses of the past	1	1	650	396
2	2	Beyond The Horizons of Venus	1	1	650	396
3	3	Ultrasonic Aquaculture	2	1	799	500
4	4	Cyrogenic Engines	2	1	499	330



```sql
delimiter //
create procedure update_price (IN temp_ISBN varchar(10), IN new_price
integer) begin
update book set price=new_price where ISBN=temp_ISBN;
end //
delimiter ;
call update_price(3, 5000);
select * from book;
```

BookId	ISBN	book_name	author	ed_num	price	pages
1	1	Glimpses of the past	1	1	650	396
2	2	Beyond The Horizons of Venus	1	1	650	396
3	3	Ultrasonic Aquaculture	2	1	5000	500
4	4	Cyrogenic Engines	2	1	499	330
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Example 3:

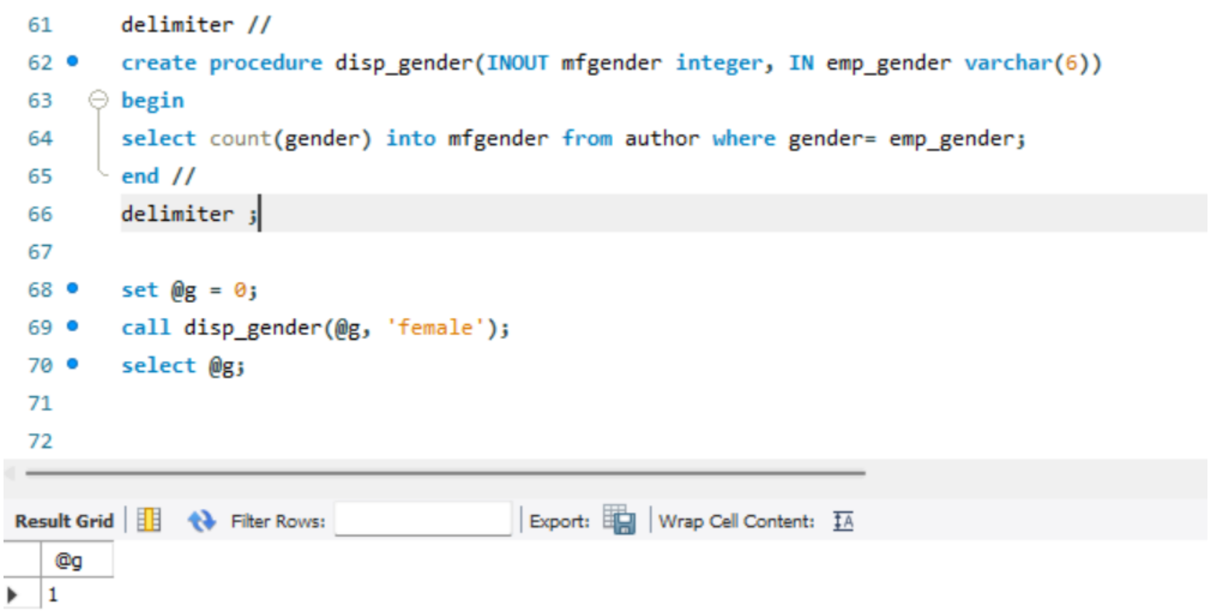


```sql
delimiter //
create procedure disp_max(OUT highestprice integer)
begin
select max(price) into highestprice from book;
end //
delimiter ;

call disp_max(@v);
select @v;
```

@v
5000

Example 4:

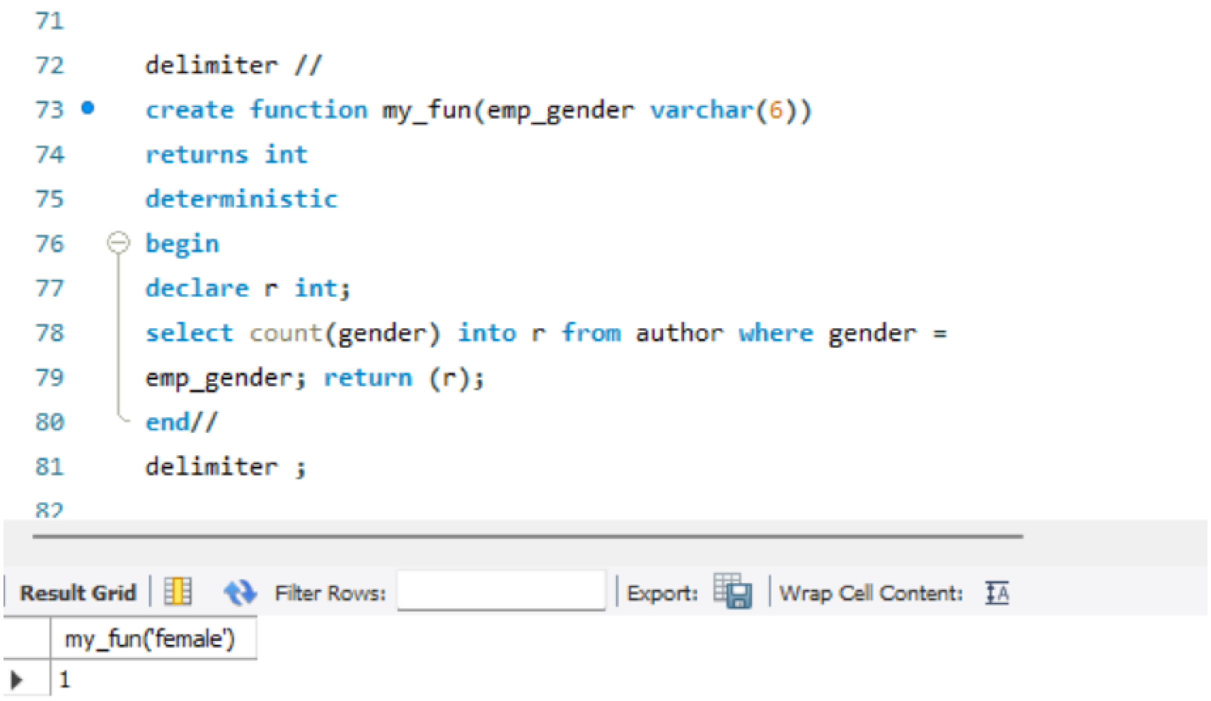
Calling procedure



```
delimiter //
create procedure disp_gender(INOUT mfgender integer, IN emp_gender varchar(6))
begin
select count(gender) into mfgender from author where gender= emp_gender;
end //
delimiter ;

set @g = 0;
call disp_gender(@g, 'female');
select @g;
```

```
@g
1
```

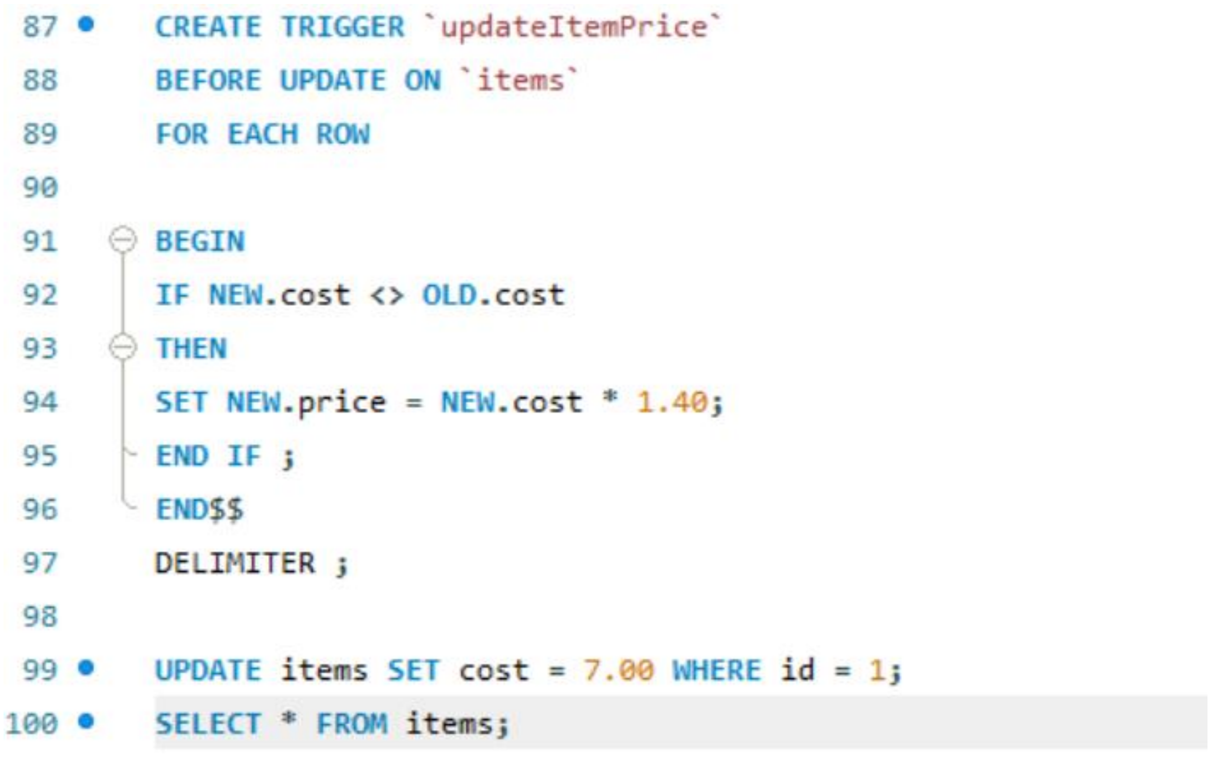
Calling function



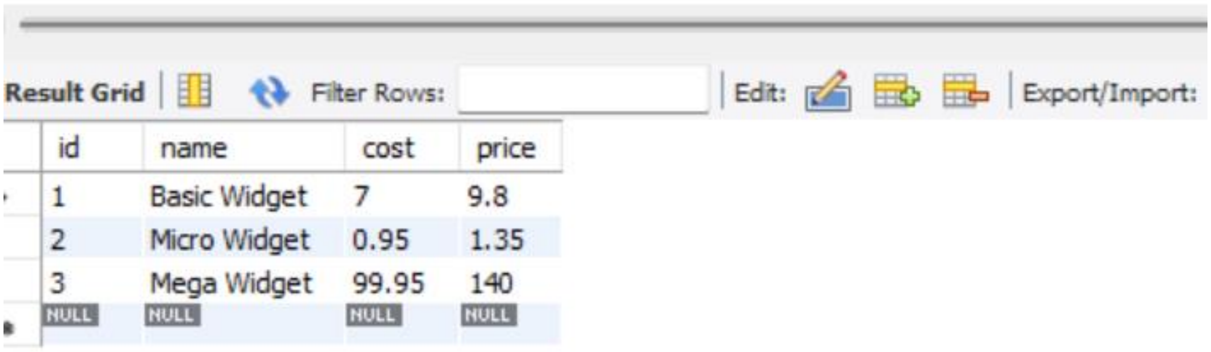
```
delimiter //
create function my_fun(emp_gender varchar(6))
returns int
deterministic
begin
declare r int;
select count(gender) into r from author where gender =
emp_gender; return (r);
end//
delimiter ;
```

```
my_fun('female')
1
```

Triggers example:

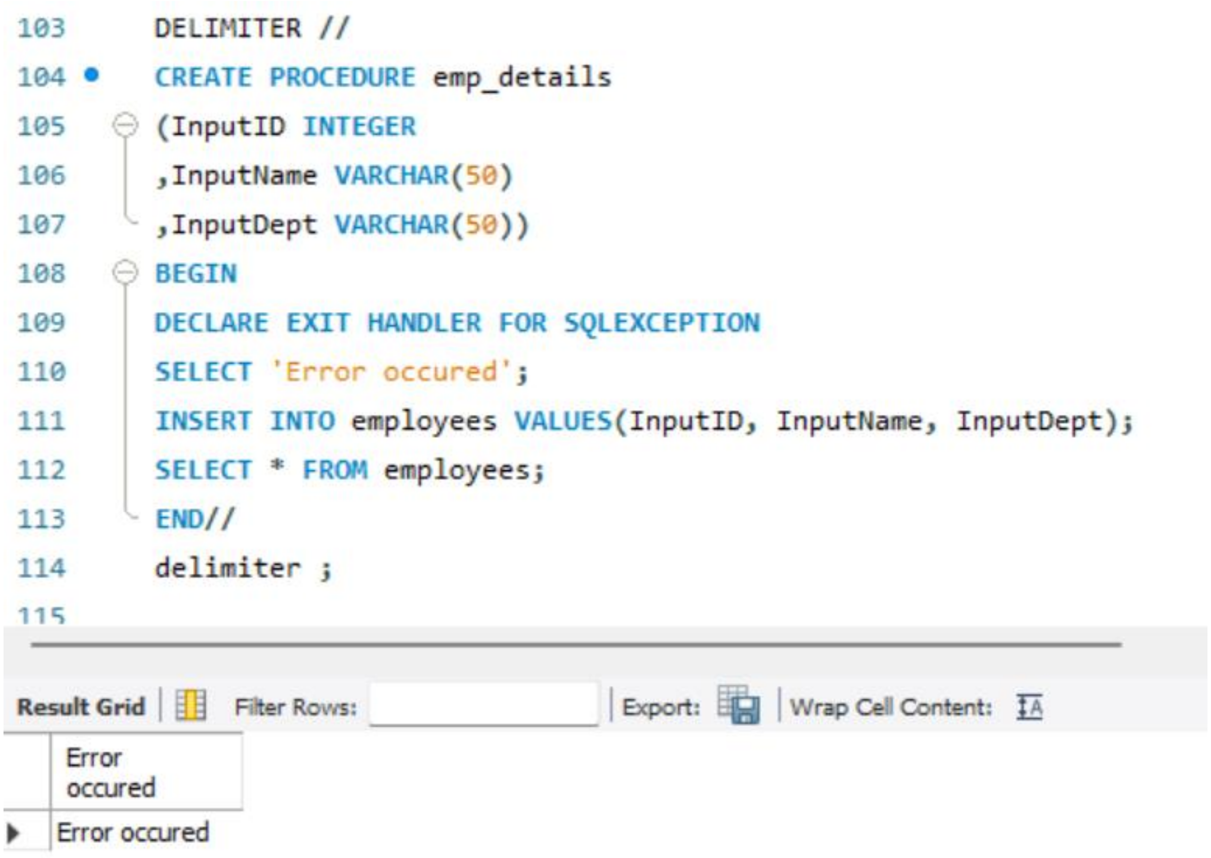


```
CREATE TRIGGER `updateItemPrice`
BEFORE UPDATE ON `items`
FOR EACH ROW

BEGIN
IF NEW.cost <> OLD.cost
THEN
SET NEW.price = NEW.cost * 1.40;
END IF ;
END$$
DELIMITER ;

UPDATE items SET cost = 7.00 WHERE id = 1;
SELECT * FROM items;
```



id	name	cost	price
1	Basic Widget	7	9.8
2	Micro Widget	0.95	1.35
3	Mega Widget	99.95	140
NULL	NULL	NULL	NULL

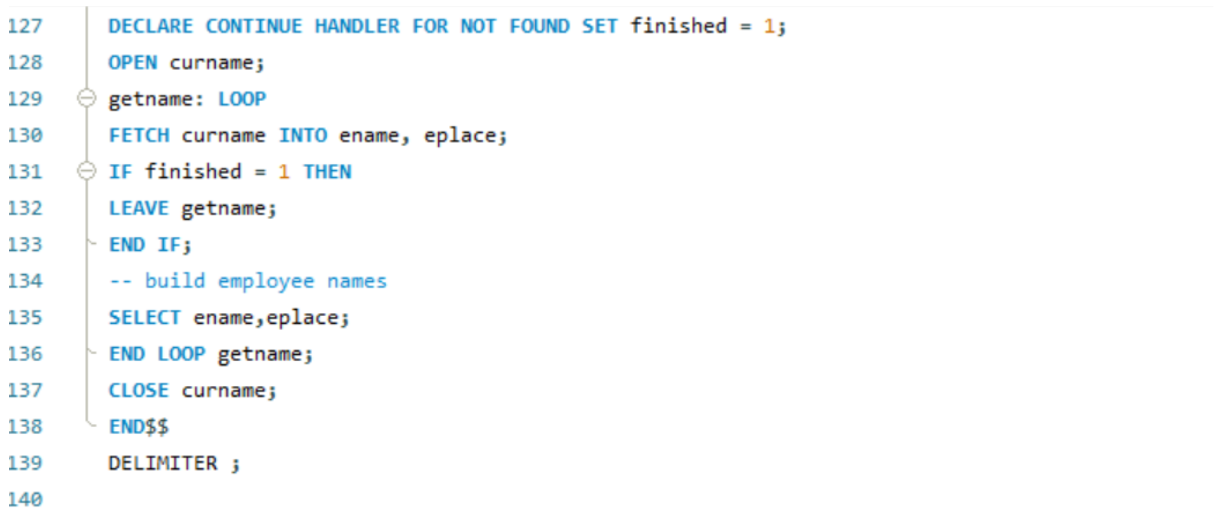
Error handling



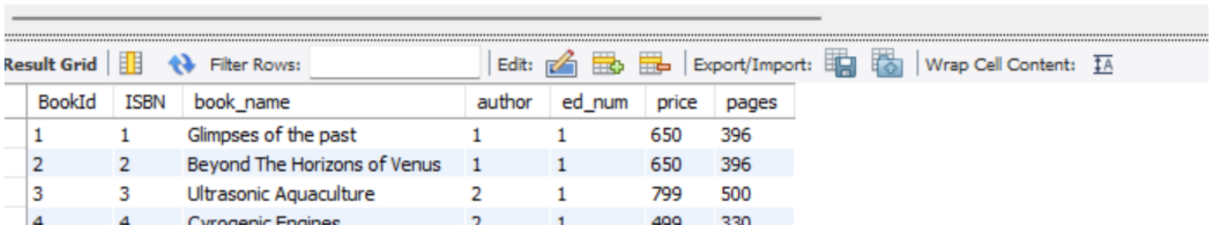
```
DELIMITER //
CREATE PROCEDURE emp_details
(InputID INTEGER
,InputName VARCHAR(50)
,InputDept VARCHAR(50))
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
SELECT 'Error occured';
INSERT INTO employees VALUES(InputID, InputName, InputDept);
SELECT * FROM employees;
END//
delimiter ;
```

Error occured
Error occured

Cursor example 1:

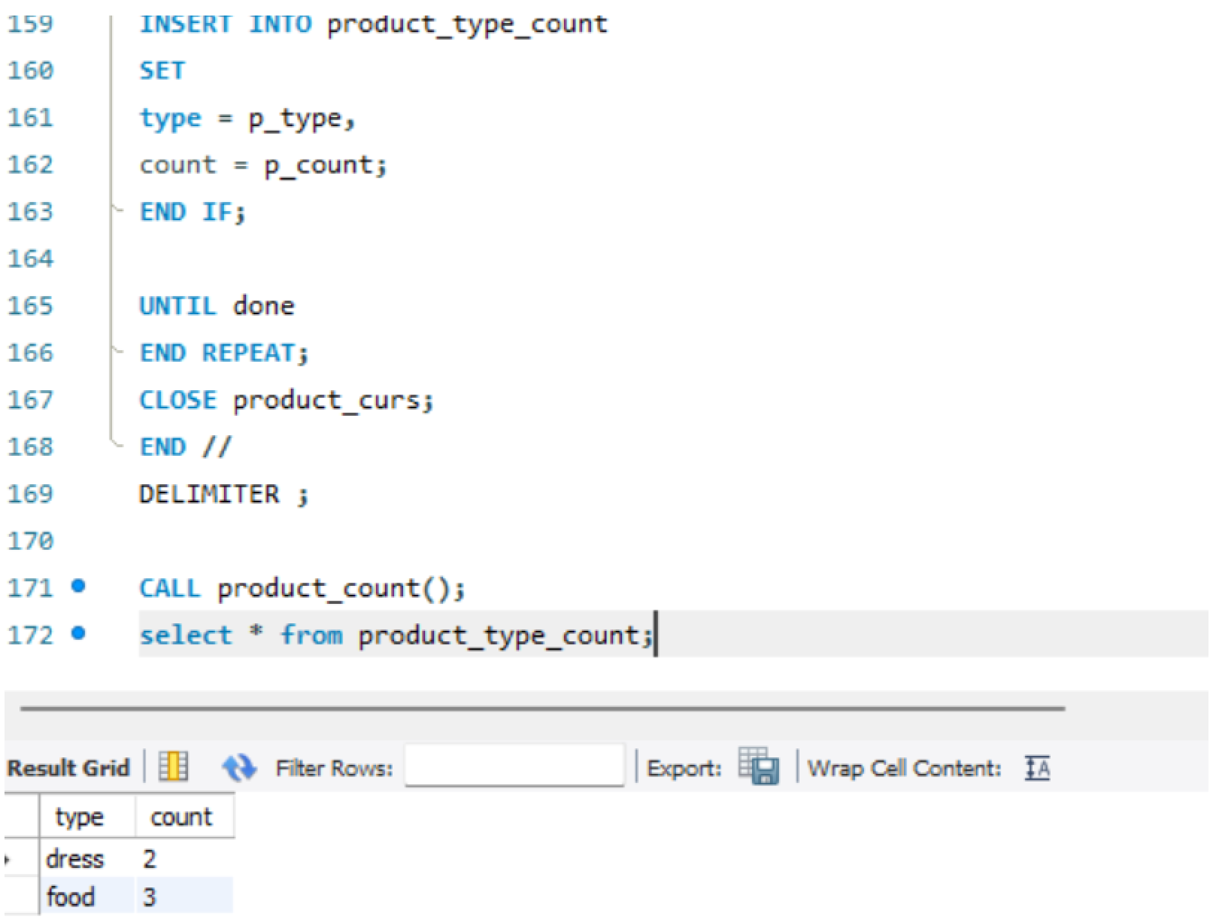


```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
OPEN curname;
getname: LOOP
FETCH curname INTO ename, eplace;
IF finished = 1 THEN
LEAVE getname;
END IF;
-- build employee names
SELECT ename,eplace;
END LOOP getname;
CLOSE curname;
END$$
DELIMITER ;
```



BookId	ISBN	book_name	author	ed_num	price	pages
1	1	Glimpses of the past	1	1	650	396
2	2	Beyond The Horizons of Venus	1	1	650	396
3	3	Ultrasonic Aquaculture	2	1	799	500
4	4	Cyrogenic Engines	2	1	499	330

Cursor example 2:



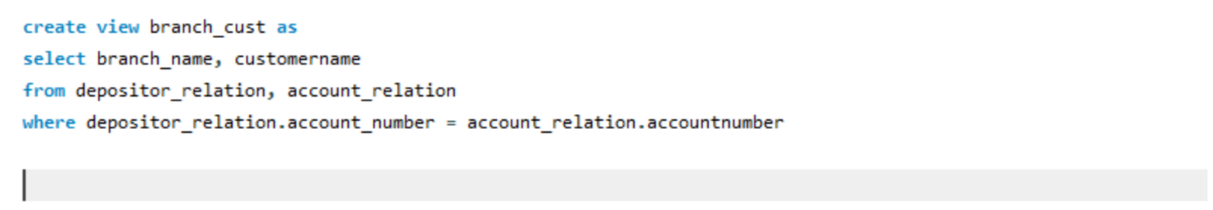
```
INSERT INTO product_type_count
SET
type = p_type,
count = p_count;
END IF;

UNTIL done
END REPEAT;
CLOSE product_curs;
END //
DELIMITER ;

CALL product_count();
select * from product_type_count;
```

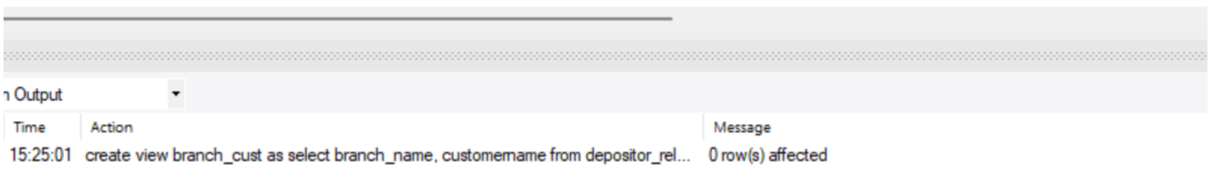
type	count
dress	2
food	3

Actual questions:

Q1:

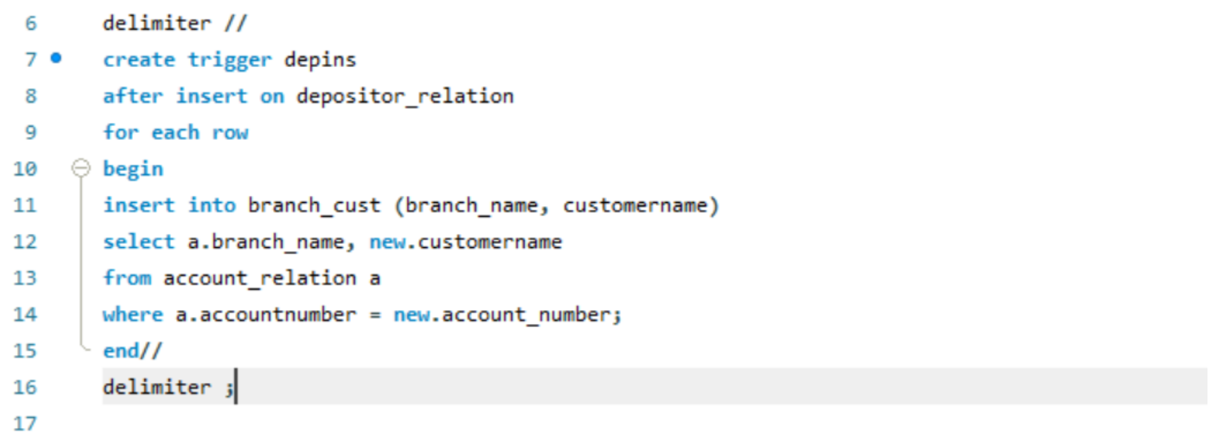
Creating the view

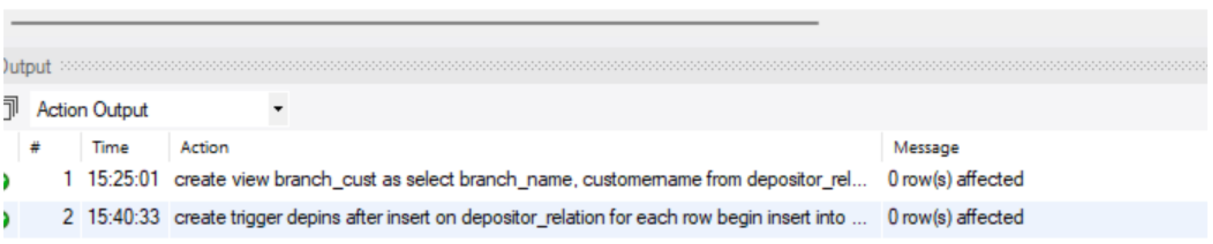

```
create view branch_cust as
select branch_name, customername
from depositor_relation, account_relation
where depositor_relation.account_number = account_relation.accountnumber
```


```
Output
Time       Action                                                                            Message
15:25:01   create view branch_cust as select branch_name, customername from depositor_rel...   0 row(s) affected
```

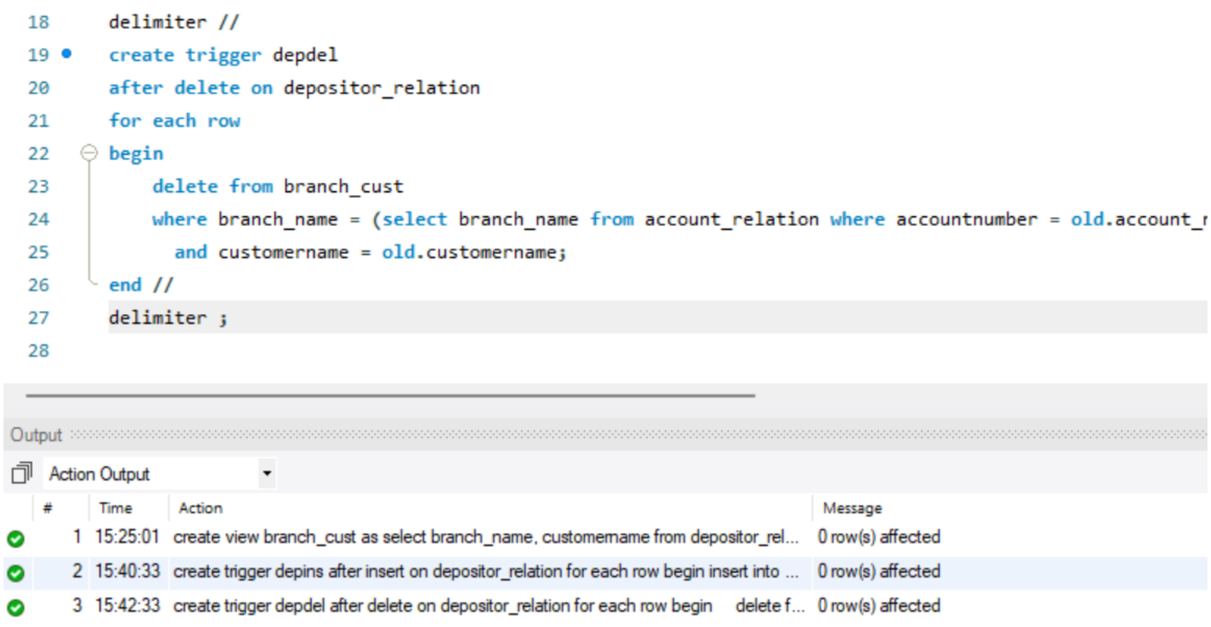
We should create 4 triggers

Trigger for insert into depositor


```
delimiter //
create trigger depins
after insert on depositor_relation
for each row
begin
insert into branch_cust (branch_name, customername)
select a.branch_name, new.customername
from account_relation a
where a.accountnumber = new.account_number;
end//
delimiter ;
```


```
utput
Action Output
#   Time       Action                                                                            Message
1   15:25:01   create view branch_cust as select branch_name, customername from depositor_rel...   0 row(s) affected
2   15:40:33   create trigger depins after insert on depositor_relation for each row begin insert into ...   0 row(s) affected
```

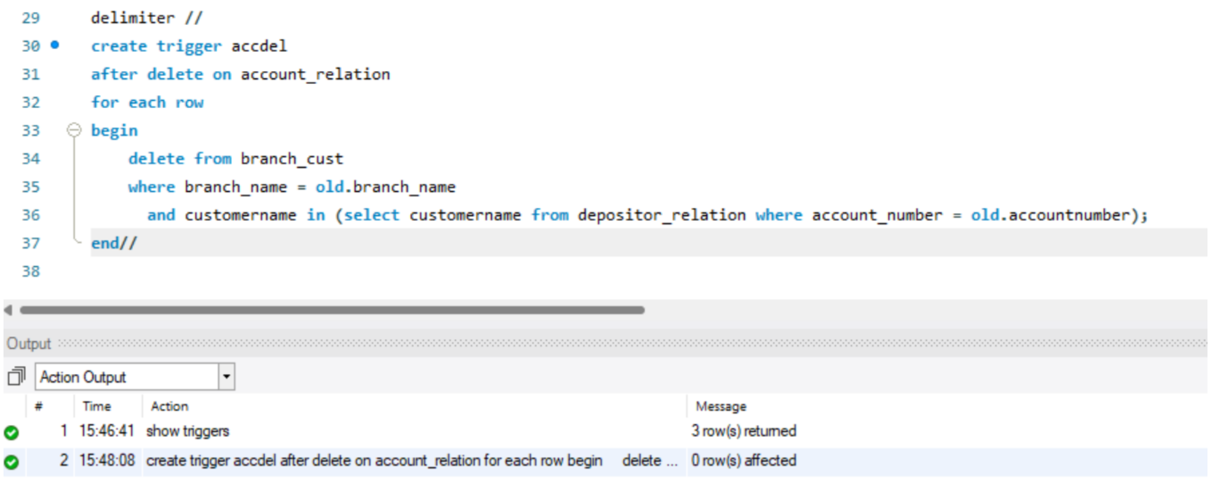
Tigger for deletions from depositor



```
delimiter //
create trigger depdel
after delete on depositor_relation
for each row
begin
    delete from branch_cust
    where branch_name = (select branch_name from account_relation where accountnumber = old.account_
      and customername = old.customername;
end //
delimiter ;
```

```
Output
Action Output
   #   Time      Action                                                                          Message
   1   15:25:01  create view branch_cust as select branch_name, customername from depositor_rel...   0 row(s) affected
   2   15:40:33  create trigger depins after insert on depositor_relation for each row begin insert into ...   0 row(s) affected
   3   15:42:33  create trigger depdel after delete on depositor_relation for each row begin     delete f...   0 row(s) affected
```

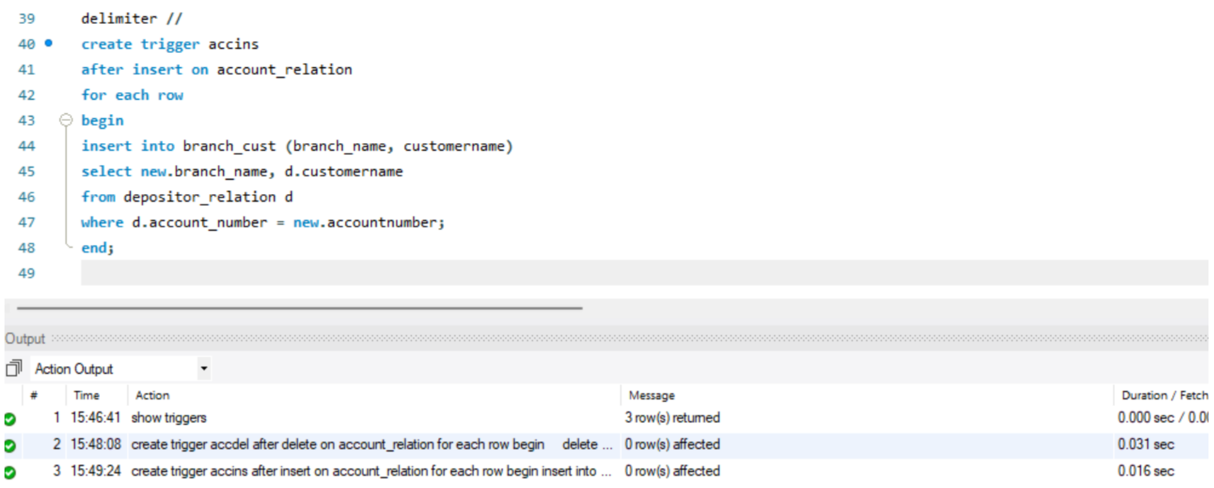
Trigger for deletions on account



```
delimiter //
create trigger accdel
after delete on account_relation
for each row
begin
    delete from branch_cust
    where branch_name = old.branch_name
      and customername in (select customername from depositor_relation where account_number = old.accountnumber);
end//
```

```
Output
Action Output
   #   Time      Action                                                                     Message
   1   15:46:41  show triggers                                                              3 row(s) returned
   2   15:48:08  create trigger accdel after delete on account_relation for each row begin     delete ...   0 row(s) affected
```

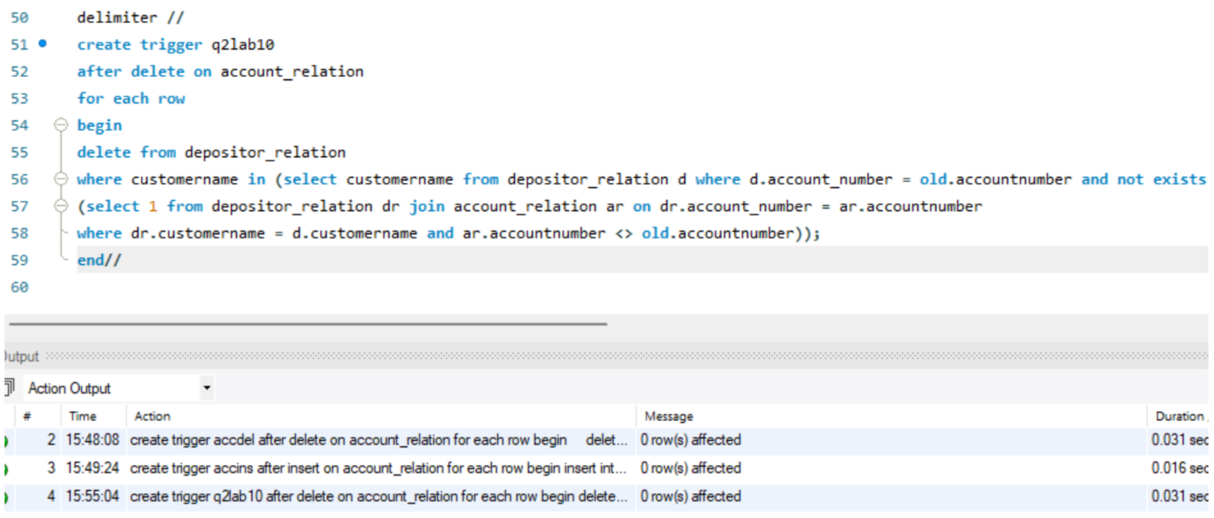
Trigger for insertions on account



```
delimiter //
create trigger accins
after insert on account_relation
for each row
begin
insert into branch_cust (branch_name, customername)
select new.branch_name, d.customername
from depositor_relation d
where d.account_number = new.accountnumber;
end;
```

```
Output
Action Output
#   Time      Action                                                                  Message               Duration / Fetch
1   15:46:41  show triggers                                                           3 row(s) returned     0.000 sec / 0.0
2   15:48:08  create trigger accdel after delete on account_relation for each row begin    delete ...  0 row(s) affected     0.031 sec
3   15:49:24  create trigger accins after insert on account_relation for each row begin insert into ...  0 row(s) affected     0.016 sec
```

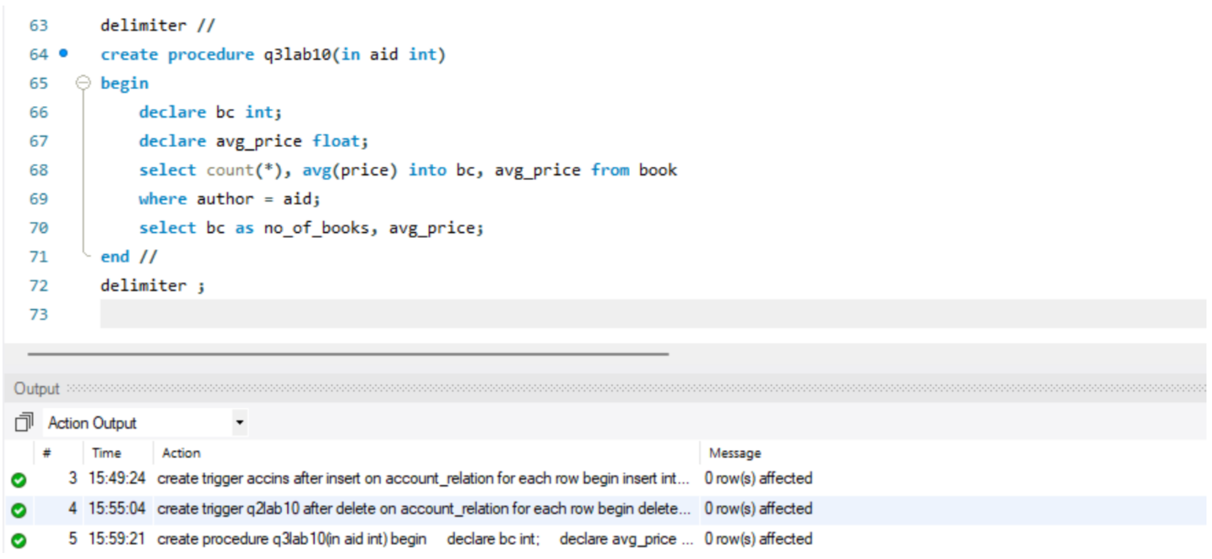
Q2



```
delimiter //
create trigger q2lab10
after delete on account_relation
for each row
begin
delete from depositor_relation
where customername in (select customername from depositor_relation d where d.account_number = old.accountnumber and not exists
(select 1 from depositor_relation dr join account_relation ar on dr.account_number = ar.accountnumber
where dr.customername = d.customername and ar.accountnumber <> old.accountnumber));
end//
```

```
utput
Action Output
#   Time      Action                                                                  Message               Duration
2   15:48:08  create trigger accdel after delete on account_relation for each row begin    delet...  0 row(s) affected     0.031 sec
3   15:49:24  create trigger accins after insert on account_relation for each row begin insert int...  0 row(s) affected     0.016 sec
4   15:55:04  create trigger q2lab10 after delete on account_relation for each row begin delete...  0 row(s) affected     0.031 sec
```

Q3

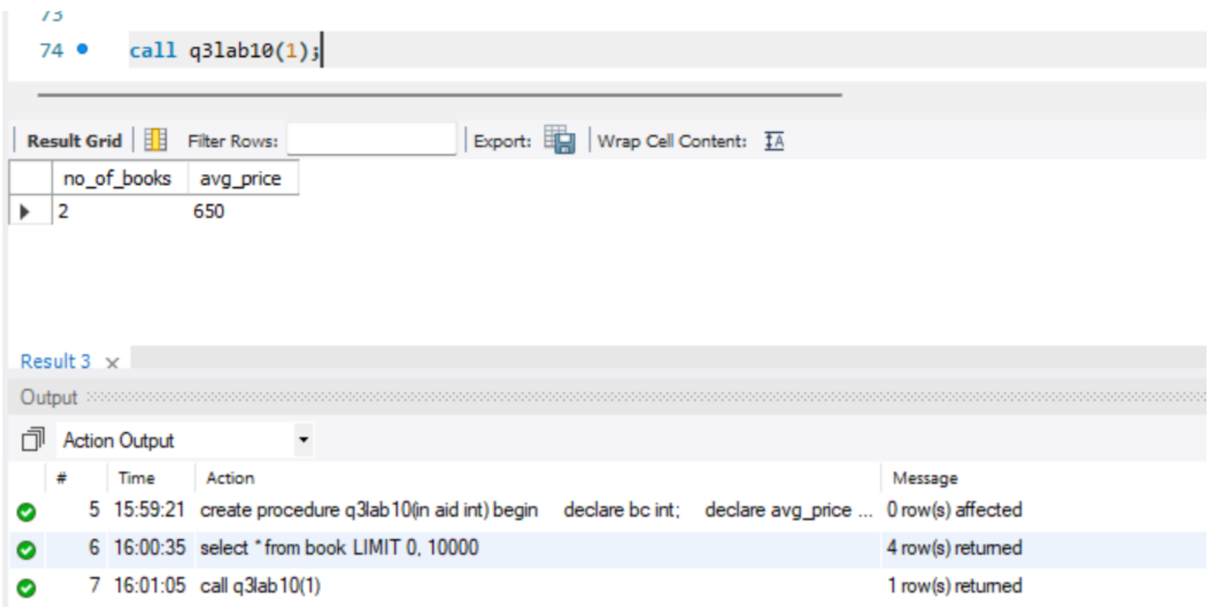


```
delimiter //
create procedure q3lab10(in aid int)
begin
    declare bc int;
    declare avg_price float;
    select count(*), avg(price) into bc, avg_price from book
    where author = aid;
    select bc as no_of_books, avg_price;
end //
delimiter ;
```

Output

Action Output

```
#  Time      Action                                                                        Message
3  15:49:24  create trigger accins after insert on account_relation for each row begin insert int...  0 row(s) affected
4  15:55:04  create trigger q2lab10 after delete on account_relation for each row begin delete...  0 row(s) affected
5  15:59:21  create procedure q3lab10(in aid int) begin    declare bc int;    declare avg_price ...  0 row(s) affected
```



```
call q3lab10(1);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

no_of_books	avg_price
2	650

Result 3

Output

Action Output

```
#  Time      Action                                                                  Message
5  15:59:21  create procedure q3lab10(in aid int) begin    declare bc int;    declare avg_price ...  0 row(s) affected
6  16:00:35  select * from book LIMIT 0, 10000                                       4 row(s) returned
7  16:01:05  call q3lab10(1)                                                         1 row(s) returned
```

Code for questions only(examples is in question anyway):

-- create view branch_cust as

-- select branch_name, customername

-- from depositor_relation, account_relation

-- where depositor_relation.account_number = account_relation.accountnumber

```
-- delimiter //
-- create trigger depins
-- after insert on depositor_relation
-- for each row
-- begin
-- insert into branch_cust (branch_name, customername)
-- select a.branch_name, new.customername
-- from account_relation a
-- where a.accountnumber = new.account_number;
-- end//
-- delimiter ;

-- delimiter //
-- create trigger depdel
-- after delete on depositor_relation
-- for each row
-- begin
--     delete from branch_cust
--     where branch_name = (select branch_name from account_relation where
accountnumber = old.account_number)
--       and customername = old.customername;
-- end //
-- delimiter ;

-- delimiter //
-- create trigger accdel
-- after delete on account_relation
-- for each row
-- begin
--     delete from branch_cust
--     where branch_name = old.branch_name
--       and customername in (select customername from depositor_relation where
account_number = old.accountnumber);
-- end//

-- delimiter //
-- create trigger accins
-- after insert on account_relation
-- for each row
-- begin
-- insert into branch_cust (branch_name, customername)
-- select new.branch_name, d.customername
-- from depositor_relation d
-- where d.account_number = new.accountnumber;
-- end;

-- delimiter //
-- create trigger q2lab10
-- after delete on account_relation
-- for each row
-- begin
-- delete from depositor_relation
-- where customername in (select customername from depositor_relation d where
d.account_number = old.accountnumber and not exists
-- (select 1 from depositor_relation dr join account_relation ar on dr.account_number
= ar.accountnumber
-- where dr.customername = d.customername and ar.accountnumber <>
old.accountnumber));
```

```
-- end//
-- delimiter ;



-- delimiter //
-- create procedure q3lab10(in aid int)
-- begin
--     declare bc int;
--     declare avg_price float;
--     select count(*), avg(price) into bc, avg_price from book
--     where author = aid;
--     select bc as no_of_books, avg_price;
-- end //
-- delimiter ;

-- call q3lab10(1);
```