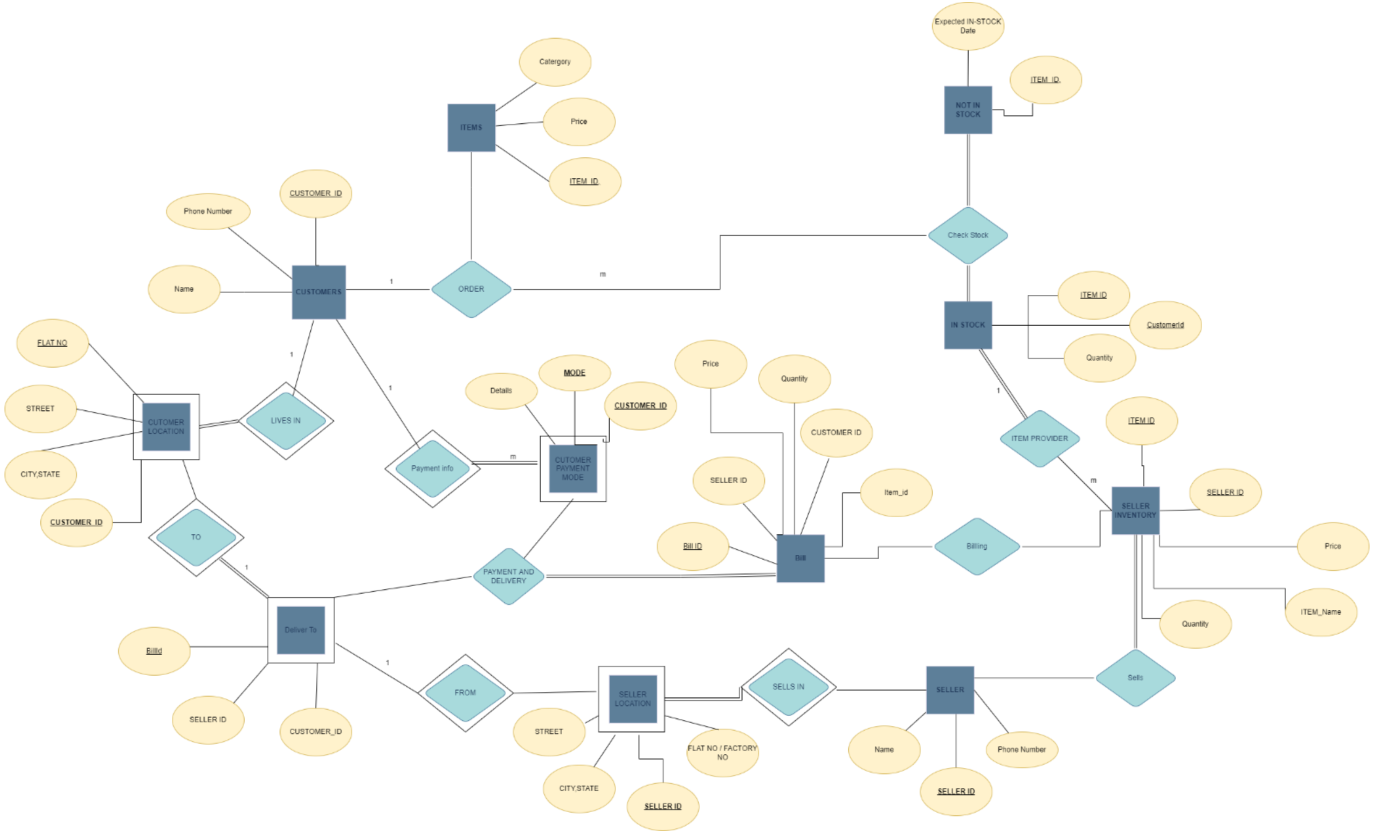
Group- 31

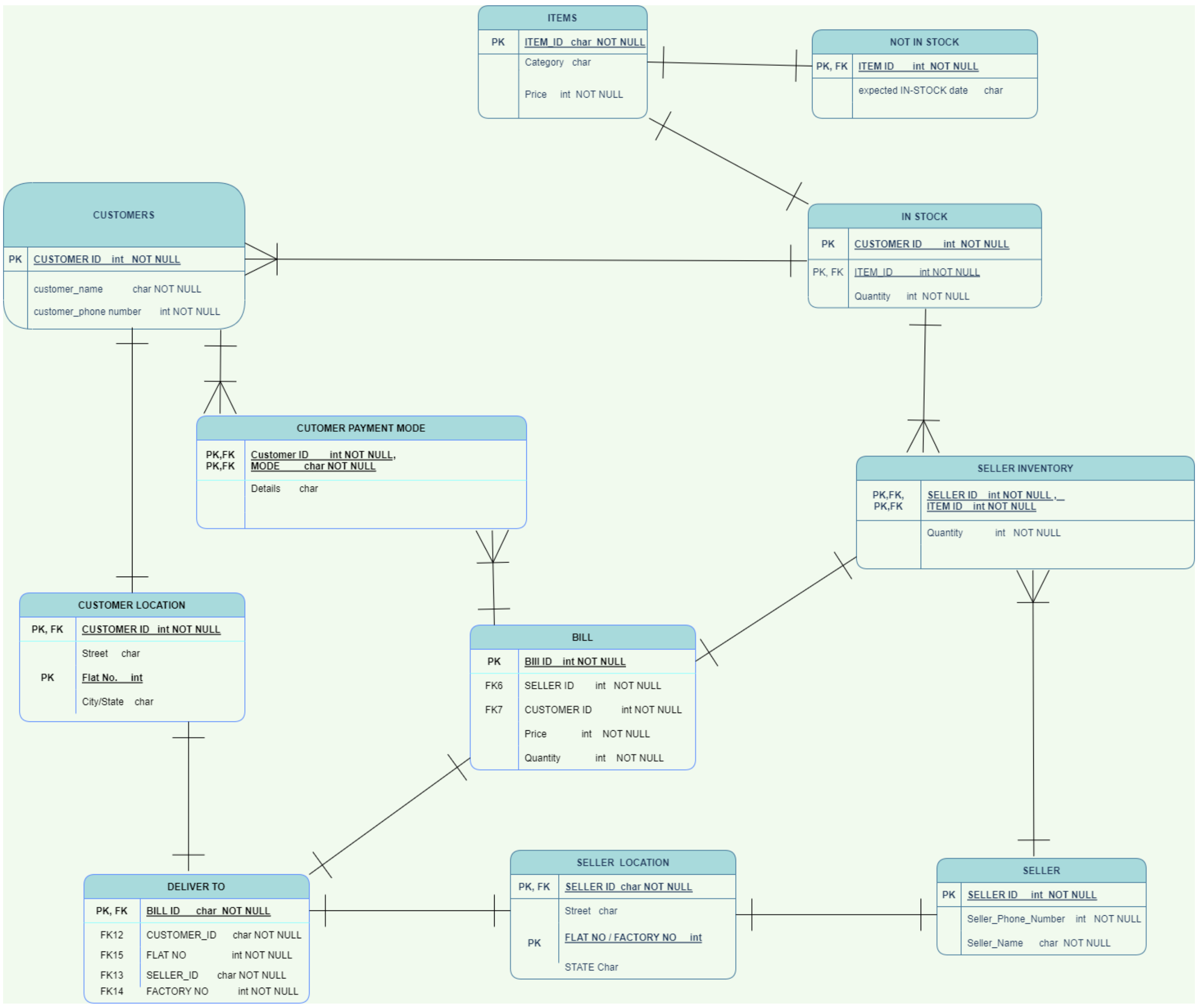
Name- Anand (2020280), Deeptanshu Barman (2020293),

Harsh Choudhary (2020433), Vaidik Kumar (2020346)

<u>Project Scope and Relational Schema-</u> The scope of this project is to understand fundamental of DBMS. In this project we are going to learn and experiment with ER- diagram and its conversion into Relational Scheme, creating a Logical Database Design and the valid constraints that arise in them. We will also learn about the how-to identity the Weak Identities, relationship orles and constraints and also how to define Entities with underlined Primary Key and establish relation with in them.

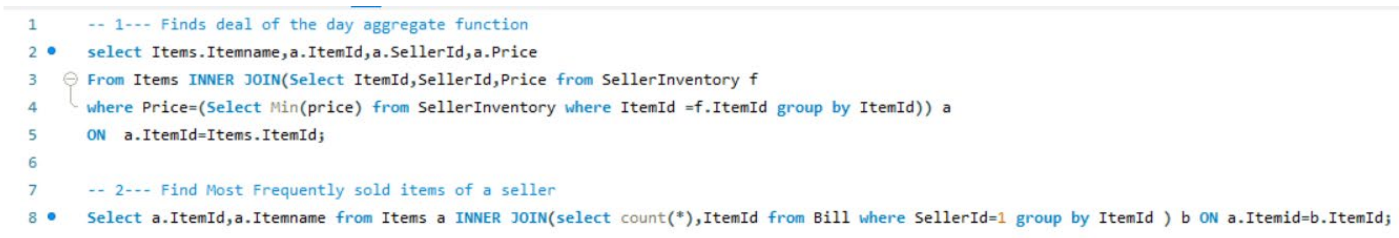
On top of this we rectified our mistakes that we found and followed the suggestions that we got from our TA. Because of this we made changes to our projects.
Now we have a new working tertiary relationship between (DELIVER TO, CUSTOMER PAYMENT MODE, BILL relation to Payment And Delivery). Now we can also add multiple address for customers and seller location. We also redesigned our tables, primary key, foreign key.





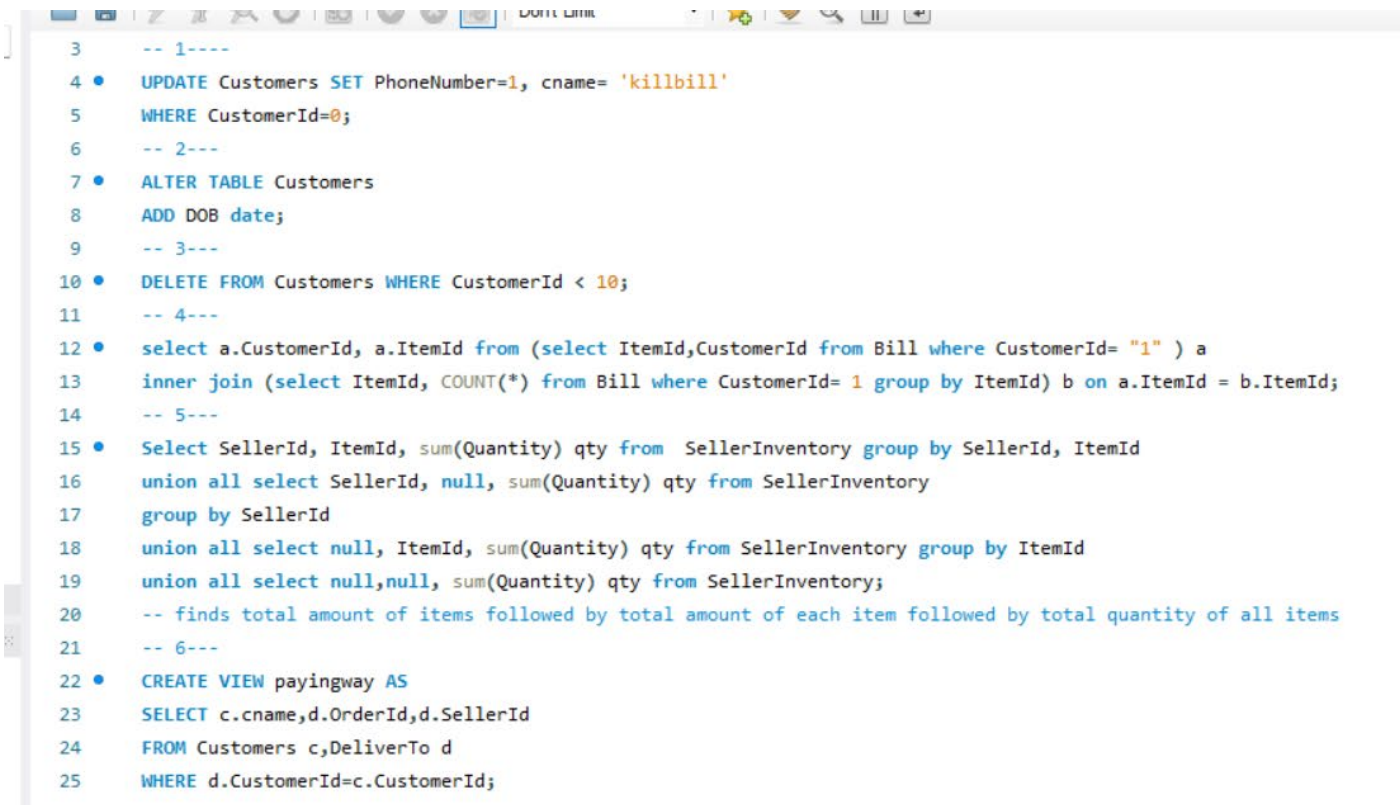
Embedded SQL Queries- Presented in front-end.

Advanced aggerated functions:

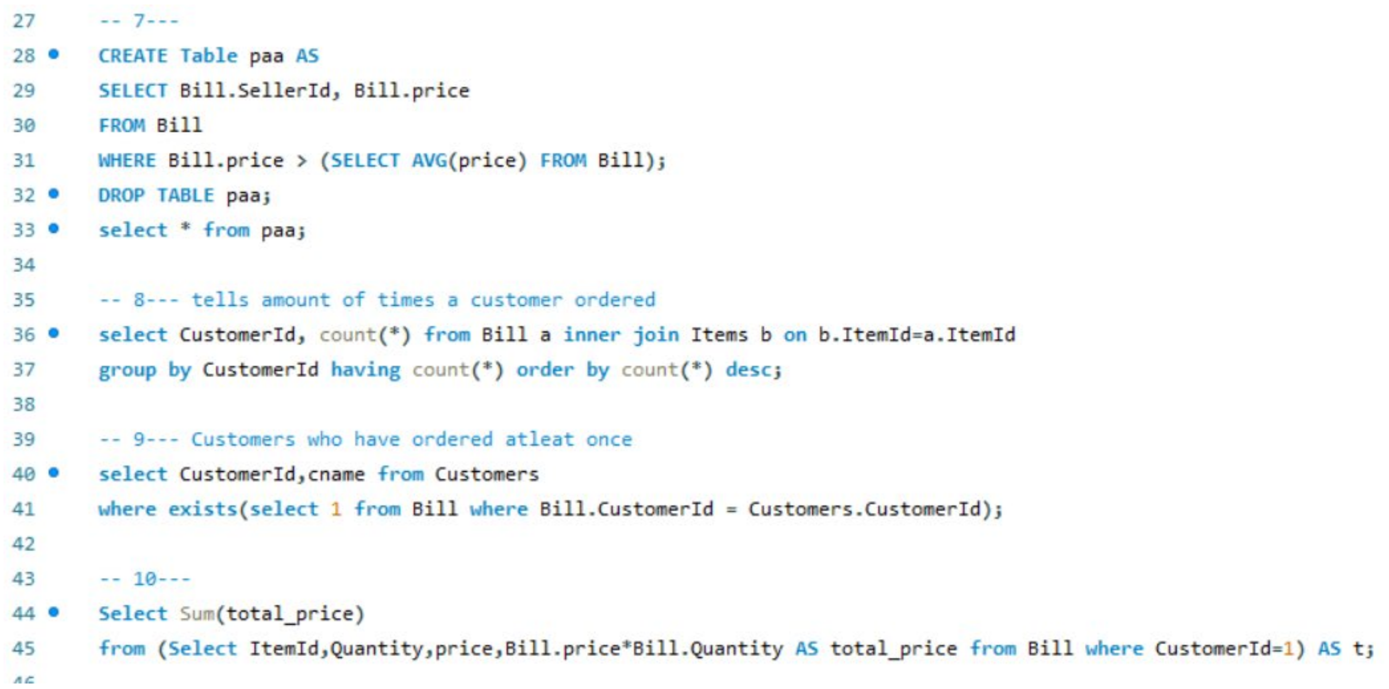


```sql
-- 1--- Finds deal of the day aggregate function
select Items.Itemname,a.ItemId,a.SellerId,a.Price
From Items INNER JOIN(Select ItemId,SellerId,Price from SellerInventory f
where Price=(Select Min(price) from SellerInventory where ItemId =f.ItemId group by ItemId)) a
ON  a.ItemId=Items.ItemId;

-- 2--- Find Most Frequently sold items of a seller
Select a.ItemId,a.Itemname from Items a INNER JOIN(select count(*),ItemId from Bill where SellerId=1 group by ItemId ) b ON a.Itemid=b.ItemId;
```

SQL Queries-



```sql
-- 1----
UPDATE Customers SET PhoneNumber=1, cname= 'killbill'
WHERE CustomerId=0;
-- 2---
ALTER TABLE Customers
ADD DOB date;
-- 3---
DELETE FROM Customers WHERE CustomerId < 10;
-- 4---
select a.CustomerId, a.ItemId from (select ItemId,CustomerId from Bill where CustomerId= "1" ) a
inner join (select ItemId, COUNT(*) from Bill where CustomerId= 1 group by ItemId) b on a.ItemId = b.ItemId;
-- 5---
Select SellerId, ItemId, sum(Quantity) qty from  SellerInventory group by SellerId, ItemId
union all select SellerId, null, sum(Quantity) qty from SellerInventory
group by SellerId
union all select null, ItemId, sum(Quantity) qty from SellerInventory group by ItemId
union all select null,null, sum(Quantity) qty from SellerInventory;
-- finds total amount of items followed by total amount of each item followed by total quantity of all items
-- 6---
CREATE VIEW payingway AS
SELECT c.cname,d.OrderId,d.SellerId
FROM Customers c,DeliverTo d
WHERE d.CustomerId=c.CustomerId;
```

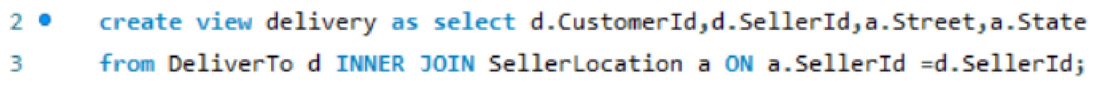


```sql
-- 7---
CREATE Table paa AS
SELECT Bill.SellerId, Bill.price
FROM Bill
WHERE Bill.price > (SELECT AVG(price) FROM Bill);
DROP TABLE paa;
select * from paa;

-- 8--- tells amount of times a customer ordered
select CustomerId, count(*) from Bill a inner join Items b on b.ItemId=a.ItemId
group by CustomerId having count(*) order by count(*) desc;

-- 9--- Customers who have ordered atleat once
select CustomerId,cname from Customers
where exists(select 1 from Bill where Bill.CustomerId = Customers.CustomerId);

-- 10---
Select Sum(total_price)
from (Select ItemId,Quantity,price,Bill.price*Bill.Quantity AS total_price from Bill where CustomerId=1) AS t;
```

Views and Grants-

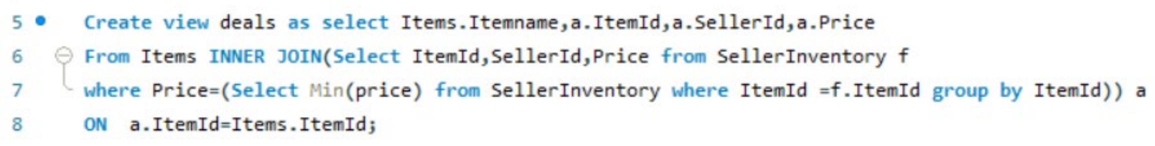
Views:

1.



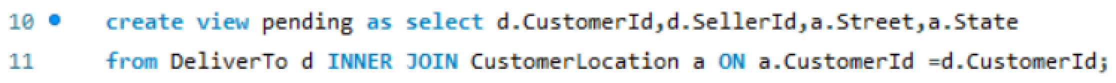
```sql
create view delivery as select d.CustomerId,d.SellerId,a.Street,a.State
from DeliverTo d INNER JOIN SellerLocation a ON a.SellerId =d.SellerId;
```

2.



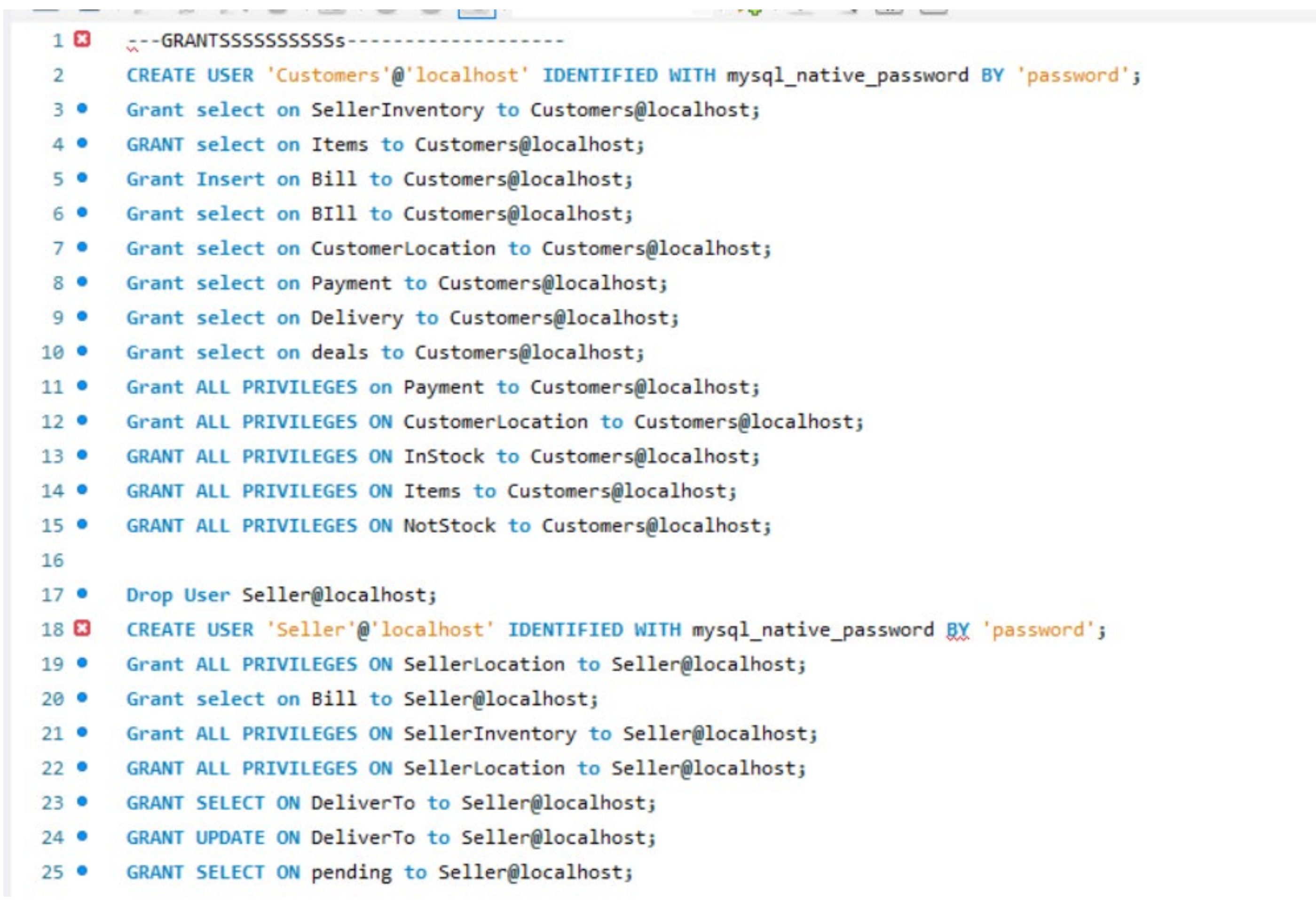
```sql
Create view deals as select Items.Itemname,a.ItemId,a.SellerId,a.Price
From Items INNER JOIN(Select ItemId,SellerId,Price from SellerInventory f
where Price=(Select Min(price) from SellerInventory where ItemId =f.ItemId group by ItemId)) a
ON  a.ItemId=Items.ItemId;
```

3.



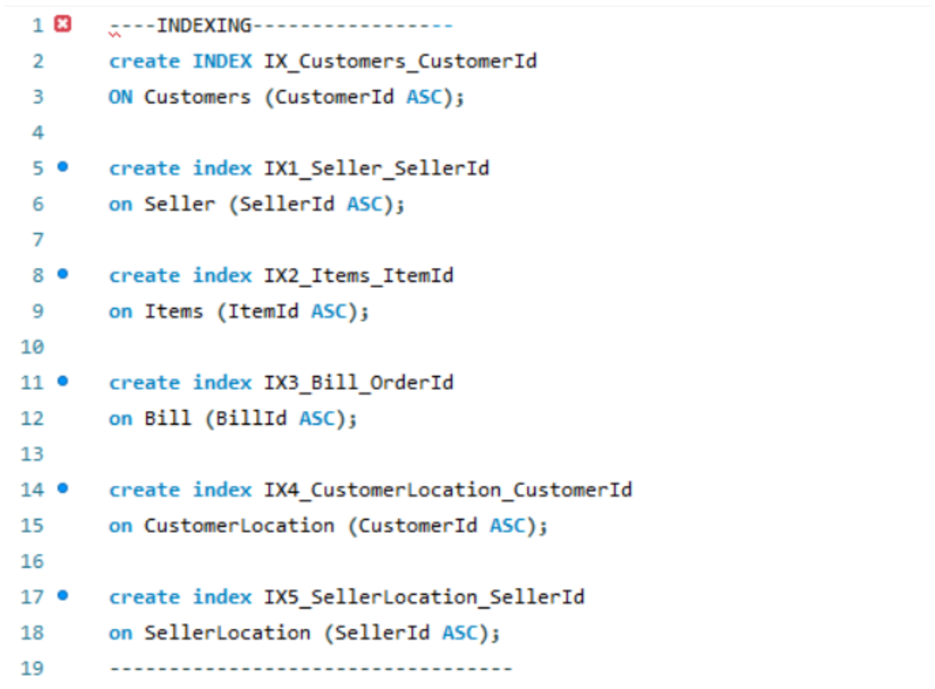
```sql
create view pending as select d.CustomerId,d.SellerId,a.Street,a.State
from DeliverTo d INNER JOIN CustomerLocation a ON a.CustomerId =d.CustomerId;
```

Grants:



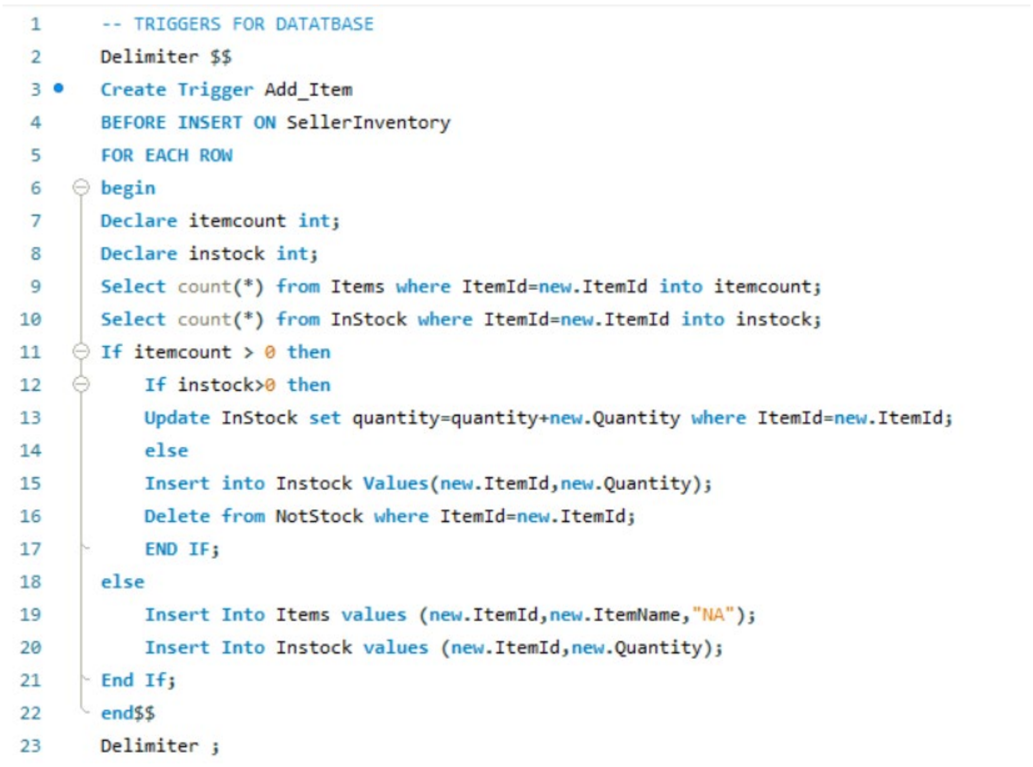
```sql
---GRANTSSSSSSSSSSs-------------------
CREATE USER 'Customers'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
Grant select on SellerInventory to Customers@localhost;
GRANT select on Items to Customers@localhost;
Grant Insert on Bill to Customers@localhost;
Grant select on BIll to Customers@localhost;
Grant select on CustomerLocation to Customers@localhost;
Grant select on Payment to Customers@localhost;
Grant select on Delivery to Customers@localhost;
Grant select on deals to Customers@localhost;
Grant ALL PRIVILEGES on Payment to Customers@localhost;
Grant ALL PRIVILEGES ON CustomerLocation to Customers@localhost;
GRANT ALL PRIVILEGES ON InStock to Customers@localhost;
GRANT ALL PRIVILEGES ON Items to Customers@localhost;
GRANT ALL PRIVILEGES ON NotStock to Customers@localhost;

Drop User Seller@localhost;
CREATE USER 'Seller'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
Grant ALL PRIVILEGES ON SellerLocation to Seller@localhost;
Grant select on Bill to Seller@localhost;
Grant ALL PRIVILEGES ON SellerInventory to Seller@localhost;
GRANT ALL PRIVILEGES ON SellerLocation to Seller@localhost;
GRANT SELECT ON DeliverTo to Seller@localhost;
GRANT UPDATE ON DeliverTo to Seller@localhost;
GRANT SELECT ON pending to Seller@localhost;
```

Indexing-



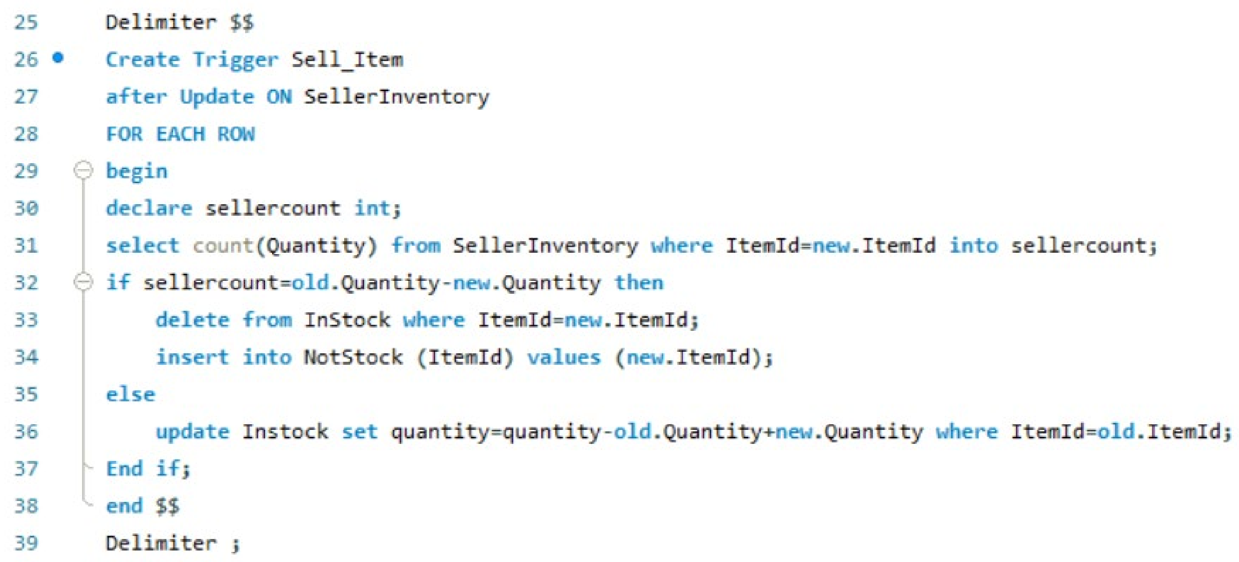
```
----INDEXING-----------------
create INDEX IX_Customers_CustomerId
ON Customers (CustomerId ASC);

create index IX1_Seller_SellerId
on Seller (SellerId ASC);

create index IX2_Items_ItemId
on Items (ItemId ASC);

create index IX3_Bill_OrderId
on Bill (BillId ASC);

create index IX4_CustomerLocation_CustomerId
on CustomerLocation (CustomerId ASC);

create index IX5_SellerLocation_SellerId
on SellerLocation (SellerId ASC);
----------------------------------
```

Triggers-



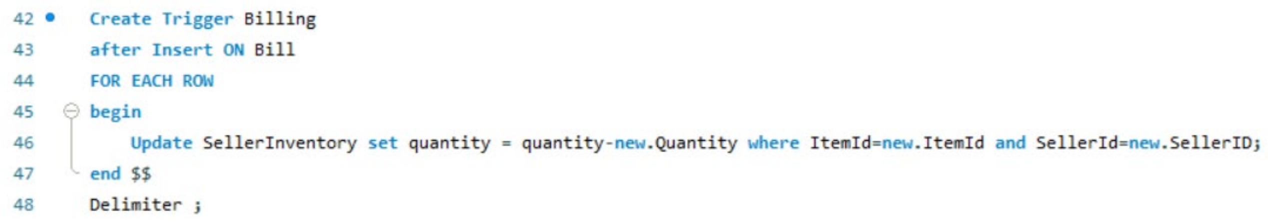
1.

```
-- TRIGGERS FOR DATATBASE
Delimiter $$
Create Trigger Add_Item
BEFORE INSERT ON SellerInventory
FOR EACH ROW
begin
Declare itemcount int;
Declare instock int;
Select count(*) from Items where ItemId=new.ItemId into itemcount;
Select count(*) from InStock where ItemId=new.ItemId into instock;
If itemcount > 0 then
    If instock>0 then
    Update InStock set quantity=quantity+new.Quantity where ItemId=new.ItemId;
    else
    Insert into Instock Values(new.ItemId,new.Quantity);
    Delete from NotStock where ItemId=new.ItemId;
    END IF;
else
    Insert Into Items values (new.ItemId,new.ItemName,"NA");
    Insert Into Instock values (new.ItemId,new.Quantity);
End If;
end$$
Delimiter ;
```



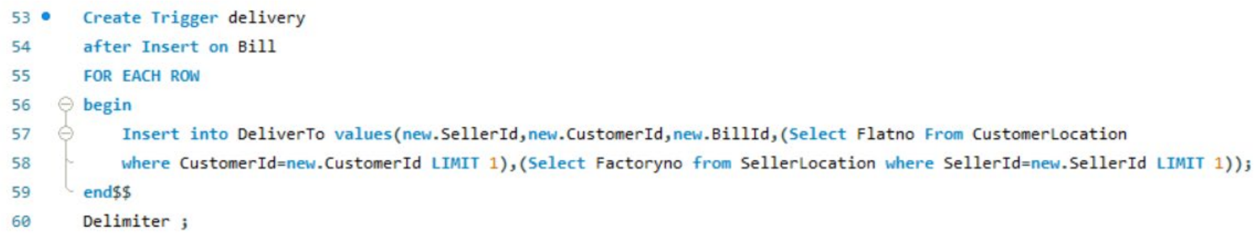
2.

```sql
Delimiter $$
Create Trigger Sell_Item
after Update ON SellerInventory
FOR EACH ROW
begin
declare sellercount int;
select count(Quantity) from SellerInventory where ItemId=new.ItemId into sellercount;
if sellercount=old.Quantity-new.Quantity then
    delete from InStock where ItemId=new.ItemId;
    insert into NotStock (ItemId) values (new.ItemId);
else
    update Instock set quantity=quantity-old.Quantity+new.Quantity where ItemId=old.ItemId;
End if;
end $$
Delimiter ;
```



3.

```sql
Create Trigger Billing
after Insert ON Bill
FOR EACH ROW
begin
    Update SellerInventory set quantity = quantity-new.Quantity where ItemId=new.ItemId and SellerId=new.SellerID;
end $$
Delimiter ;
```



4.

```sql
Create Trigger delivery
after Insert on Bill
FOR EACH ROW
begin
    Insert into DeliverTo values(new.SellerId,new.CustomerId,new.BillId,(Select Flatno From CustomerLocation
    where CustomerId=new.CustomerId LIMIT 1),(Select Factoryno from SellerLocation where SellerId=new.SellerId LIMIT 1));
end$$
Delimiter ;
```