# **DBMS Project**

# **▼** triggers and procedures

### **▼** Triggers

- add\_user adds new user details to the respective table based on cv\_flag.
- update\_stock → when new transaction is inserted into transactions table.(specific to vendor) or stock is updated.

#### **▼** procedures

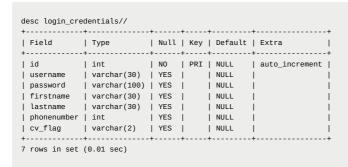
- get\_expenditure gets aggregate of money spent by customer
- get\_budget get customer budget
- show\_stock shows stock of each vendor separately
- enter\_transaction- enters new transaction.
- view\_transaction view transaction filtered by id
- update\_stock updates vendor stock
- view\_profit view vendor profit based on cost price and selling price
- add\_product adds new product to list so stock can have this product as well
- insert\_new\_user inserts new user

# ▼ MySQL commands used

# ▼ MySQL tables

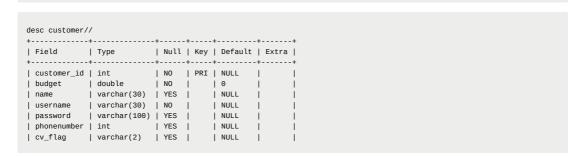
•  $login\_credentials \rightarrow enter data in signup page, check data in <math>login page$ 

create table login\_credentials( id int not null primary key auto\_increment, username varchar(30), password varchar(30), firstname varchar(30), password varchar(30), firstname varchar(30), password v



• customer → all customers

create table customer(customer\_id int not null primary key, budget double not null default 0, name varchar(30), username varchar(30)



```
+-----+
7 rows in set (0.10 sec)
```

## vendor

create table vendor(vendor\_id int not null primary key, name varchar(30), username varchar(30) not null, password varchar(1

Field   Type	Null	Key	I	Default	Extra	Ī
+		+	+-		+	+
vendor_id   int	NO	PRI		NULL	1	1
name   varchar(30)	YES	1	1	NULL	1	1
username   varchar(30)	NO	1	1	NULL	1	1
password   varchar(100)	YES	1	1	NULL	1	1
phonenumber   int	YES	1	Ι	NULL	1	1
cv_flag   varchar(2)	NO	1	1	NULL	1	1

## • product

create table product(product\_id int(11) not null primary key, type varchar(30), name varchar(30), cost\_price double, selling the contract of t

Field								Default		
+	-+		-+		+		+-		+-	+
product_id	1	int	1	NO	1	PRI	I	NULL	1	- 1
type	1	varchar(30)	-	YES	1		I	NULL	1	- 1
name	1	varchar(30)	-	YES	1		I	NULL	1	- 1
cost_price	1	double	-	YES	1		1	NULL	1	- 1
selling_price	1	double	-	YES	1		I	NULL	1	- 1
+	-+		-+		+		+-		+-	+

### stock

create table stock(vendor\_id int,foreign key(vendor\_id) references vendor(vendor\_id),product\_id int,foreign key(product\_id

	+		+		+		+		+-	+
Field	1	Туре	1	Null	1	Key	1	Default	1	Extra
+	+		+		+		+		+-	+
vendor_id	1	int	1	YES	I	MUL	I	NULL	Τ	- 1
product_id	1	int	1	YES	1	MUL	I	NULL	1	- 1
quantity	1	double	1	YES	1		I	NULL	1	- 1
quantity_unit	1	varchar(30)	1	YES	Ι		Τ	NULL	Τ	1

## transaction

create table transaction(transaction\_id int(11) not null primary key, customer\_id int(11), foreign key(customer\_id) reference t\_id), date\_time datetime, quantity\_double, quantity\_unit varchar(30))//

desc transaction									
Field	Туре	•	Nul	ιį	Key	İ	Default	i	Extra
transaction_id   customer_id   vendor_id	int   int   int		NO YES	i	MUL	i	NULL NULL NULL	1	   

# **▼** Procedures and triggers

• add\_user trigger to fill vendor and customer

```
create trigger add_user after insert on login_credentials for each row begin if (new.cv_flag = '0') then insert into custom
```

· view\_transactions

```
create procedure view_transactions(in x integer)
   -> begin
   -> if((select cv_flag from login_credentials where id = x) = '0') then select * from transaction where customer_id = x;
   -> end if;
   -> if((select cv_flag from login_credentials where id = x) = '1') then select * from transaction where vendor_id = x;
   -> end if;
   -> end //
```

· view\_profit():

```
create procedure view_profit(in ven_id integer)
-> begin
-> select sum((product.selling_price - product.cost_price) * transaction.quantity) from transaction join product on product
-> end //
```

• trigger update\_stock

```
create trigger update_stock after insert on transaction for each row
   -> begin
   -> declare t real;
   -> select quantity into t from stock where vendor_id = new.vendor_id and product_id = new.product_id;
   -> update stock set quantity = t - new.quantity where vendor_id = new.vendor_id and product_id = new.product_id;
   -> end//
```

• update stock procedure:

```
create procedure update_stock(in id integer, in pid integer, in qty real)
   -> begin
   -> declare t real;
   -> if exists(select * from stock where vendor_id = id and product_id = pid) then
   -> select quantity into t from stock where vendor_id = id and product_id = pid;
   -> update stock set quantity = t + qty where vendor_id = id and product_id = pid;
   -> else
   -> insert into stock(vendor_id, product_id, quantity) values(id, pid, qty);
   -> end if;
   -> end //
```

delete user

```
create procedure delete_user(in x integer)
  -> begin
  -> delete from login_credentials where id = x;
  -> update vendor set cv_flag = 2 where vendor_id = x;
  -> update customer set cv_flag = 2 where customer_id = x;
  -> end //
```

show vendor stock

```
create procedure show_vendor_stock(in id integer)
  -> begin
  -> select * from stock where vendor_id = id;
  -> end//
```

· add product:

```
create procedure add_product(in id integer, in ty varchar(30), in nm varchar(30), in cp real, in sp real)
   -> begin
   -> insert into product values(id, ty, nm, cp, sp);
   -> end //
```

• insert\_transaction():

```
create procedure enter_transaction(in id integer, in cid integer, in vid integer, in pid integer, in qt real, in qu varchar
-> begin
-> insert into transaction values(id, cid, vid, pid, now(), qt, qu);
-> end //
```

• get\_expenditure():

```
create procedure get_expenditure(in cust_id integer)
-> begin
-> select sum(transaction.quantity * product.selling_price) from transaction join product on product.product_id = transacti
-> end//
```

• get budget():

```
create procedure get_budget(in cust_id integer)
-> begin
-> select budget from customer where customer_id = cust_id;
-> end//
```

show\_stock() → shows vendor stock in different tables \*\*\*\*\*\*\*\*on customer stock page\*\*\*\*\*

```
create procedure show_stock()
begin
declare id integer;
declare fin integer default 0;
declare c cursor for select vendor_id from vendor;
declare continue handler for not found set fin= 1;
open c;
ven:loop
fetch c into id;
if(fin = 1) then leave ven;
end if;
select * from stock where vendor_id = id ;
end loop ven;
close c;
end//
```

• insert\_new\_user signup page

```
create procedure insert_new_user(uname varchar(30), pass varchar(100), fname varchar(30), lname varchar(30), pnum int(1 begin insert into login_credentials values (DEFAULT,uname,pass,fname,lname,pnum,cv_flag); end//
```