

# S20230010225

## DBMS LAB 10 Screenshots and code

### Creating database and tables with data

```
1 • create database S20230010225_LAB10;
2
3 • create table author (author_id integer primary key, authorName varchar(30), email varchar (25), gender
4   varchar (6));
5 • create table book (BookId integer not null unique, ISBN integer primary key, book_name varchar (30)
6   not null, author integer, ed_num integer, price integer, pages integer, foreign key (author) references
7   author (author_id) on delete cascade);
8 • insert into author values (1, "Kraig Muller", "Wordnewton@gmail.com",
9   "Male"); insert into author values(2, "Karrie Nicolette", "karrie23@gmail.com",
10  "Female"); insert into book values(1, 001, "Glimpses of the past", 1, 1, 650, 396);
11 • insert into book values (2, 002, "Beyond The Horizons of Venus", 1, 1, 650,
12  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 VARCHAR...	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:

```

31     delimiter //
32 •   create procedure display_book()
33     begin
34     select *from book;
35     end //
36     delimiter ;
37 •   call display_book();
38
39
40
41

```

Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:							
	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 isbnns

```

39 •   select * from book;
40
41
42
43

```

Result Grid     Filter Rows: <input type="text"/>   Edit:      Export/Import:     Wrap Cell							
	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

```

41     delimiter //
42 • create procedure update_price (IN temp_ISBN varchar(10), IN new_price
43     integer) begin
44     update book set price=new_price where ISBN=temp_ISBN;
45     end //
46     delimiter ;
47 • call update_price(3, 5000);
48 • select * from book;
49

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Conte

	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:

```

51     delimiter //
52 • create procedure disp_max(OUT highestprice integer)
53     begin
54     select max(price) into highestprice from book;
55     end //
56     delimiter ;
57
58 • call disp_max(@v);
59 • select @v;
60
61

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
@v			
5000			

#### Example 4:

##### Calling procedure

```
61 delimiter //
62 • create procedure disp_gender(INOUT mfgender integer, IN emp_gender varchar(6))
63 • begin
64 • select count(gender) into mfgender from author where gender= emp_gender;
65 • end //
66 delimiter ;
67
68 • set @g = 0;
69 • call disp_gender(@g, 'female');
70 • select @g;
71
72
```

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

	@g
▶ 1	

##### Calling function




```
71
72 delimiter //
73 • create function my_fun(emp_gender varchar(6))
74 • returns int
75 • deterministic
76 • begin
77 • declare r int;
78 • select count(gender) into r from author where gender =
79 • emp_gender; return (r);
80 • end//
81 delimiter ;
82
```

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

	my_fun('female')
▶ 1	

Triggers example:

```
87 • CREATE TRIGGER `updateItemPrice`
88     BEFORE UPDATE ON `items`
89     FOR EACH ROW
90
91     BEGIN
92     IF NEW.cost <> OLD.cost
93     THEN
94     SET NEW.price = NEW.cost * 1.40;
95     END IF ;
96     END$$
97     DELIMITER ;
98
99 • UPDATE items SET cost = 7.00 WHERE id = 1;
100 • SELECT * FROM items;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:      Export/Import:				
	id	name	cost	price
1	1	Basic Widget	7	9.8
2	2	Micro Widget	0.95	1.35
3	3	Mega Widget	99.95	140
4	NULL	NULL	NULL	NULL

Error handling

```

103 DELIMITER //
104 • CREATE PROCEDURE emp_details
105   (InputID INTEGER
106    ,InputName VARCHAR(50)
107    ,InputDept VARCHAR(50))
108   BEGIN
109     DECLARE EXIT HANDLER FOR SQLEXCEPTION
110     SELECT 'Error occured';
111     INSERT INTO employees VALUES(InputID, InputName, InputDept);
112     SELECT * FROM employees;
113   END//
114 delimiter ;
115

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Error occured			
▶	Error occured			

Cursor example 1:

```

127 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
128 OPEN curname;
129 getname: LOOP
130   FETCH curname INTO ename, eplace;
131   IF finished = 1 THEN
132     LEAVE getname;
133   END IF;
134   -- build employee names
135   SELECT ename,eplace;
136 END LOOP getname;
137 CLOSE curname;
138 END$$
139 DELIMITER ;
140

```

Result Grid							
Filter Rows:							
Edit:							
Export/Import:							
Wrap Cell Content:							
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	400	330	

Cursor example 2:

```

159     INSERT INTO product_type_count
160     SET
161     type = p_type,
162     count = p_count;
163     END IF;
164
165     UNTIL done
166     END REPEAT;
167     CLOSE product_curs;
168     END //
169     DELIMITER ;
170
171 • CALL product_count();
172 • select * from product_type_count;

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	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

```

6 delimiter //
7 • create trigger depins
8 after insert on depositor_relation
9 for each row
10 begin
11 insert into branch_cust (branch_name, customername)
12 select a.branch_name, new.customername
13 from account_relation a
14 where a.accountnumber = new.account_number;
15 end//
16 delimiter ;
17

```

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

## Trigger for deletions from depositor

```
18 delimiter //
```

```
19 • create trigger depdel
```

```
20 after delete on depositor_relation
```

```
21 for each row
```

```
22 begin
```

```
23     delete from branch_cust
```

```
24     where branch_name = (select branch_name from account_relation where accountnumber = old.account_r
```

```
25         and customername = old.customername;
```

```
26 end //
```

```
27 delimiter ;
```

```
28
```

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

```
29 delimiter //
```

```
30 • create trigger accdel
```

```
31 after delete on account_relation
```

```
32 for each row
```

```
33 begin
```

```
34     delete from branch_cust
```

```
35     where branch_name = old.branch_name
```

```
36         and customername in (select customername from depositor_relation where account_number = old.accountnumber);
```

```
37 end//
```

```
38
```

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

```
39 delimiter //
40 • create trigger accins
41 after insert on account_relation
42 for each row
43 begin
44 insert into branch_cust (branch_name, customername)
45 select new.branch_name, d.customername
46 from depositor_relation d
47 where d.account_number = new.accountnumber;
48 end;
49
```

Output

#	Time	Action	Message	Duration / Fetch
1	15:46:41	show triggers	3 row(s) returned	0.000 sec / 0.01
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

```
50 delimiter //
51 • create trigger q2lab10
52 after delete on account_relation
53 for each row
54 begin
55 delete from depositor_relation
56 where customername in (select customername from depositor_relation d where d.account_number = old.accountnumber and not exists
57 (select 1 from depositor_relation dr join account_relation ar on dr.account_number = ar.accountnumber
58 where dr.customername = d.customername and ar.accountnumber <> old.accountnumber));
59 end//
60
```

Output

#	Time	Action	Message	Duration
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 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

```
63 delimiter //
64 • create procedure q3lab10(in aid int)
65 begin
66     declare bc int;
67     declare avg_price float;
68     select count(*), avg(price) into bc, avg_price from book
69     where author = aid;
70     select bc as no_of_books, avg_price;
71 end //
72 delimiter ;
73
```

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

```
74 • call q3lab10(1);
```

Result Grid

	no_of_books	avg_price
▶	2	650

Result 3

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);
```