DBMS PROJECT

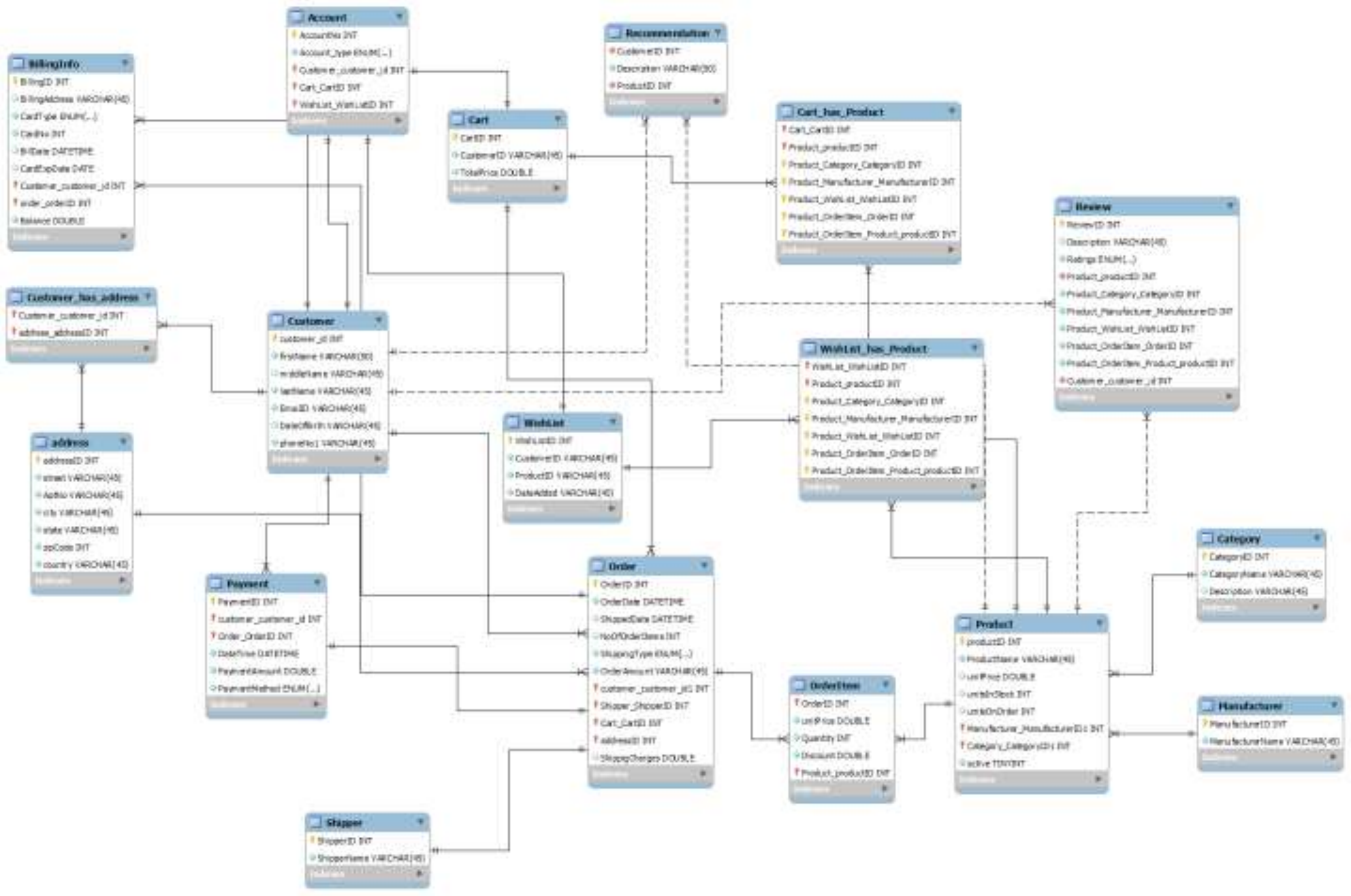
ANN SARA SAJEE – 001813733

AMAZON DATABASE

PROJECT SUMMARY

E-commerce databases are quite interesting and hence I choose an E-commerce database- AMAZON to work on. The purpose was to deliver an E-commerce database that is used for e-commerce operations like check out products, place an order, do payment, review products, find for recommendations but also for data analysis to extract valuable insights from these data for business betterment. I am using queries, joins, views, stored procedures, triggers and a transaction and a few analytical queries to show the working of this project.

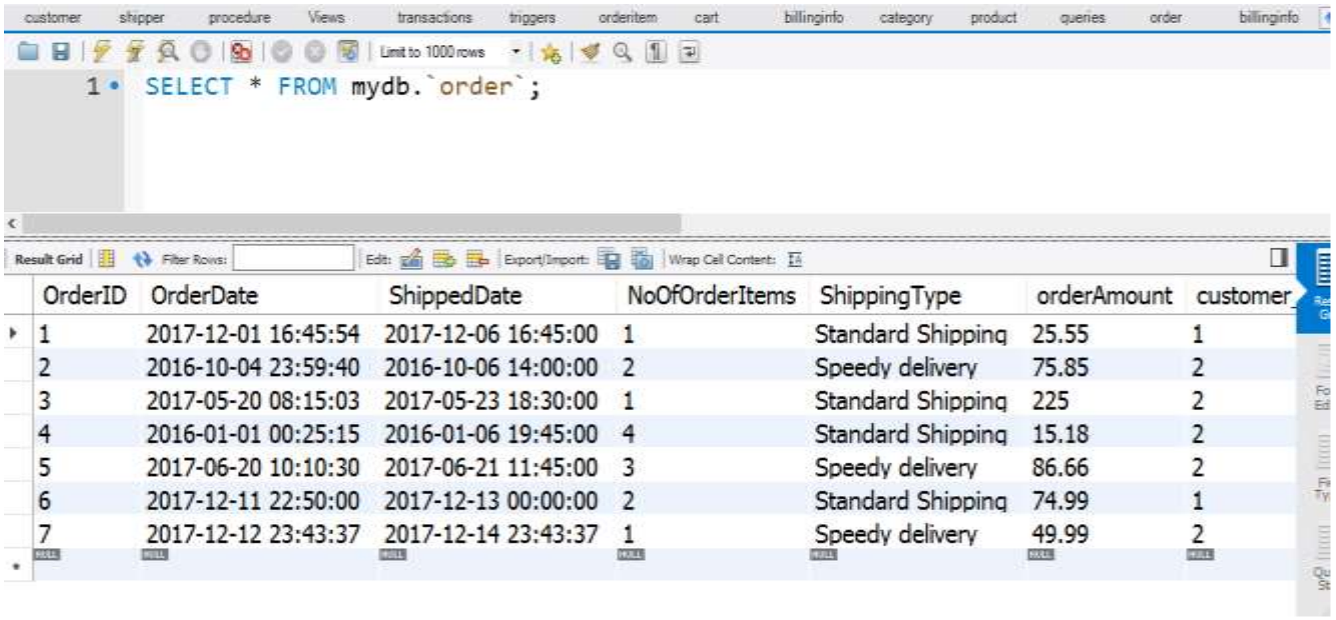
EER DIAGRAM


Account
AccountNo INT
Account_type ENUM(...)
Customer_customer_id INT
Cart_CartID INT
WishList_WishListID INT
BillingInfo
BillingID INT
BillingAddress VARCHAR(45)
CardType ENUM(...)
CardNo INT
BillDate DATETIME
CardExpDate DATE
Customer_customer_id INT
order_orderID INT
Balance DOUBLE
Recommendation
CustomerID INT
Description VARCHAR(30)
ProductID INT
Cart
CartID INT
CustomerID VARCHAR(45)
TotalPrice DOUBLE
Cart_has_Product
Cart_CartID INT
Product_productID INT
Product_Category_CategoryID INT
Product_Manufacturer_ManufacturerID INT
Product_WishList_WishListID INT
Product_OrderItem_OrderID INT
Product_OrderItem_Product_productID INT
Review
ReviewID INT
Description VARCHAR(45)
Ratings ENUM(...)
Product_productID INT
Product_Category_CategoryID INT
Product_Manufacturer_ManufacturerID INT
Product_WishList_WishListID INT
Product_OrderItem_OrderID INT
Product_OrderItem_Product_productID INT
Customer_customer_id INT
Customer_has_address
Customer_customer_id INT
address_addressID INT
Customer
customer_id INT
firstName VARCHAR(30)
middleName VARCHAR(45)
lastName VARCHAR(45)
EmailID VARCHAR(45)
DateOfBirth VARCHAR(45)
phoneNo1 VARCHAR(45)
WishList_has_Product
WishList_WishListID INT
Product_productID INT
Product_Category_CategoryID INT
Product_Manufacturer_ManufacturerID INT
Product_WishList_WishListID INT
Product_OrderItem_OrderID INT
Product_OrderItem_Product_productID INT
address
addressID INT
street VARCHAR(45)
AptNo VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipCode INT
country VARCHAR(45)
WishList
WishListID INT
CustomerID VARCHAR(45)
ProductID VARCHAR(45)
DateAdded VARCHAR(45)
Category
CategoryID INT
CategoryName VARCHAR(45)
Description VARCHAR(45)
Payment
PaymentID INT
customer_customer_id INT
Order_OrderID INT
DateTime DATETIME
PaymentAmount DOUBLE
PaymentMethod ENUM(...)
Order
OrderID INT
OrderDate DATETIME
ShippedDate DATETIME
NoOfOrderItems INT
ShippingType ENUM(...)
OrderAmount VARCHAR(45)
customer_customer_id1 INT
Shipper_ShipperID INT
Cart_CartID INT
addressID INT
ShippingCharges DOUBLE
OrderItem
OrderID INT
unitPrice DOUBLE
Quantity INT
Discount DOUBLE
Product_productID INT
Product
productID INT
ProductName VARCHAR(45)
unitPrice DOUBLE
unitsInStock INT
unitsOnOrder INT
Manufacturer_ManufacturerID1 INT
Category_CategoryID1 INT
active TINYINT
Manufacturer
ManufacturerID INT
ManufacturerName VARCHAR(45)
Shipper
ShipperID INT
ShipperName VARCHAR(45)

TABLES AND DATA

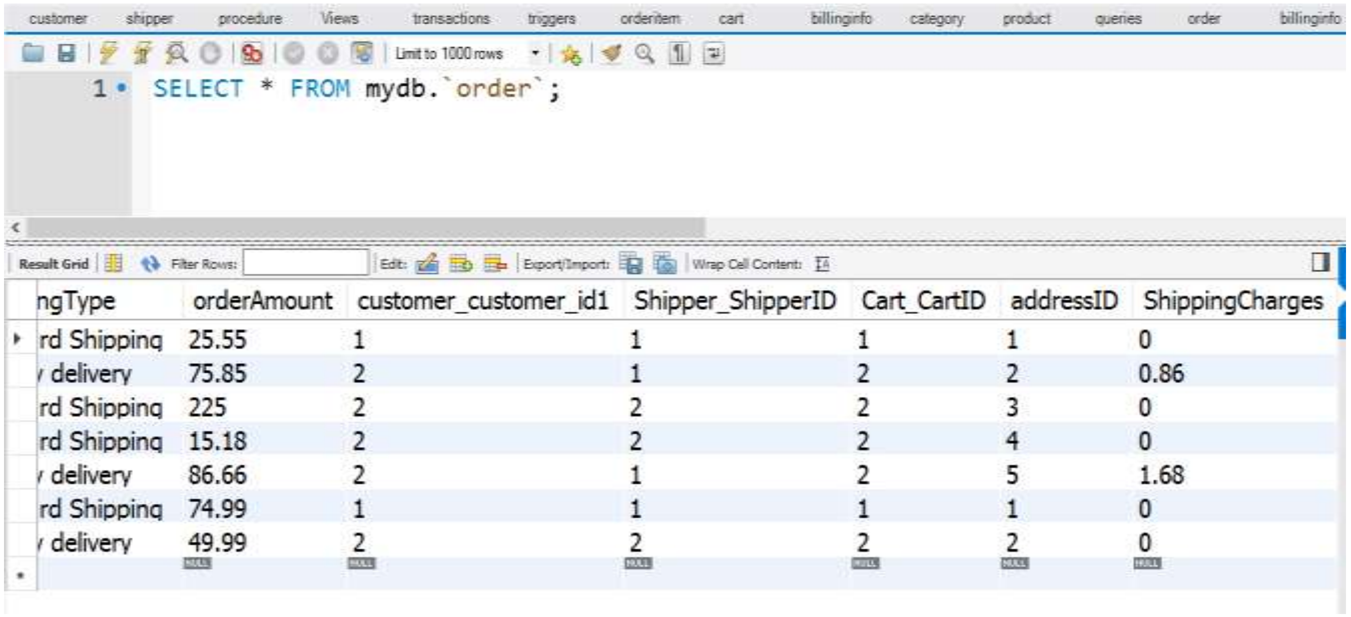
The following tables are included in my project:

1. **ORDERS:** The table contains the following columns:



```
SELECT * FROM mydb.`order`;
```

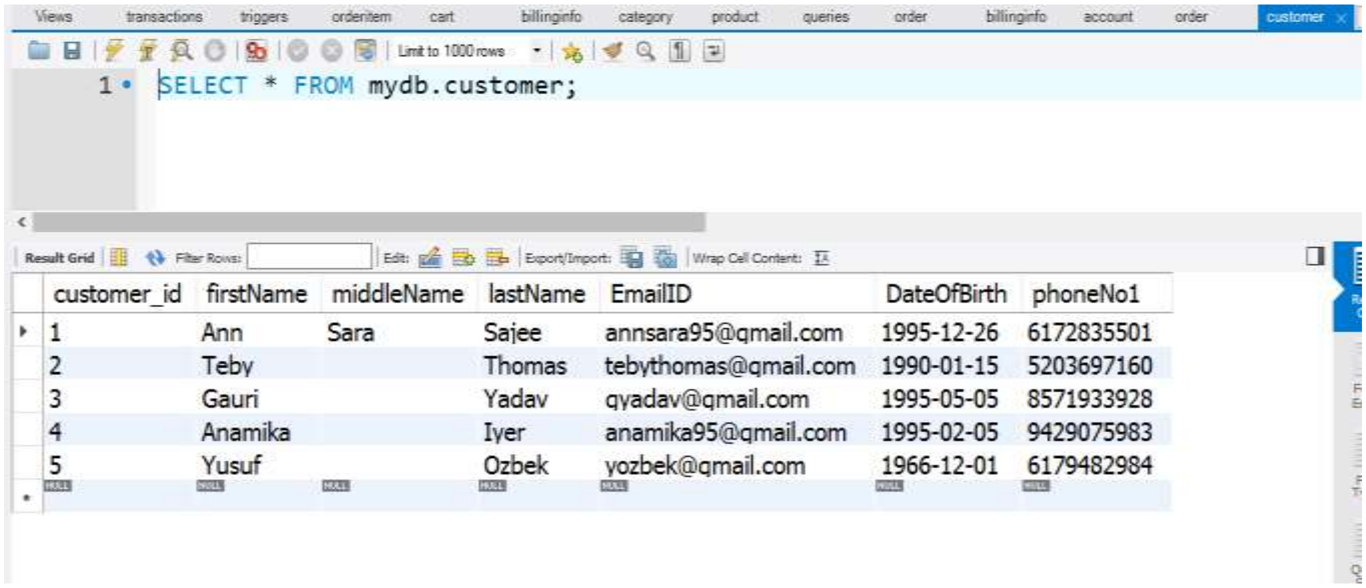
OrderID	OrderDate	ShippedDate	NoOfOrderItems	ShippingType	orderAmount	customer_
1	2017-12-01 16:45:54	2017-12-06 16:45:00	1	Standard Shipping	25.55	1
2	2016-10-04 23:59:40	2016-10-06 14:00:00	2	Speedy delivery	75.85	2
3	2017-05-20 08:15:03	2017-05-23 18:30:00	1	Standard Shipping	225	2
4	2016-01-01 00:25:15	2016-01-06 19:45:00	4	Standard Shipping	15.18	2
5	2017-06-20 10:10:30	2017-06-21 11:45:00	3	Speedy delivery	86.66	2
6	2017-12-11 22:50:00	2017-12-13 00:00:00	2	Standard Shipping	74.99	1
7	2017-12-12 23:43:37	2017-12-14 23:43:37	1	Speedy delivery	49.99	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
SELECT * FROM mydb.`order`;
```

ngType	orderAmount	customer_customer_id1	Shipper_ShipperID	Cart_CartID	addressID	ShippingCharges
rd Shipping	25.55	1	1	1	1	0
/ delivery	75.85	2	1	2	2	0.86
rd Shipping	225	2	2	2	3	0
rd Shipping	15.18	2	2	2	4	0
/ delivery	86.66	2	1	2	5	1.68
rd Shipping	74.99	1	1	1	1	0
/ delivery	49.99	2	2	2	2	0
	NULL	NULL	NULL	NULL	NULL	NULL

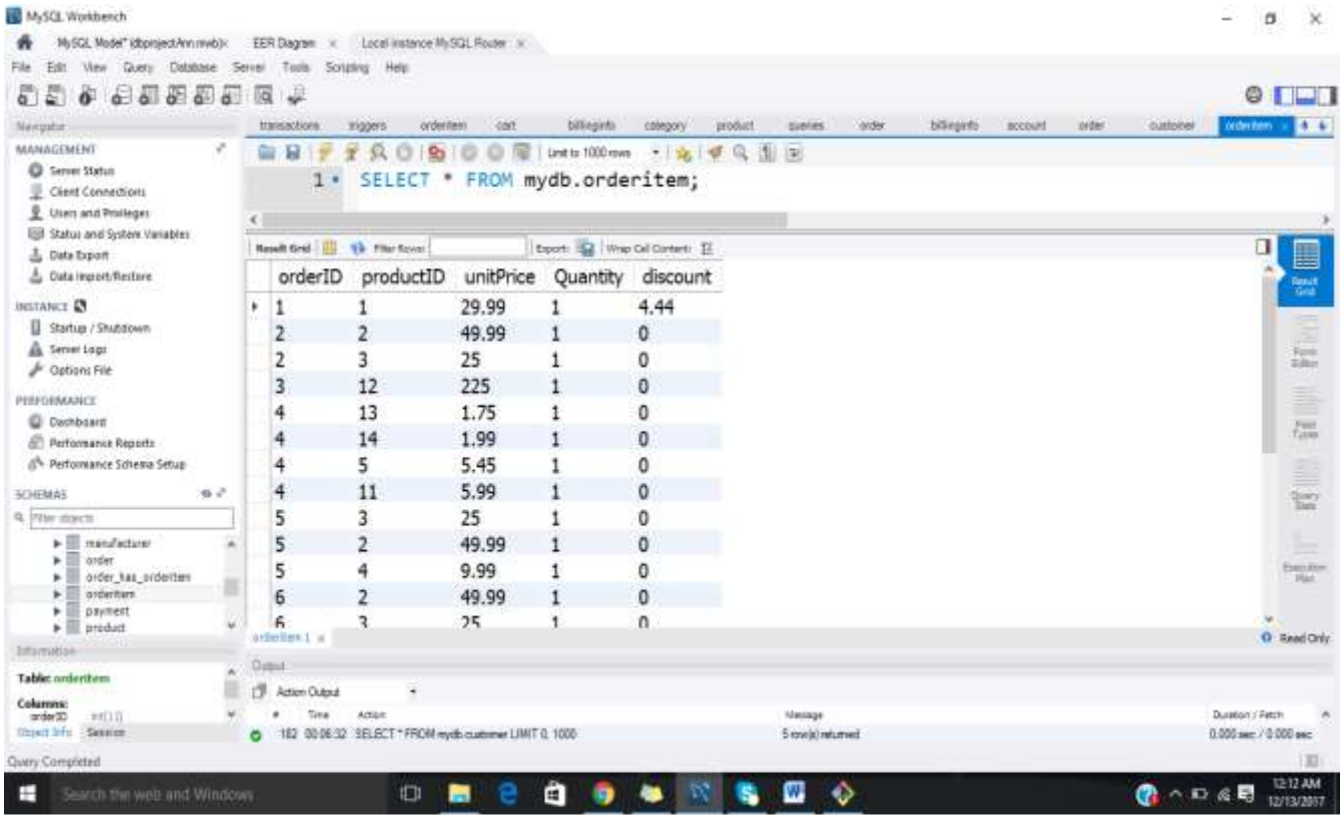
2. **CUSTOMERS:** This table contains basic information about the customers



```
SELECT * FROM mydb.customer;
```

customer_id	firstName	middleName	lastName	EmailID	DateOfBirth	phoneNo1
1	Ann	Sara	Sajee	annsara95@gmail.com	1995-12-26	6172835501
2	Teby		Thomas	tebythomas@gmail.com	1990-01-15	5203697160
3	Gauri		Yadav	gyadav@gmail.com	1995-05-05	8571933928
4	Anamika		Iyer	anamika95@gmail.com	1995-02-05	9429075983
5	Yusuf		Ozbek	yozbek@gmail.com	1966-12-01	6179482984
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. **ORDER ITEMS:** An order contains a number of order items. It's basically a product with a specific unit price, quantity and discounts. It has the following attributes :

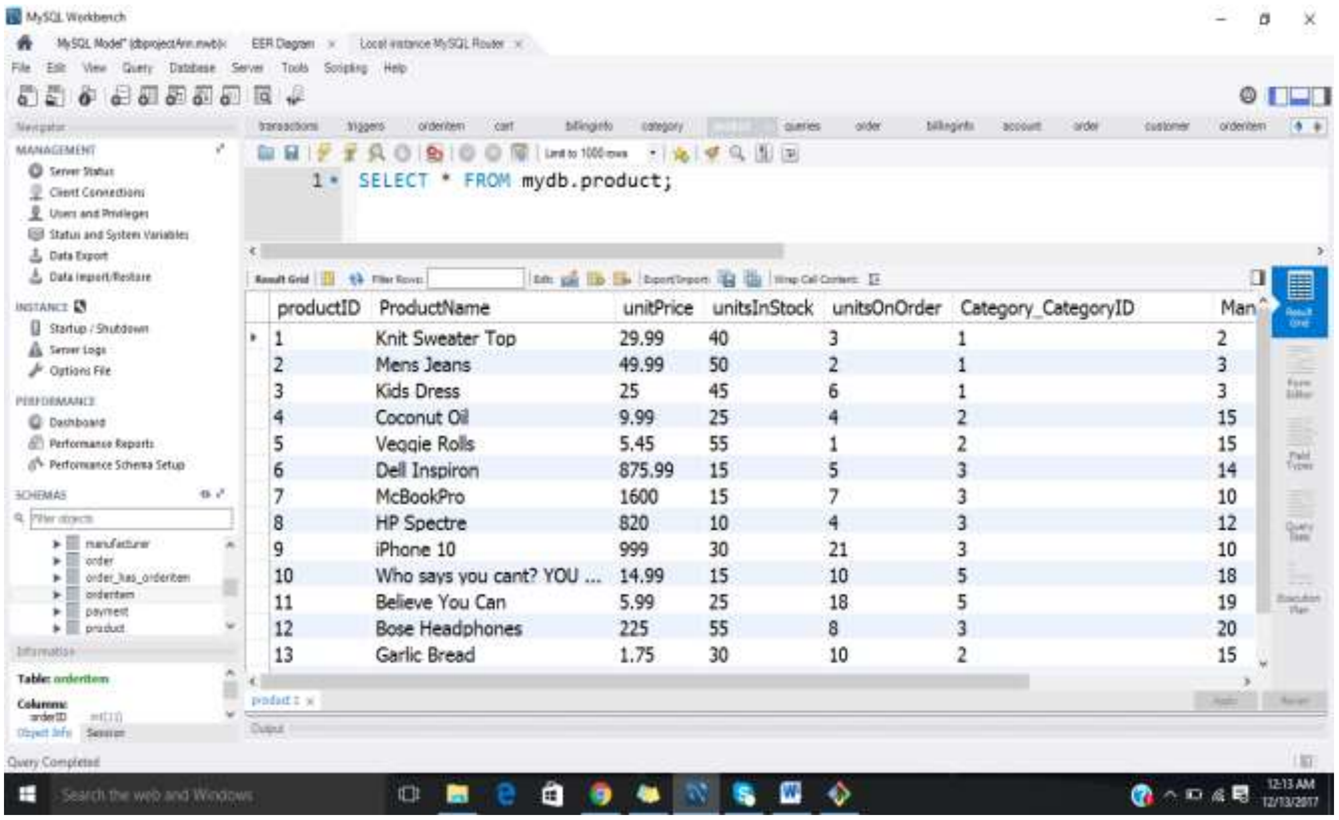


```
SELECT * FROM mydb.orderitem;
```

orderID	productID	unitPrice	Quantity	discount
1	1	29.99	1	4.44
2	2	49.99	1	0
2	3	25	1	0
3	12	225	1	0
4	13	1.75	1	0
4	14	1.99	1	0
4	5	5.45	1	0
4	11	5.99	1	0
5	3	25	1	0
5	2	49.99	1	0
5	4	9.99	1	0
6	2	49.99	1	0
6	3	25	1	0

```
182  00:06:32  SELECT * FROM mydb.customer LIMIT 0, 1000    5 row(s) returned    0.000 sec / 0.000 sec
```

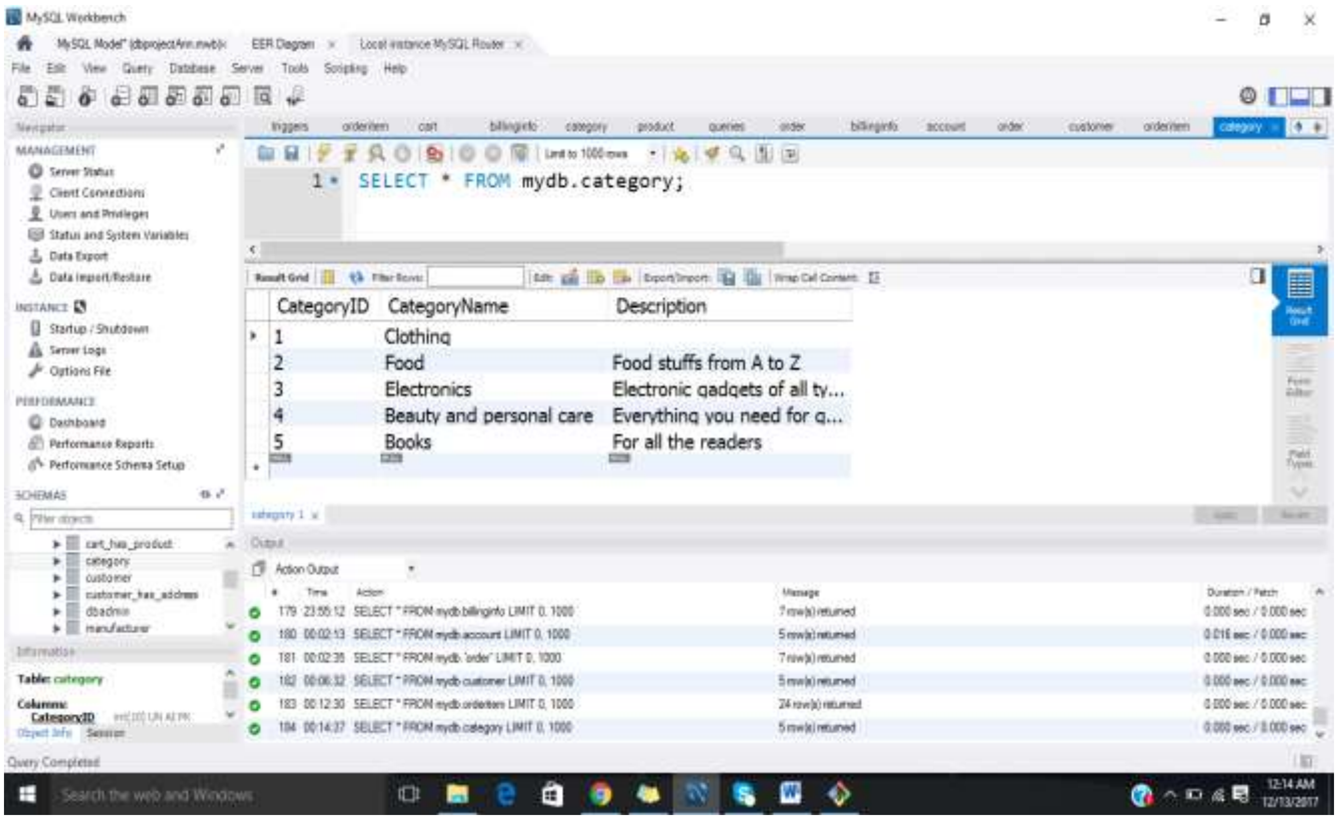
4. **PRODUCTS:** Product is an item that is delivered to the customers. It has the following attributes :



```
SELECT * FROM mydb.product;
```

productID	ProductName	unitPrice	unitsInStock	unitsOnOrder	Category_CategoryID	Man
1	Knit Sweater Top	29.99	40	3	1	2
2	Mens Jeans	49.99	50	2	1	3
3	Kids Dress	25	45	6	1	3
4	Coconut Oil	9.99	25	4	2	15
5	Veggie Rolls	5.45	55	1	2	15
6	Dell Inspiron	875.99	15	5	3	14
7	McBookPro	1600	15	7	3	10
8	HP Spectre	820	10	4	3	12
9	iPhone 10	999	30	21	3	10
10	Who says you cant? YOU ...	14.99	15	10	5	18
11	Believe You Can	5.99	25	18	5	19
12	Bose Headphones	225	55	8	3	20
13	Garlic Bread	1.75	30	10	2	15

5. **CATEGORY:** Every product belongs to a certain category. Its attributes are as follows:

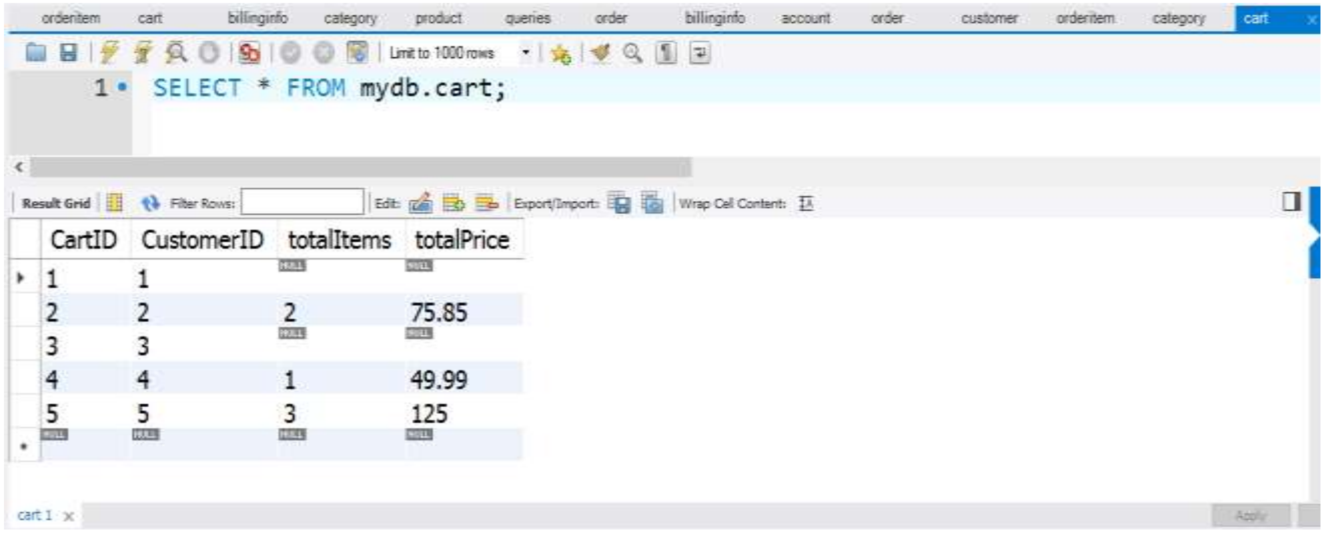


```
SELECT * FROM mydb.category;
```

CategoryID	CategoryName	Description
1	Clothing	
2	Food	Food stuffs from A to Z
3	Electronics	Electronic gadgets of all ty...
4	Beauty and personal care	Everything you need for g...
5	Books	For all the readers
NULL	NULL	NULL

```
#    Time      Action                                              Message             Duration / Fetch
179  23:55:12  SELECT * FROM mydb.billinginfo LIMIT 0, 1000        7 row(s) returned   0.000 sec / 0.000 sec
180  00:02:13  SELECT * FROM mydb.account LIMIT 0, 1000            5 row(s) returned   0.016 sec / 0.000 sec
181  00:02:35  SELECT * FROM mydb.`order` LIMIT 0, 1000            7 row(s) returned   0.000 sec / 0.000 sec
182  00:06:32  SELECT * FROM mydb.customer LIMIT 0, 1000           5 row(s) returned   0.000 sec / 0.000 sec
183  00:12:30  SELECT * FROM mydb.orderitem LIMIT 0, 1000          24 row(s) returned  0.000 sec / 0.000 sec
184  00:14:37  SELECT * FROM mydb.category LIMIT 0, 1000           5 row(s) returned   0.000 sec / 0.000 sec
```

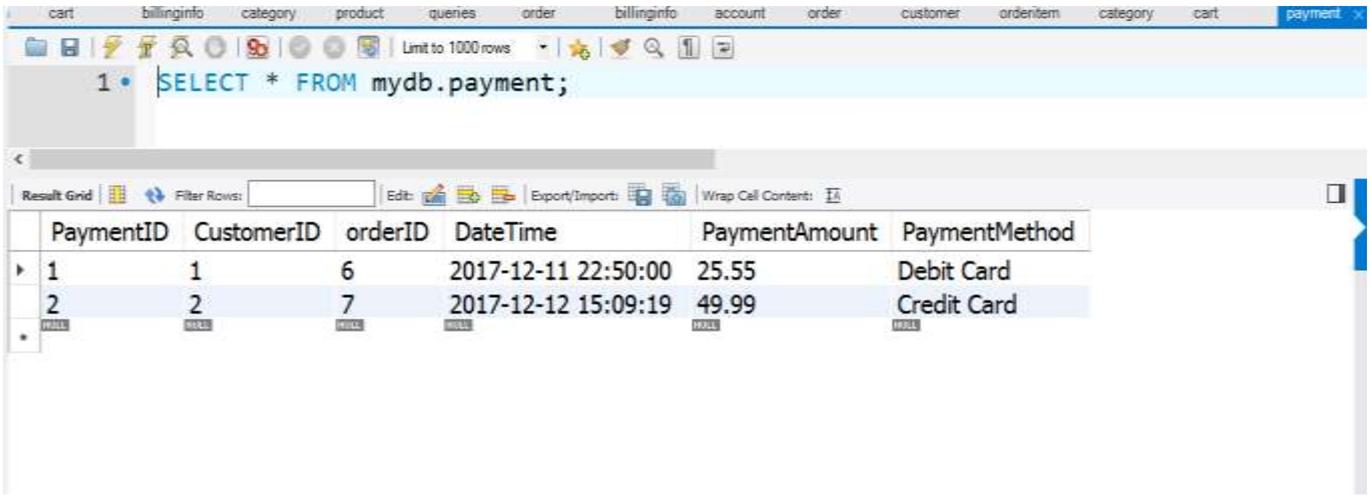
6. **CART:** A temporary list of items the customer wants to purchase. It has the following attributes:



```
SELECT * FROM mydb.cart;
```

CartID	CustomerID	totalItems	totalPrice
1	1	NULL	NULL
2	2	2	75.85
3	3	NULL	NULL
4	4	1	49.99
5	5	3	125
NULL	NULL	NULL	NULL

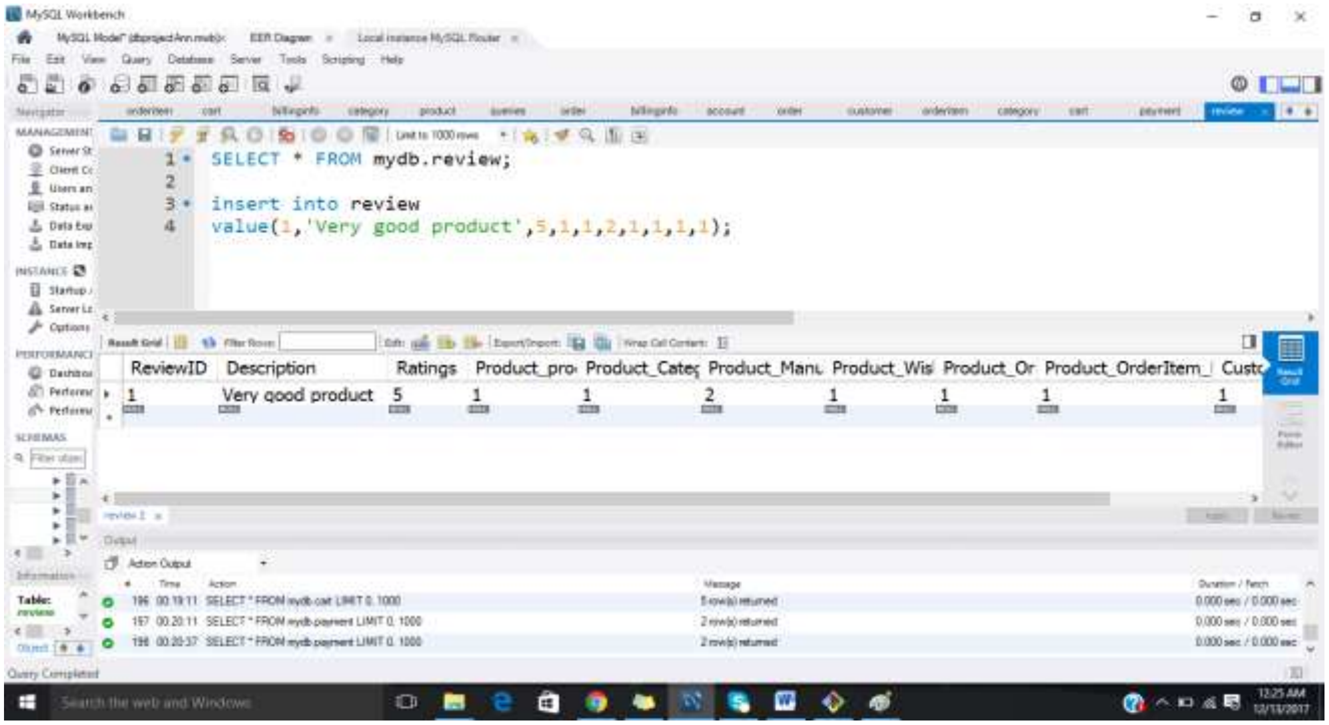
7. **PAYMENT:** When a customer purchases items, a payment is made. This entity has the following attributes:



```
SELECT * FROM mydb.payment;
```

PaymentID	CustomerID	orderID	DateTime	PaymentAmount	PaymentMethod
1	1	6	2017-12-11 22:50:00	25.55	Debit Card
2	2	7	2017-12-12 15:09:19	49.99	Credit Card

8. **REVIEWS:** A customer can give reviews for the product he bought



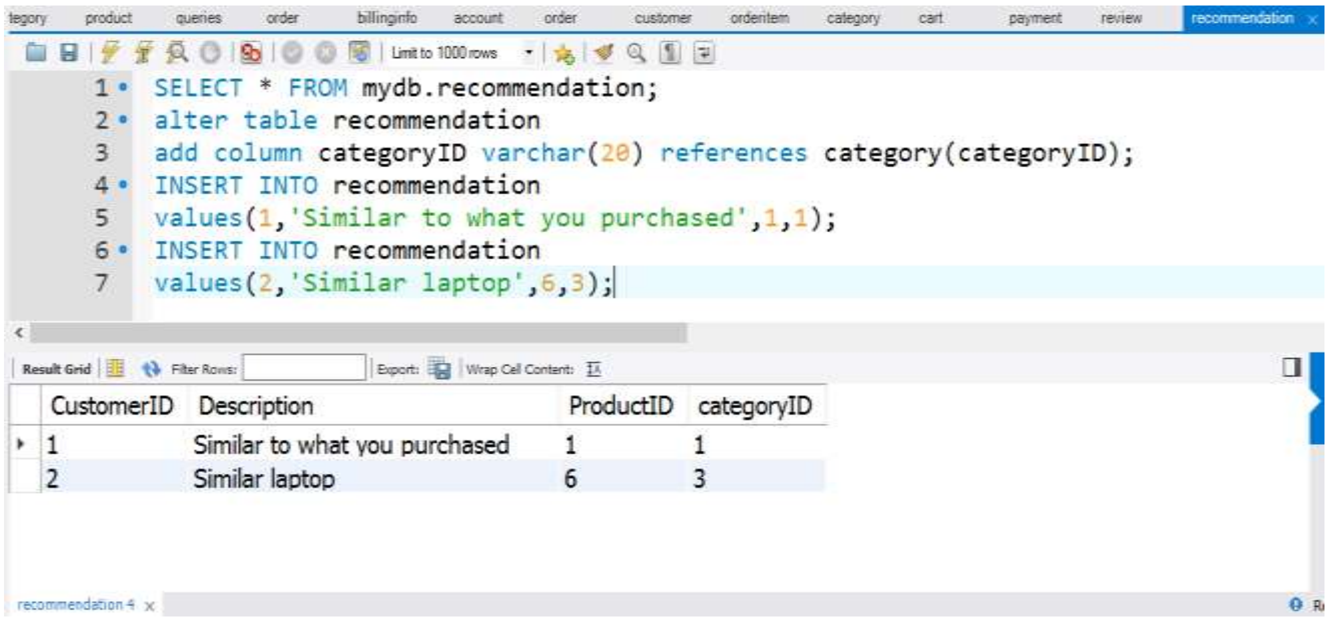
```
SELECT * FROM mydb.review;

insert into review
value(1,'Very good product',5,1,1,2,1,1,1,1);
```

ReviewID	Description	Ratings	Product_pro	Product_Categ	Product_Manu	Product_Wis	Product_Or	Product_OrderItem_	Custo
1	Very good product	5	1	1	2	1	1	1	1

```
#    Time      Action                                     Message            Duration / Fetch
196  00:19:11  SELECT * FROM mydb.cart LIMIT 0, 1000      5 row(s) returned  0.000 sec / 0.000 sec
197  00:20:11  SELECT * FROM mydb.payment LIMIT 0, 1000   2 row(s) returned  0.000 sec / 0.000 sec
198  00:20:37  SELECT * FROM mydb.payment LIMIT 0, 1000   2 row(s) returned  0.000 sec / 0.000 sec
```

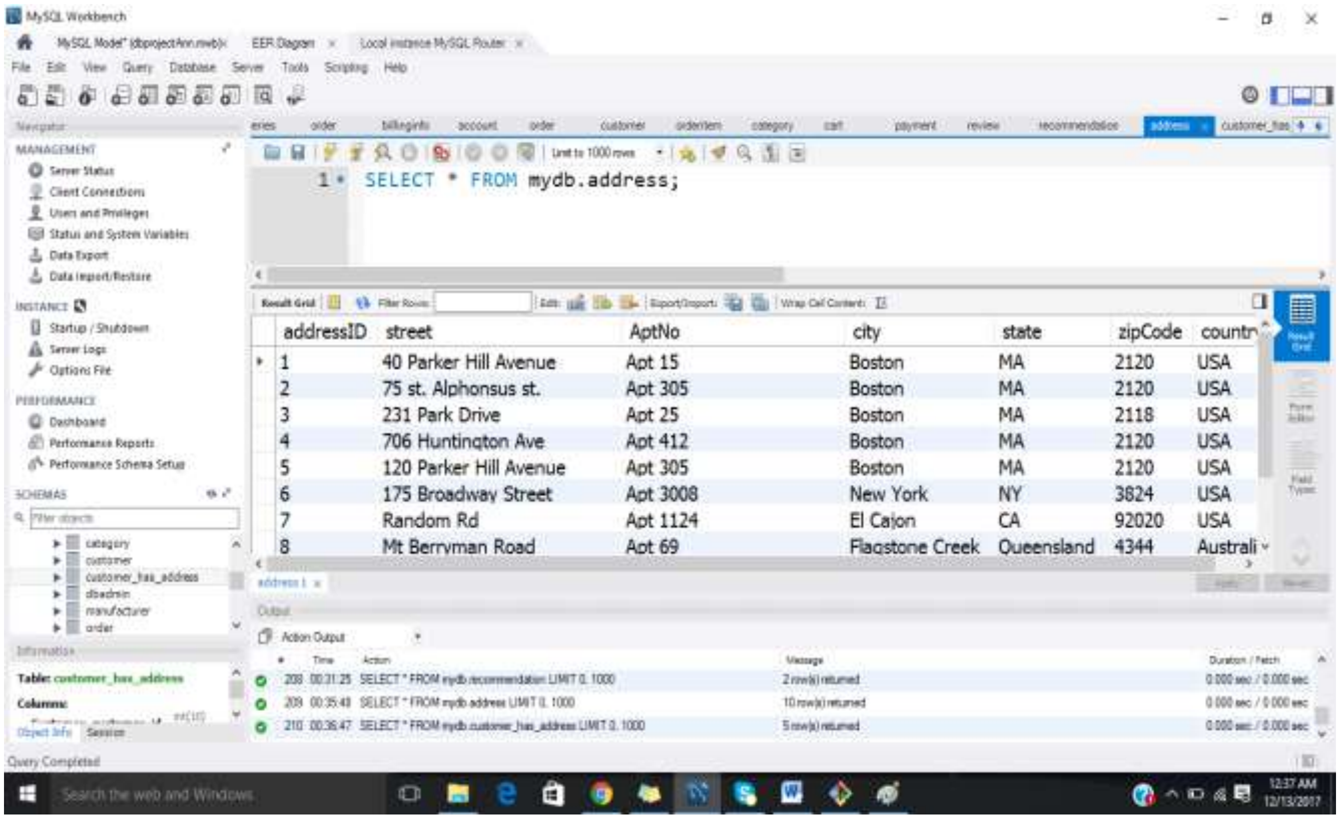
9. **RECOMMEMDATIONS:** Every customer gets a recommendation based on his/her purchases



```
SELECT * FROM mydb.recommendation;
alter table recommendation
add column categoryID varchar(20) references category(categoryID);
INSERT INTO recommendation
values(1,'Similar to what you purchased',1,1);
INSERT INTO recommendation
values(2,'Similar laptop',6,3);
```

CustomerID	Description	ProductID	categoryID
1	Similar to what you purchased	1	1
2	Similar laptop	6	3

10. **ADDRESS:** Customer addresses are placed in separate column for normalization of data

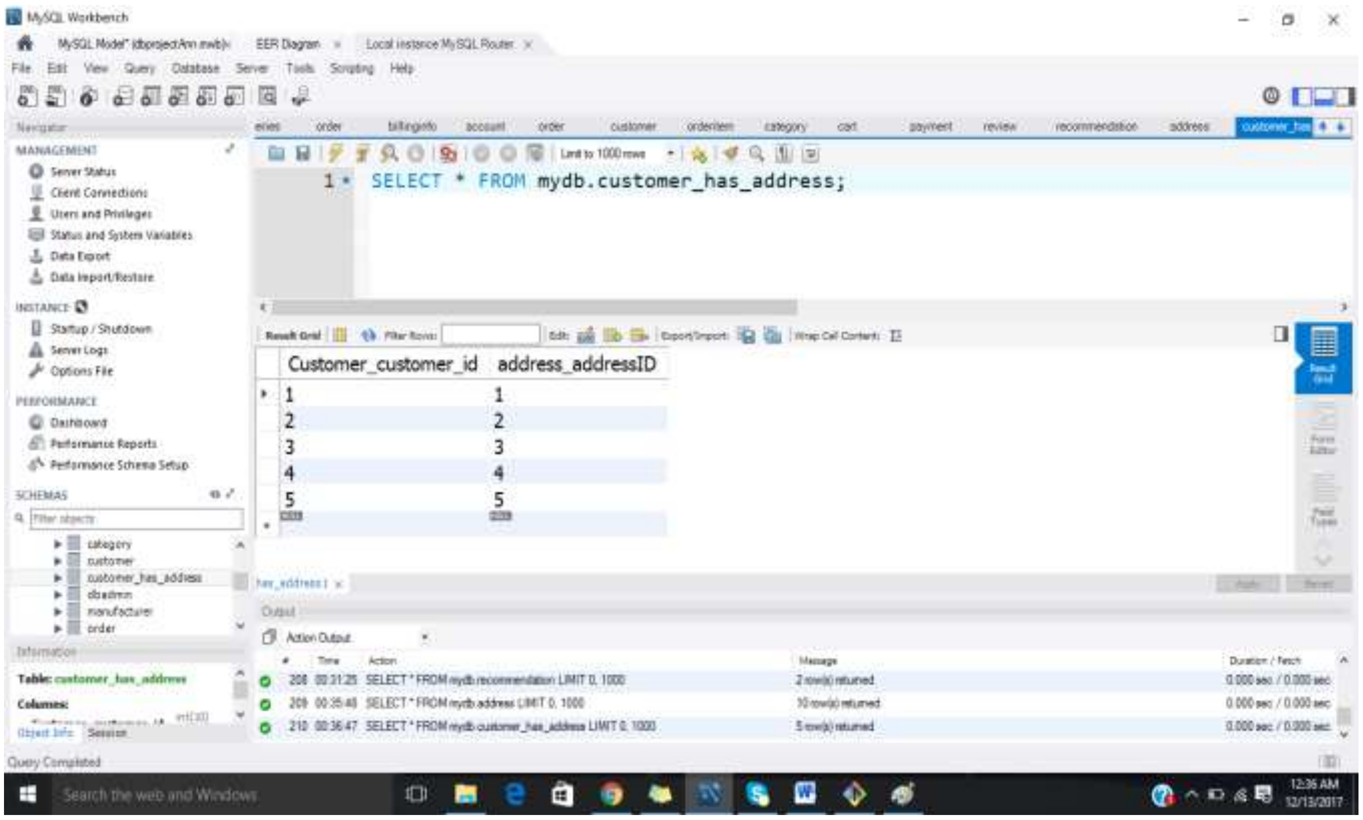


```
SELECT * FROM mydb.address;
```

addressID	street	AptNo	city	state	zipCode	countr
1	40 Parker Hill Avenue	Apt 15	Boston	MA	2120	USA
2	75 st. Alphonsus st.	Apt 305	Boston	MA	2120	USA
3	231 Park Drive	Apt 25	Boston	MA	2118	USA
4	706 Huntington Ave	Apt 412	Boston	MA	2120	USA
5	120 Parker Hill Avenue	Apt 305	Boston	MA	2120	USA
6	175 Broadway Street	Apt 3008	New York	NY	3824	USA
7	Random Rd	Apt 1124	El Cajon	CA	92020	USA
8	Mt Berryman Road	Apt 69	Flagstone Creek	Queensland	4344	Australi

```
#    Time      Action                                                  Message            Duration / Fetch
208  00:31:25  SELECT * FROM mydb.recommendation LIMIT 0, 1000         2 row(s) returned  0.000 sec / 0.000 sec
209  00:35:48  SELECT * FROM mydb.address LIMIT 0, 1000                10 row(s) returned 0.000 sec / 0.000 sec
210  00:36:47  SELECT * FROM mydb.customer_has_address LIMIT 0, 1000   5 row(s) returned  0.000 sec / 0.000 sec
```

11. Customer_has_address: Bridge table to connect the address and customer tables

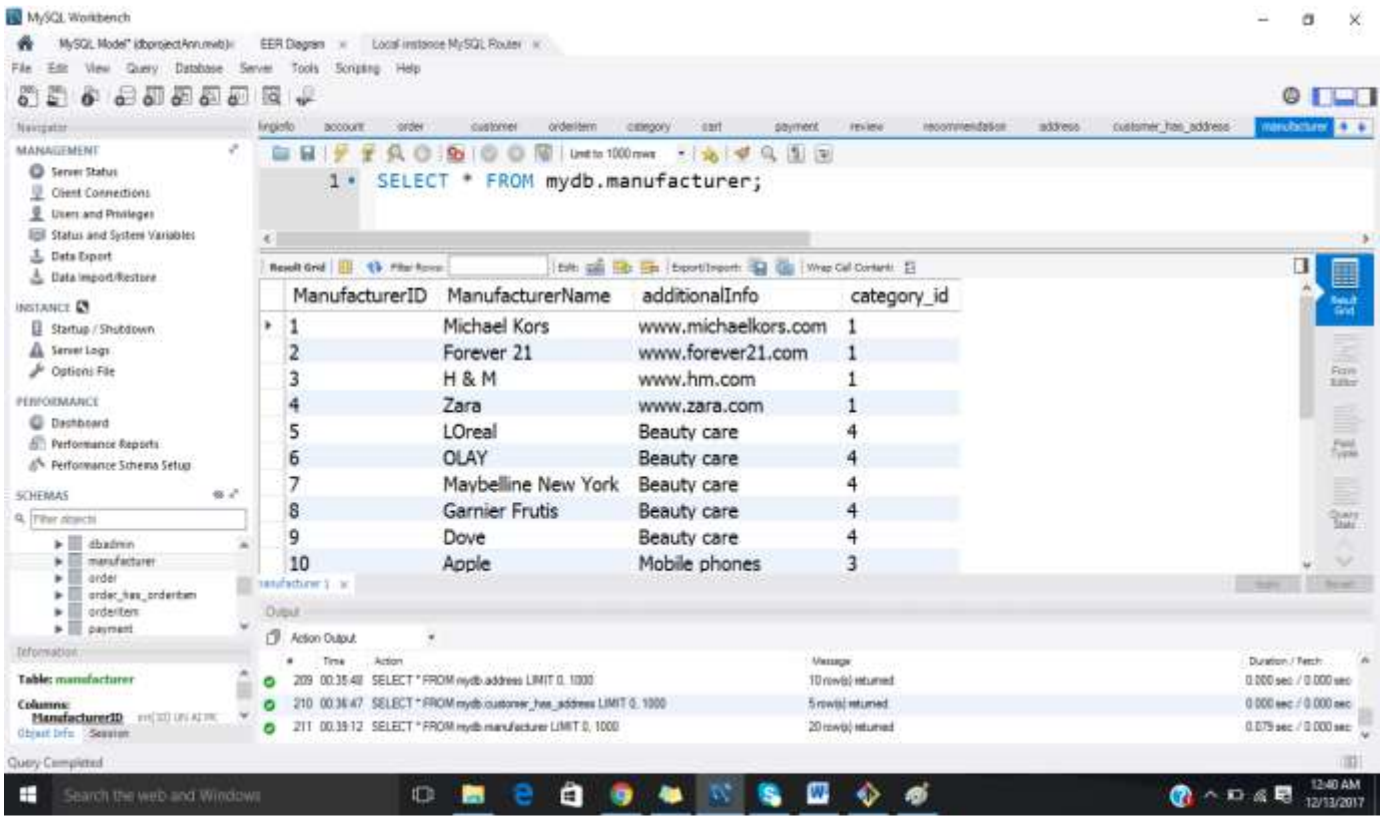


```
SELECT * FROM mydb.customer_has_address;
```

Customer_customer_id	address_addressID
1	1
2	2
3	3
4	4
5	5
NULL	NULL

```
#    Time      Action                                                      Message              Duration / Fetch
208  00:31:25  SELECT * FROM mydb.recommendation LIMIT 0, 1000             2 row(s) returned    0.000 sec / 0.000 sec
209  00:35:48  SELECT * FROM mydb.address LIMIT 0, 1000                    10 row(s) returned   0.000 sec / 0.000 sec
210  00:36:47  SELECT * FROM mydb.customer_has_address LIMIT 0, 1000       5 row(s) returned    0.000 sec / 0.000 sec
```

12. MANUFACTURER: A list of all manufacturers. It has the following attributes:

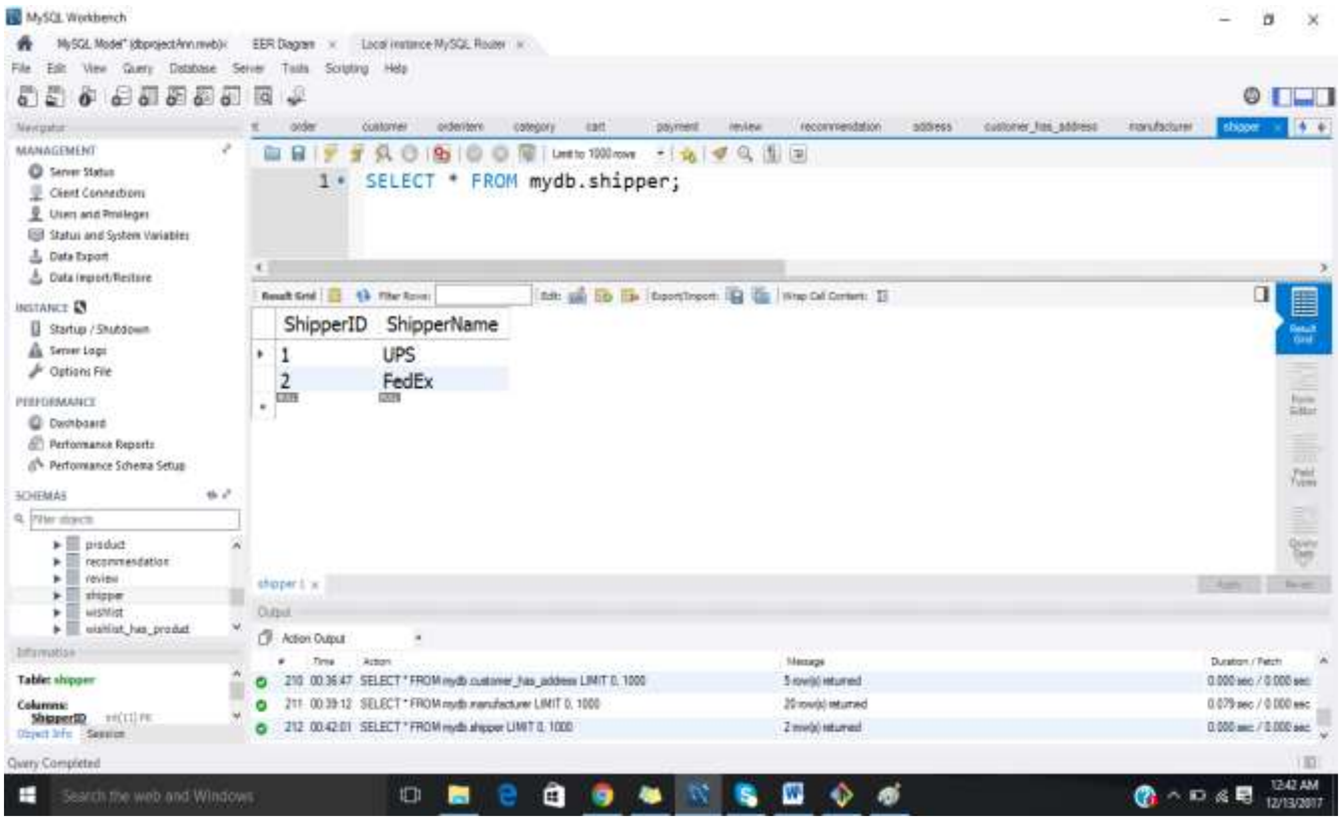


```
SELECT * FROM mydb.manufacturer;
```

ManufacturerID	ManufacturerName	additionalInfo	category_id
1	Michael Kors	www.michaelkors.com	1
2	Forever 21	www.forever21.com	1
3	H & M	www.hm.com	1
4	Zara	www.zara.com	1
5	LOreal	Beauty care	4
6	OLAY	Beauty care	4
7	Maybelline New York	Beauty care	4
8	Garnier Frutis	Beauty care	4
9	Dove	Beauty care	4
10	Apple	Mobile phones	3

```
#    Time      Action                                                      Message              Duration / Fetch
209  00:35:48  SELECT * FROM mydb.address LIMIT 0, 1000                    10 row(s) returned   0.000 sec / 0.000 sec
210  00:36:47  SELECT * FROM mydb.customer_has_address LIMIT 0, 1000       5 row(s) returned    0.000 sec / 0.000 sec
211  00:39:12  SELECT * FROM mydb.manufacturer LIMIT 0, 1000               20 row(s) returned   0.079 sec / 0.000 sec
```

13. SHIPPER: A shipper ships the customer its ordered product.



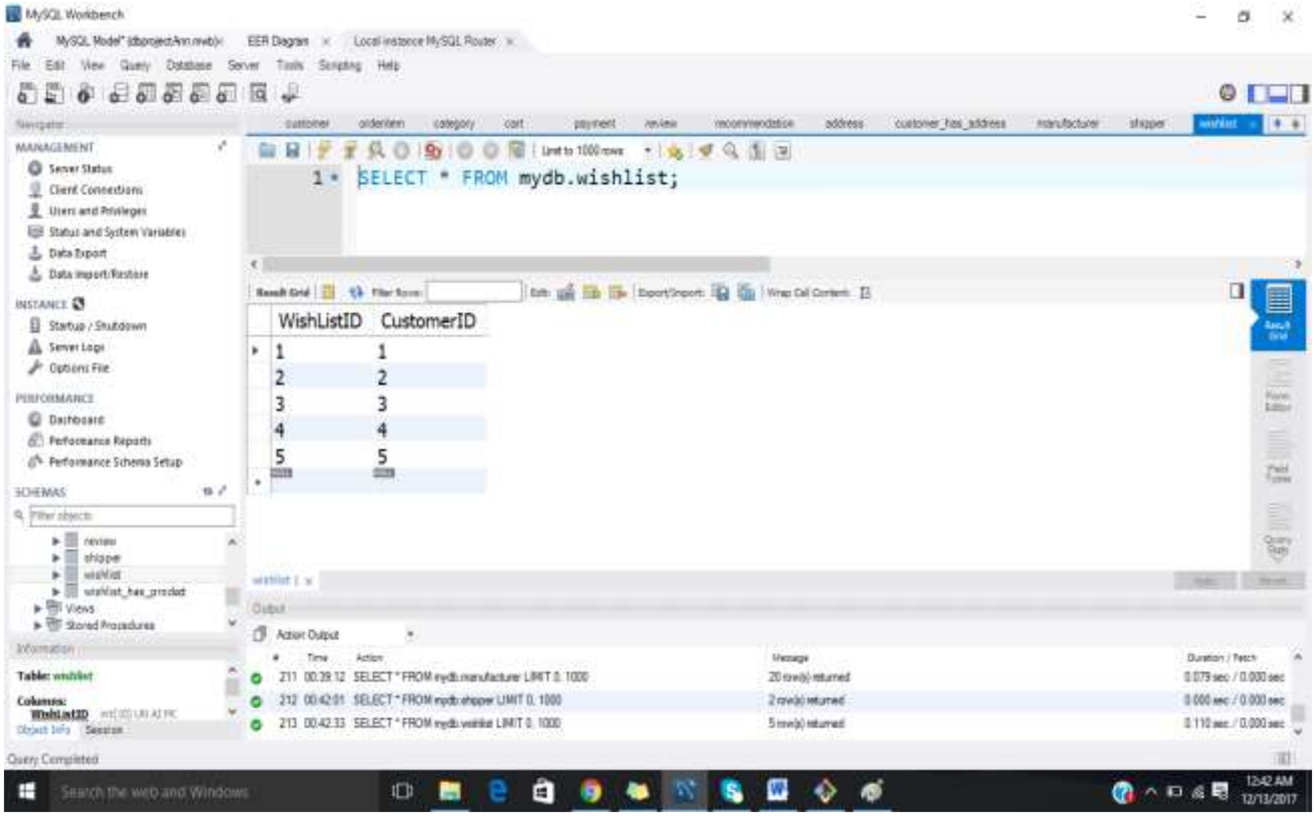
```
SELECT * FROM mydb.shipper;
```

ShipperID	ShipperName
1	UPS
2	FedEx
NULL	NULL

```
#    Time      Action                                                   Message            Duration / Fetch
210  00:36:47  SELECT * FROM mydb.customer_has_address LIMIT 0, 1000    5 row(s) returned  0.000 sec / 0.000 sec
211  00:39:12  SELECT * FROM mydb.manufacturer LIMIT 0, 1000            20 row(s) returned 0.079 sec / 0.000 sec
212  00:42:01  SELECT * FROM mydb.shipper LIMIT 0, 1000                 2 row(s) returned  0.000 sec / 0.000 sec
```

Query Completed

14. WISHLIST: Every customer has a wishlist where he/she can add products

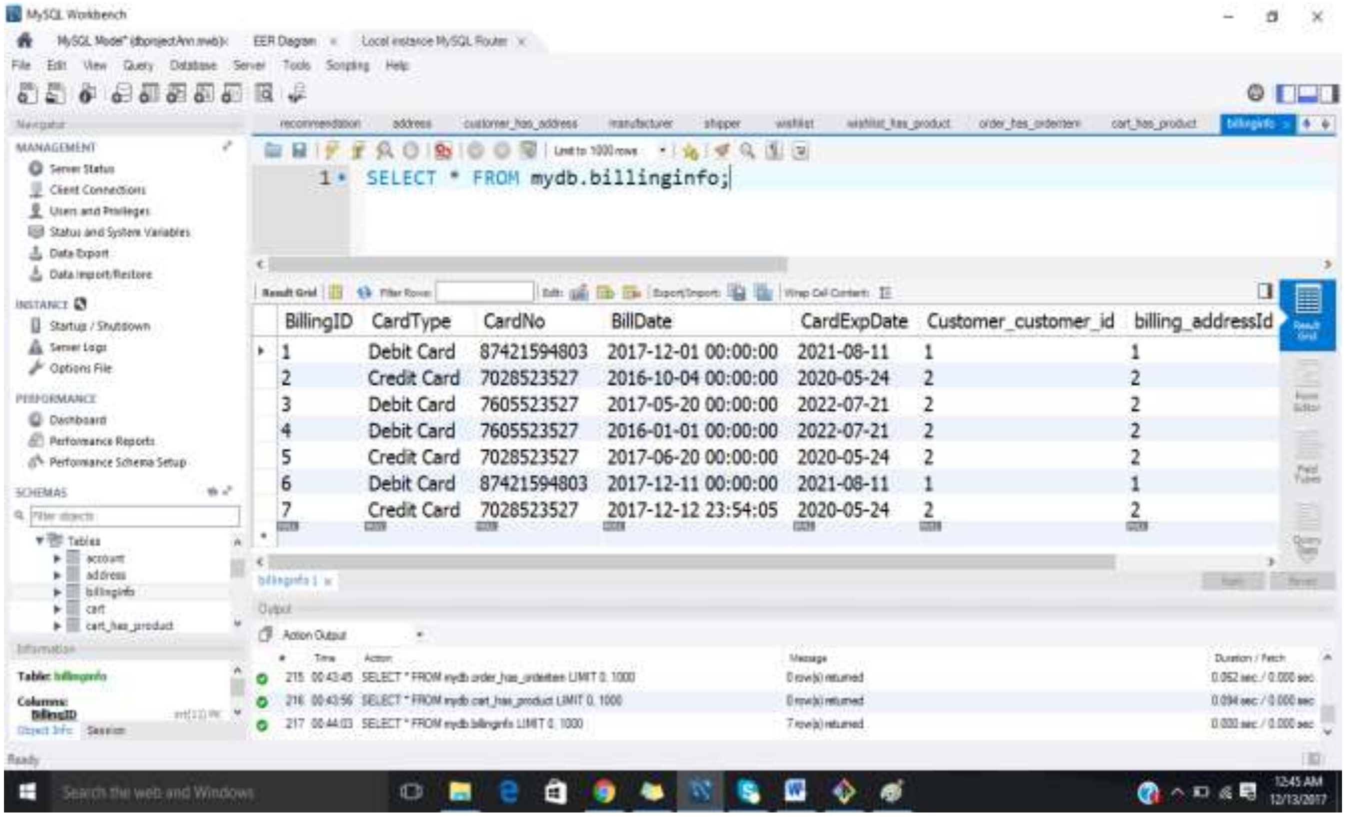


```
SELECT * FROM mydb.wishlist;
```

WishListID	CustomerID
1	1
2	2
3	3
4	4
5	5
NULL	NULL

```
#    Time      Action                                           Message            Duration / Fetch
211  00:39:12  SELECT * FROM mydb.manufacturer LIMIT 0, 1000    20 row(s) returned 0.079 sec / 0.000 sec
212  00:42:01  SELECT * FROM mydb.shipper LIMIT 0, 1000         2 row(s) returned  0.000 sec / 0.000 sec
213  00:42:33  SELECT * FROM mydb.wishlist LIMIT 0, 1000        5 row(s) returned  0.110 sec / 0.000 sec
```

15. BILLINGINFO: There is a separate table for billingInfo about each customer. It has the following attributes:



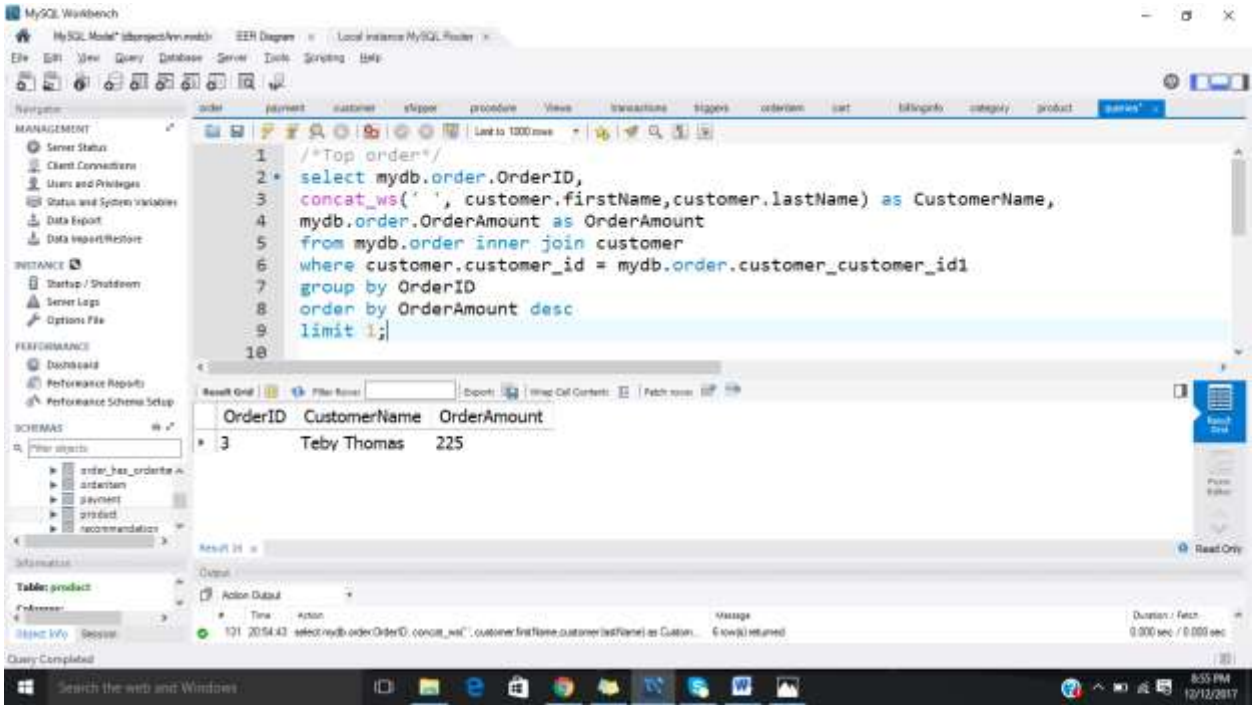
```
SELECT * FROM mydb.billinginfo;
```

BillingID	CardType	CardNo	BillDate	CardExpDate	Customer_customer_id	billing_addressId
1	Debit Card	87421594803	2017-12-01 00:00:00	2021-08-11	1	1
2	Credit Card	7028523527	2016-10-04 00:00:00	2020-05-24	2	2
3	Debit Card	7605523527	2017-05-20 00:00:00	2022-07-21	2	2
4	Debit Card	7605523527	2016-01-01 00:00:00	2022-07-21	2	2
5	Credit Card	7028523527	2017-06-20 00:00:00	2020-05-24	2	2
6	Debit Card	87421594803	2017-12-11 00:00:00	2021-08-11	1	1
7	Credit Card	7028523527	2017-12-12 23:54:05	2020-05-24	2	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#    Time      Action                                             Message           Duration / Fetch
215  00:43:45  SELECT * FROM mydb.order_has_orderitem LIMIT 0, 1000  0 row(s) returned 0.062 sec / 0.000 sec
216  00:43:56  SELECT * FROM mydb.cart_has_product LIMIT 0, 1000     0 row(s) returned 0.094 sec / 0.000 sec
217  00:44:03  SELECT * FROM mydb.billinginfo LIMIT 0, 1000          7 row(s) returned 0.000 sec / 0.000 sec
```

ANALYTICAL QUERIES

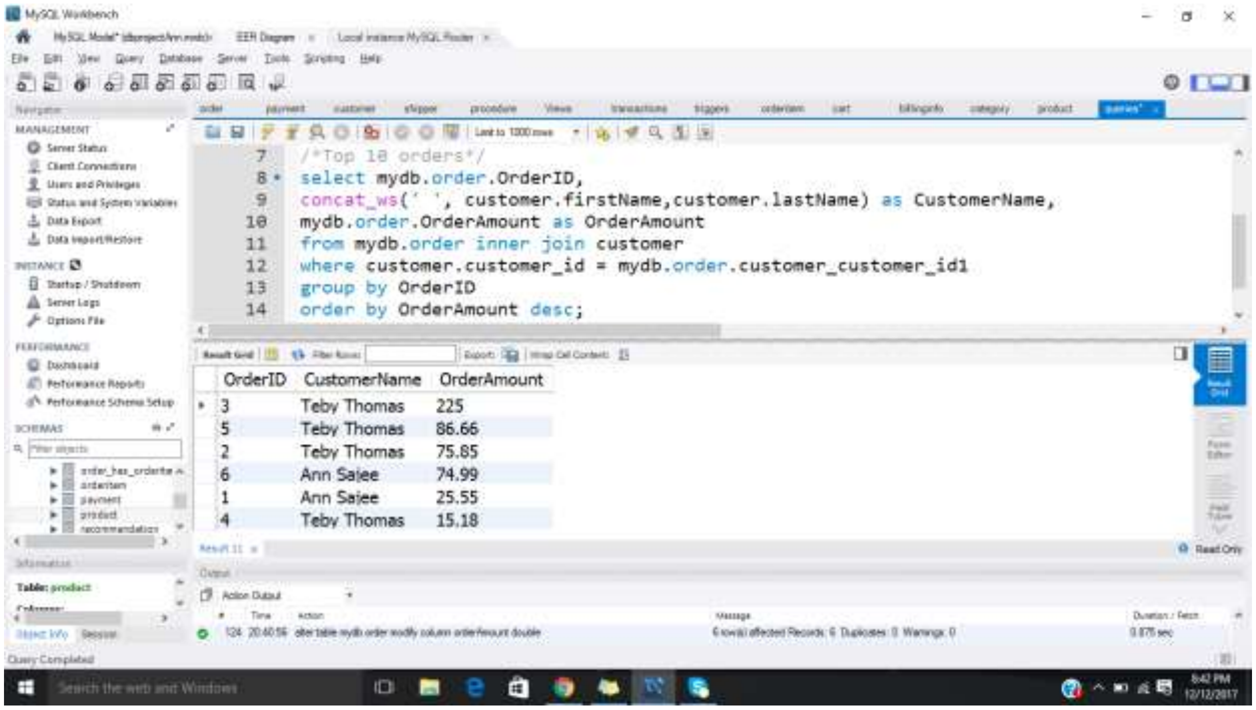
1. Highest order placed



```
/*Top order*/
select mydb.order.OrderID,
concat_ws(' ', customer.firstName,customer.lastName) as CustomerName,
mydb.order.OrderAmount as OrderAmount
from mydb.order inner join customer
where customer.customer_id = mydb.order.customer_customer_id1
group by OrderID
order by OrderAmount desc
limit 1;
```

OrderID	CustomerName	OrderAmount
3	Teby Thomas	225

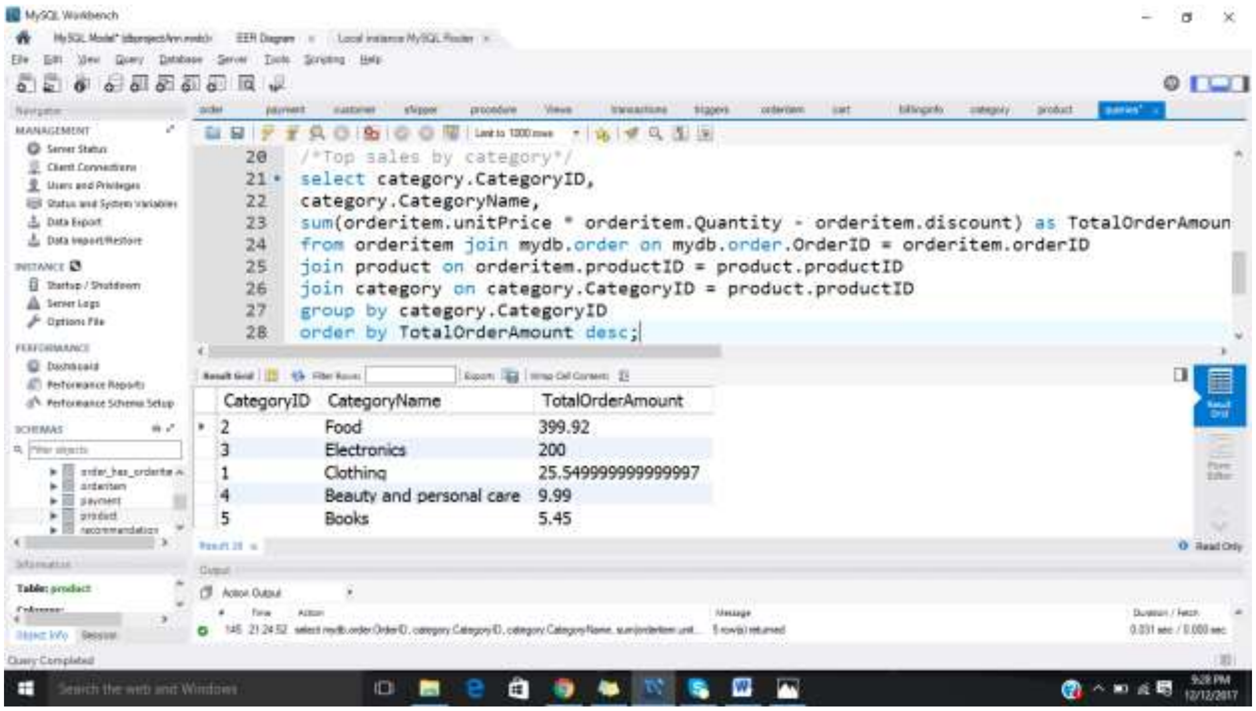
2. Top orders



```
/*Top 10 orders*/
select mydb.order.OrderID,
concat_ws(' ', customer.firstName,customer.lastName) as CustomerName,
mydb.order.OrderAmount as OrderAmount
from mydb.order inner join customer
where customer.customer_id = mydb.order.customer_customer_id1
group by OrderID
order by OrderAmount desc;
```

OrderID	CustomerName	OrderAmount
3	Teby Thomas	225
5	Teby Thomas	86.66
2	Teby Thomas	75.85
6	Ann Sajee	74.99
1	Ann Sajee	25.55
4	Teby Thomas	15.18

3. Top sales by category



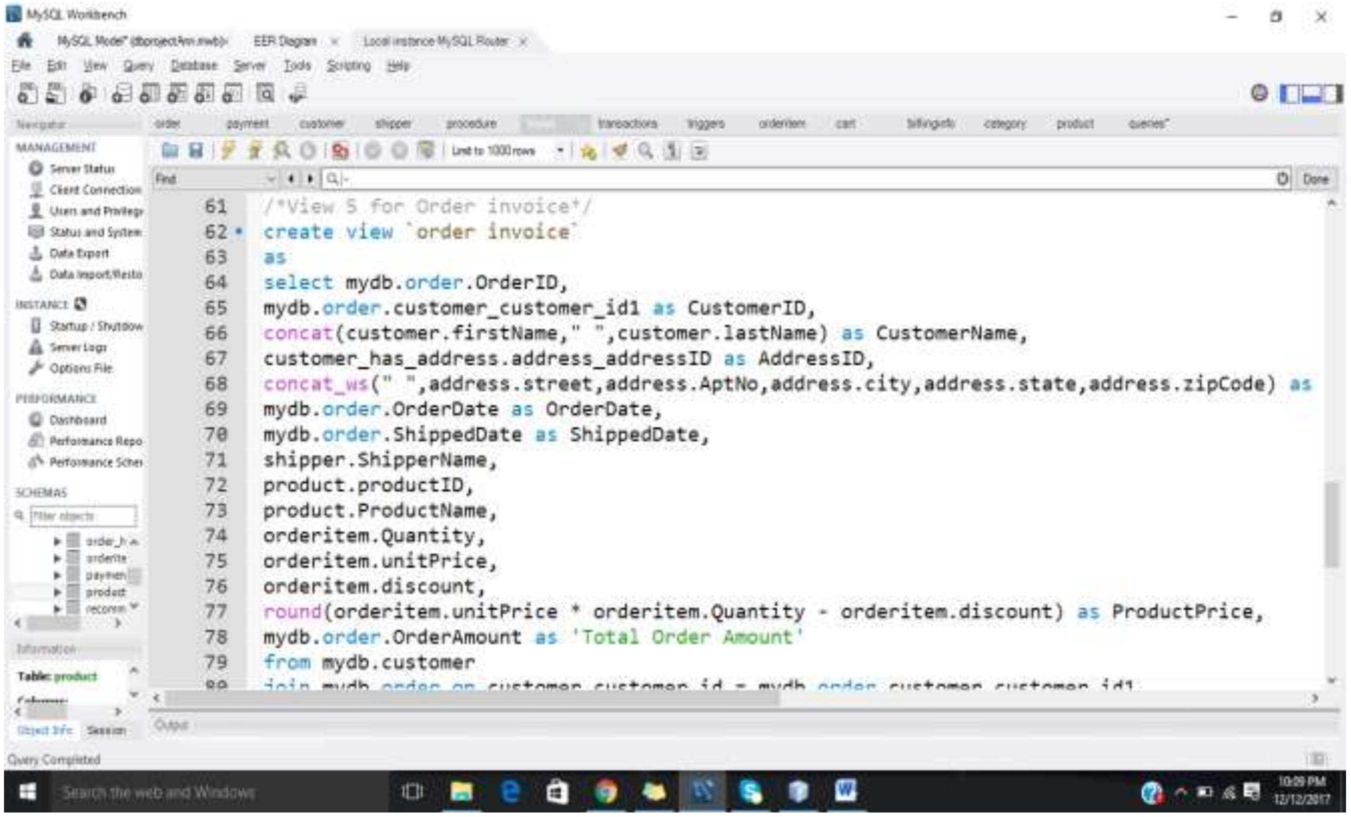
```sql
/*Top sales by category*/
select category.CategoryID,
category.CategoryName,
sum(orderitem.unitPrice * orderitem.Quantity - orderitem.discount) as TotalOrderAmoun
from orderitem join mydb.order on mydb.order.OrderID = orderitem.orderID
join product on orderitem.productID = product.productID
join category on category.CategoryID = product.productID
group by category.CategoryID
order by TotalOrderAmount desc;
```

CategoryID	CategoryName	TotalOrderAmount
2	Food	399.92
3	Electronics	200
1	Clothing	25.549999999999997
4	Beauty and personal care	9.99
5	Books	5.45

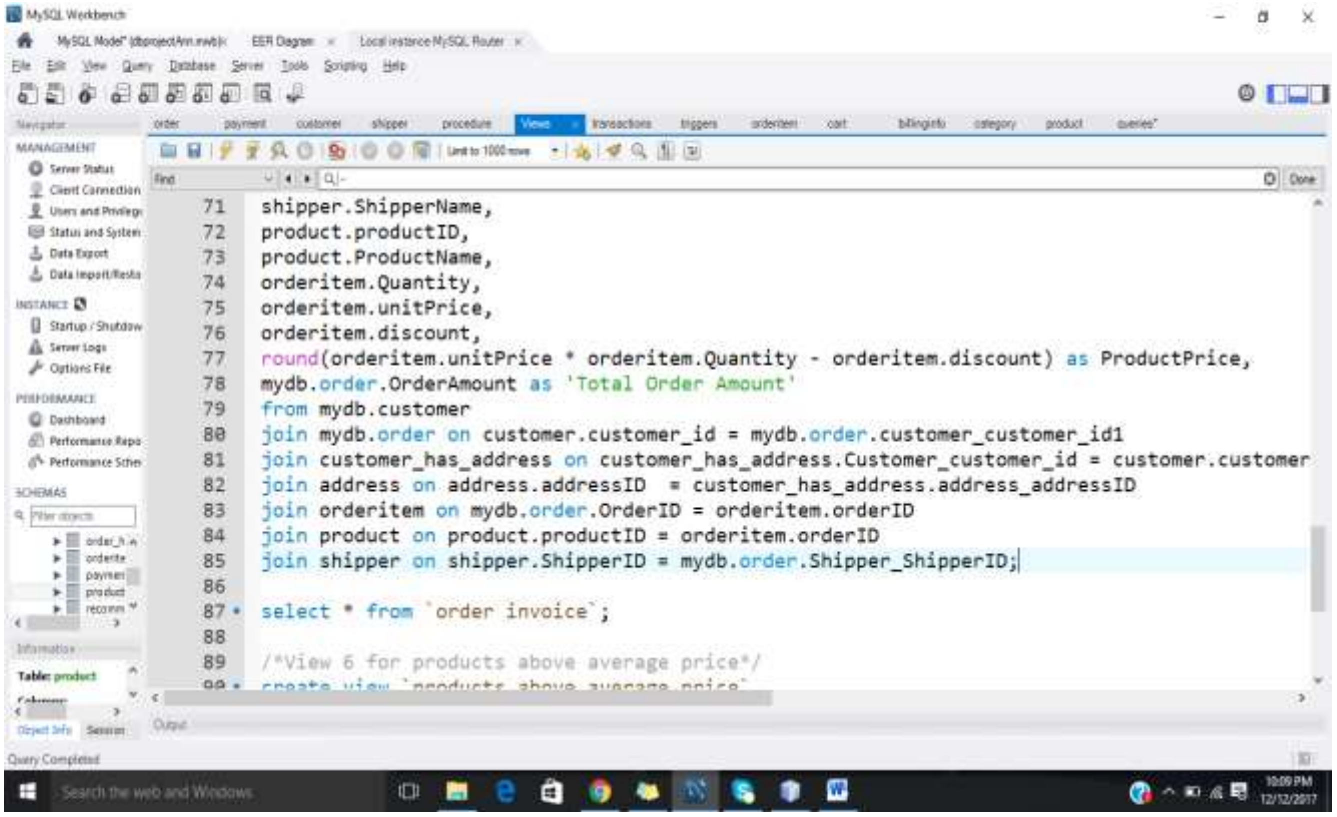
```
#    Time      Action                                                                          Message            Duration / Fetch
145  21:24:52  select mydb.order.OrderID, category.CategoryID, category.CategoryName, sum(orderitem.unit...  5 row(s) returned  0.031 sec / 0.000 sec
```

VIEWS

1. Order Invoice



```
/*View 5 for Order invoice*/
create view `order invoice`
as
select mydb.order.OrderID,
mydb.order.customer_customer_id1 as CustomerID,
concat(customer.firstName," ",customer.lastName) as CustomerName,
customer_has_address.address_addressID as AddressID,
concat_ws(" ",address.street,address.AptNo,address.city,address.state,address.zipCode) as
mydb.order.OrderDate as OrderDate,
mydb.order.ShippedDate as ShippedDate,
shipper.ShipperName,
product.productID,
product.ProductName,
orderitem.Quantity,
orderitem.unitPrice,
orderitem.discount,
round(orderitem.unitPrice * orderitem.Quantity - orderitem.discount) as ProductPrice,
mydb.order.OrderAmount as 'Total Order Amount'
from mydb.customer
join mydb.order on customer.customer_id = mydb.order.customer_customer_id1
```

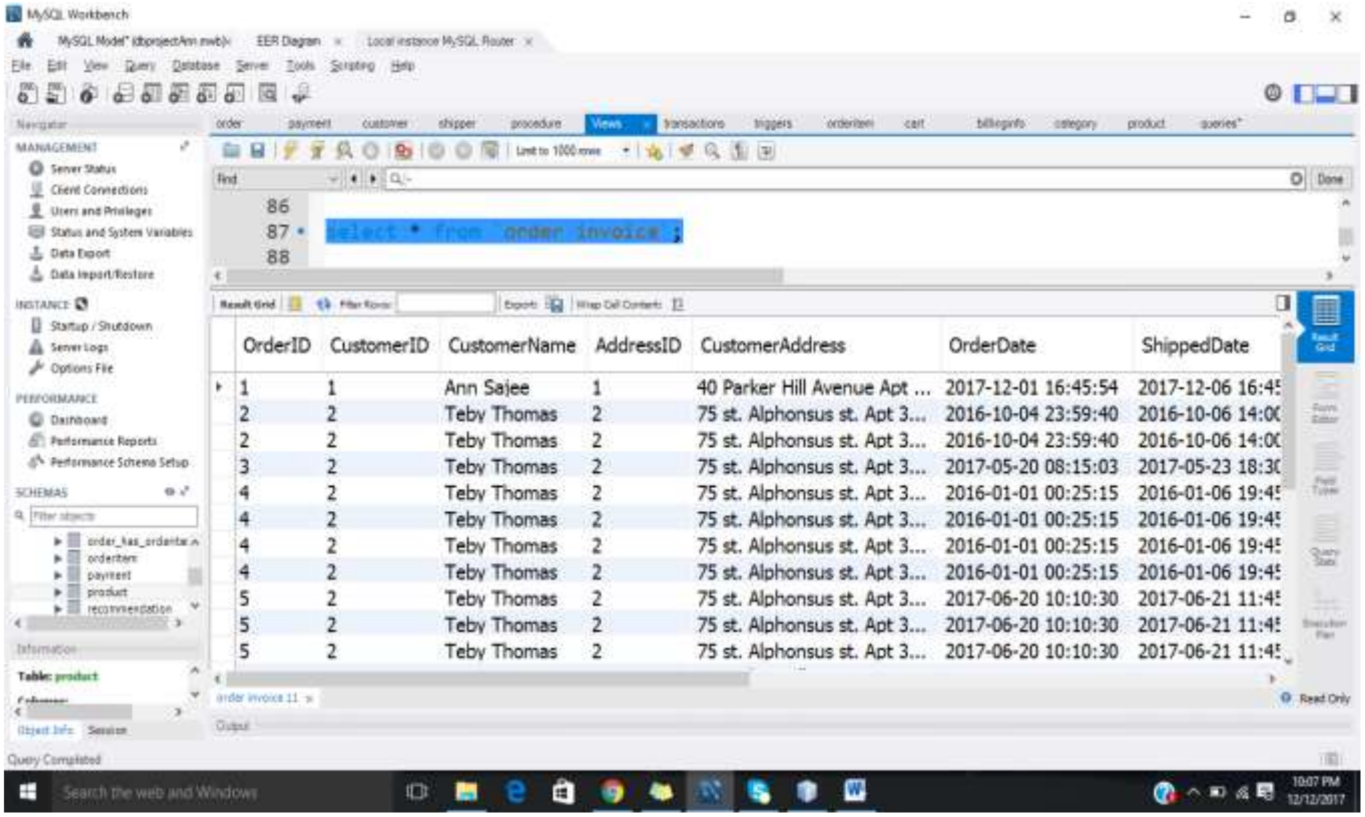


```
shipper.ShipperName,
product.productID,
product.ProductName,
orderitem.Quantity,
orderitem.unitPrice,
orderitem.discount,
round(orderitem.unitPrice * orderitem.Quantity - orderitem.discount) as ProductPrice,
mydb.order.OrderAmount as 'Total Order Amount'
from mydb.customer
join mydb.order on customer.customer_id = mydb.order.customer_customer_id1
join customer_has_address on customer_has_address.Customer_customer_id = customer.customer
join address on address.addressID  = customer_has_address.address_addressID
join orderitem on mydb.order.OrderID = orderitem.orderID
join product on product.productID = orderitem.orderID
join shipper on shipper.ShipperID = mydb.order.Shipper_ShipperID;

select * from `order invoice`;

/*View 6 for products above average price*/
create view `products above average price`
```

OUTPUT:

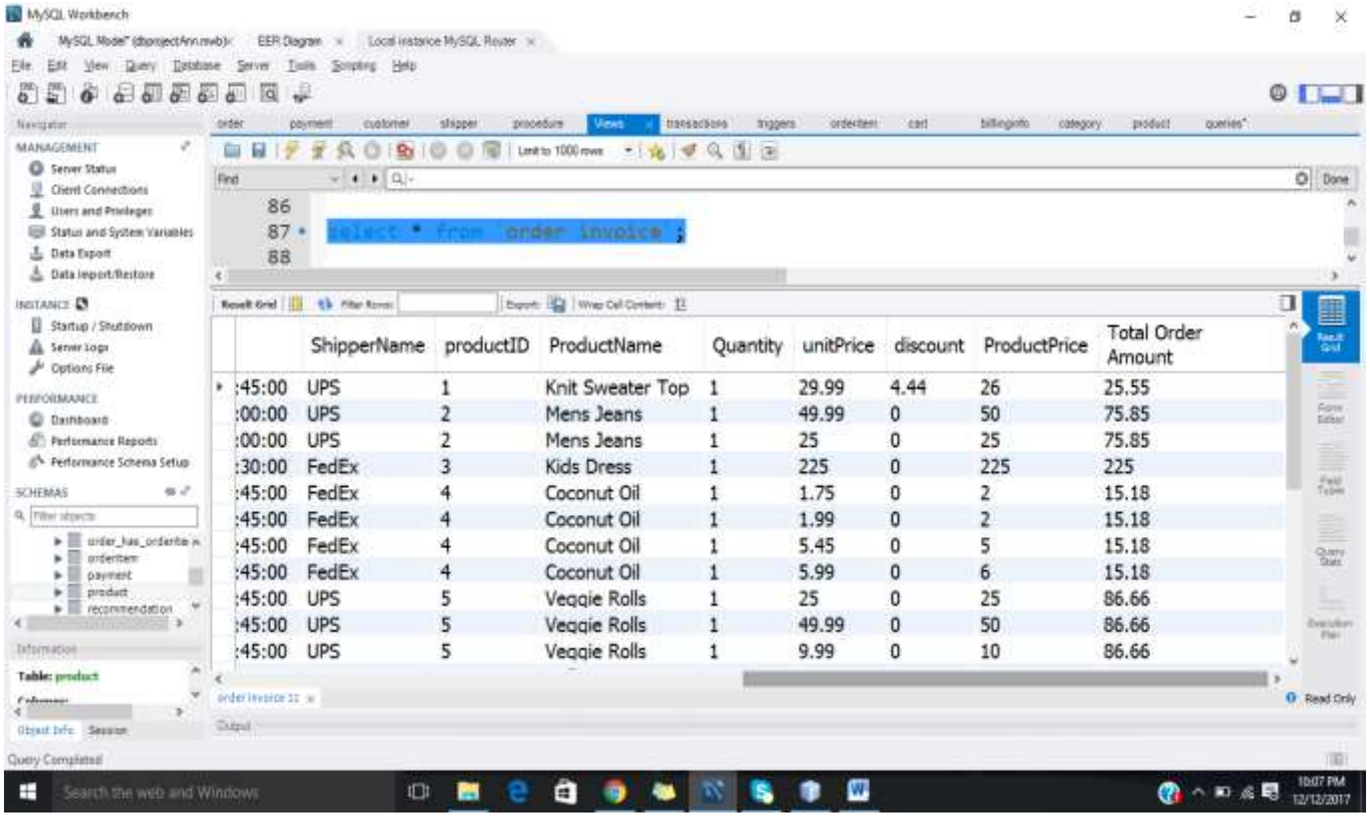


```
select * from `order invoice`;
```

	OrderID	CustomerID	CustomerName	AddressID	CustomerAddress	OrderDate	ShippedDate
▸	1	1	Ann Sajee	1	40 Parker Hill Avenue Apt ...	2017-12-01 16:45:54	2017-12-06 16:4
	2	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-10-04 23:59:40	2016-10-06 14:0
	2	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-10-04 23:59:40	2016-10-06 14:0
	3	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2017-05-20 08:15:03	2017-05-23 18:3
	4	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-01-01 00:25:15	2016-01-06 19:4
	4	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-01-01 00:25:15	2016-01-06 19:4
	4	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-01-01 00:25:15	2016-01-06 19:4
	4	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2016-01-01 00:25:15	2016-01-06 19:4
	5	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2017-06-20 10:10:30	2017-06-21 11:4
	5	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2017-06-20 10:10:30	2017-06-21 11:4
	5	2	Teby Thomas	2	75 st. Alphonsus st. Apt 3...	2017-06-20 10:10:30	2017-06-21 11:4

order invoice 11

Query Completed



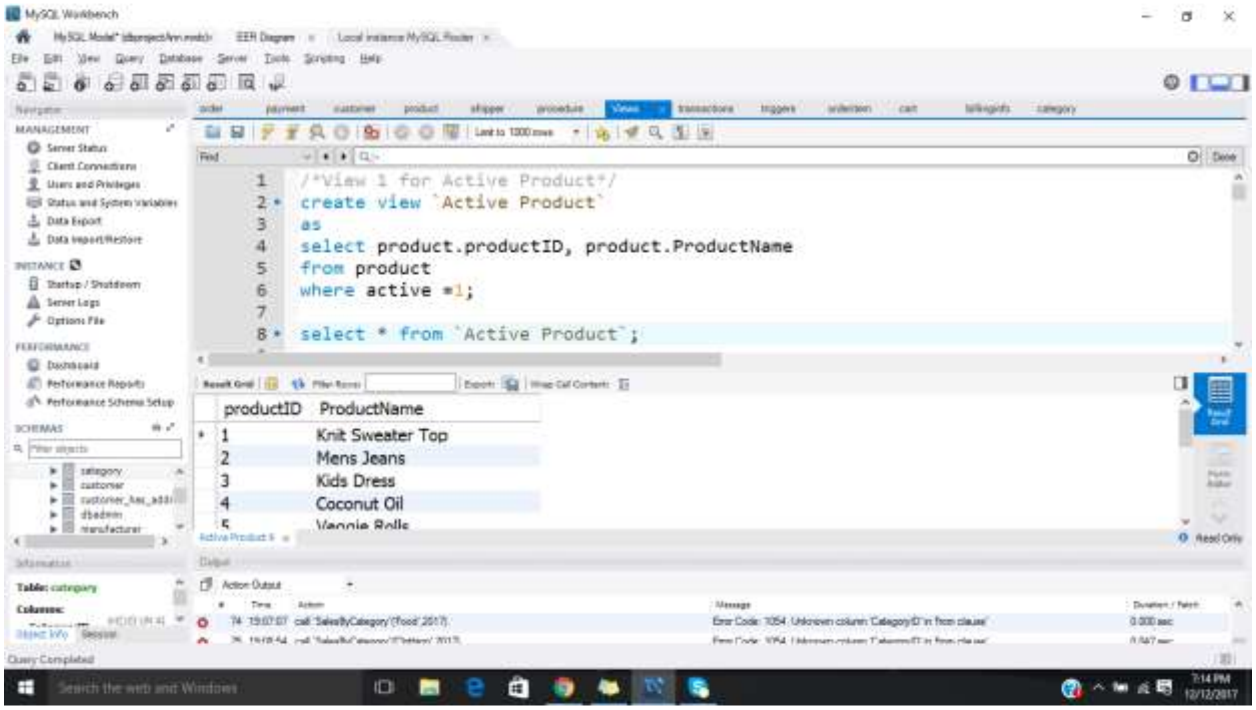
```
select * from `order invoice`;
```

		ShipperName	productID	ProductName	Quantity	unitPrice	discount	ProductPrice	Total Order Amount
▸	:45:00	UPS	1	Knit Sweater Top	1	29.99	4.44	26	25.55
	:00:00	UPS	2	Mens Jeans	1	49.99	0	50	75.85
	:00:00	UPS	2	Mens Jeans	1	25	0	25	75.85
	:30:00	FedEx	3	Kids Dress	1	225	0	225	225
	:45:00	FedEx	4	Coconut Oil	1	1.75	0	2	15.18
	:45:00	FedEx	4	Coconut Oil	1	1.99	0	2	15.18
	:45:00	FedEx	4	Coconut Oil	1	5.45	0	5	15.18
	:45:00	FedEx	4	Coconut Oil	1	5.99	0	6	15.18
	:45:00	UPS	5	Veggie Rolls	1	25	0	25	86.66
	:45:00	UPS	5	Veggie Rolls	1	49.99	0	50	86.66
	:45:00	UPS	5	Veggie Rolls	1	9.99	0	10	86.66

order invoice 11

Query Completed

2. List of active product



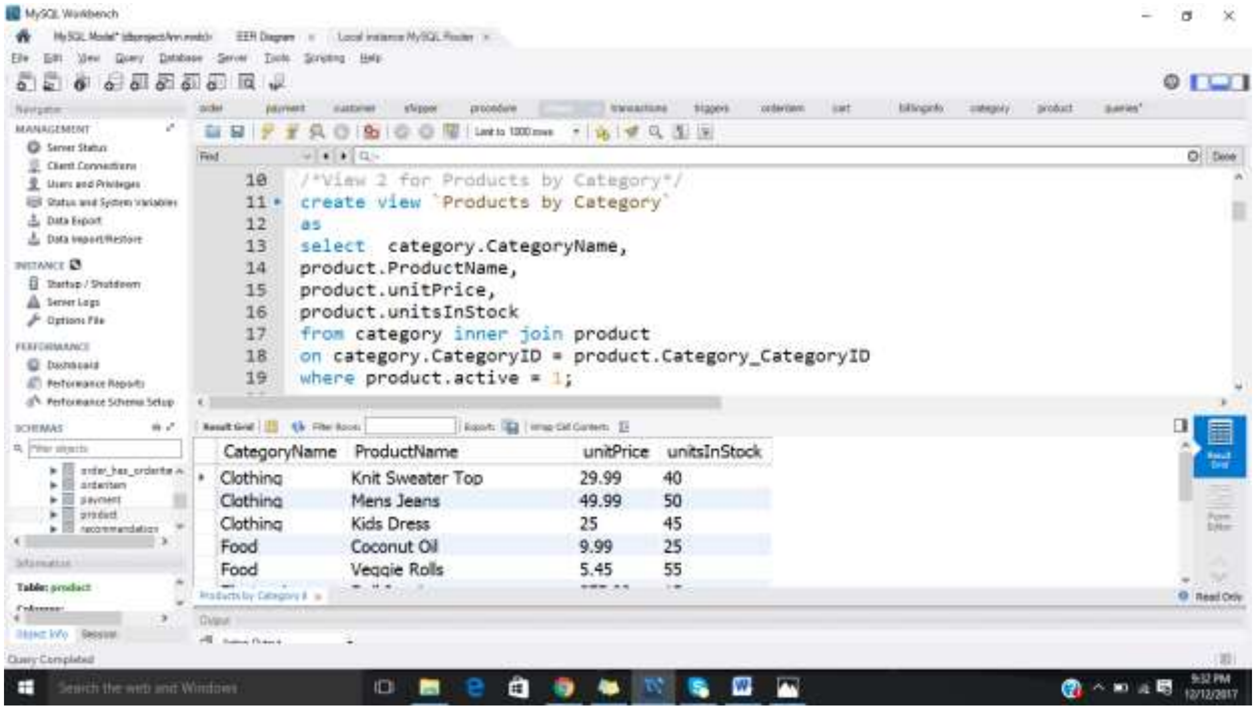
```
/*View 1 for Active Product*/
create view `Active Product`
as
select product.productID, product.ProductName
from product
where active =1;

select * from `Active Product`;
```

productID	ProductName
1	Knit Sweater Top
2	Mens Jeans
3	Kids Dress
4	Coconut Oil
5	Veggie Rolls

```
74 19:07:07 call `SalesByCategory`('Food' 2017)    Error Code: 1054. Unknown column 'CategoryID' in 'from clause'    0.000 sec
```

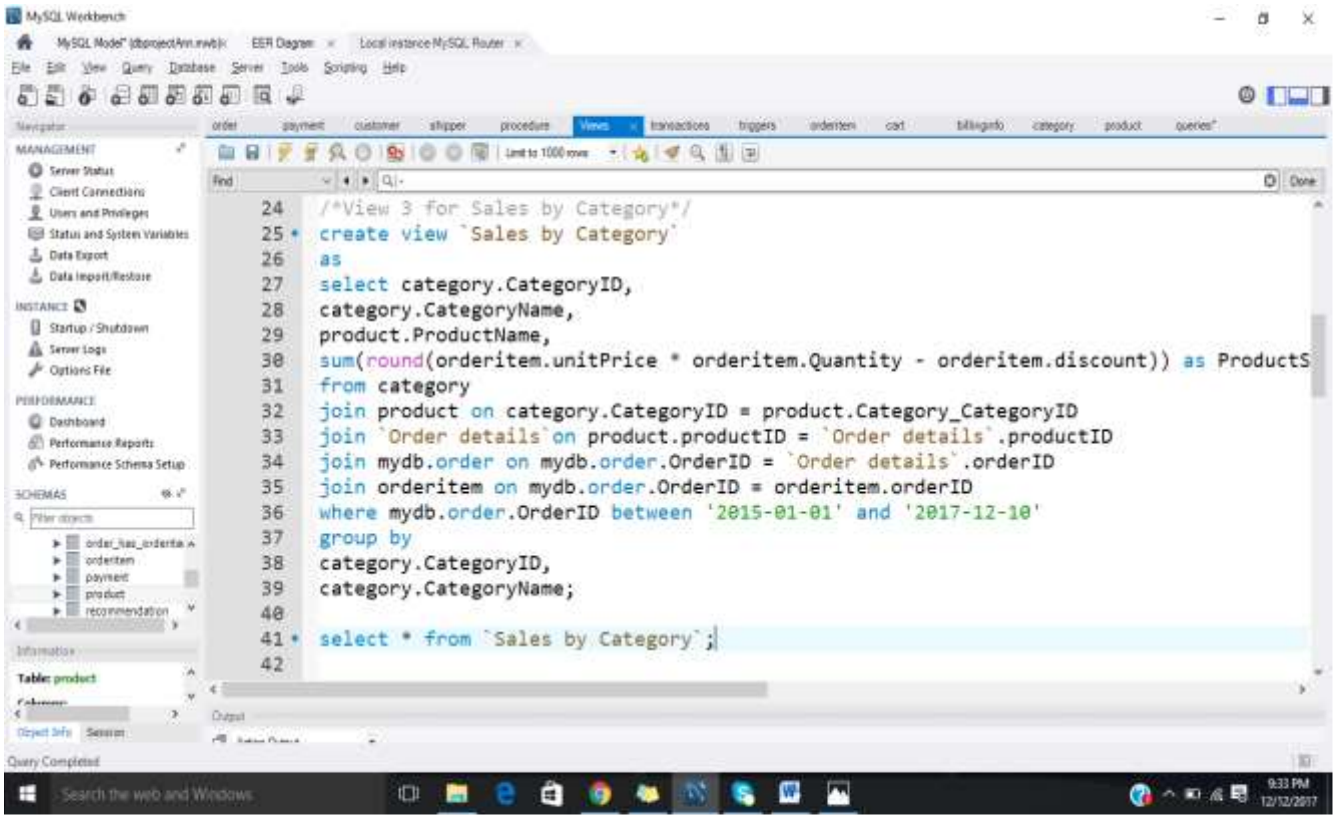
3. Products by category



```
/*View 2 for Products by Category*/
create view `Products by Category`
as
select  category.CategoryName,
product.ProductName,
product.unitPrice,
product.unitsInStock
from category inner join product
on category.CategoryID = product.Category_CategoryID
where product.active = 1;
```

CategoryName	ProductName	unitPrice	unitsInStock
Clothing	Knit Sweater Top	29.99	40
Clothing	Mens Jeans	49.99	50
Clothing	Kids Dress	25	45
Food	Coconut Oil	9.99	25
Food	Veggie Rolls	5.45	55

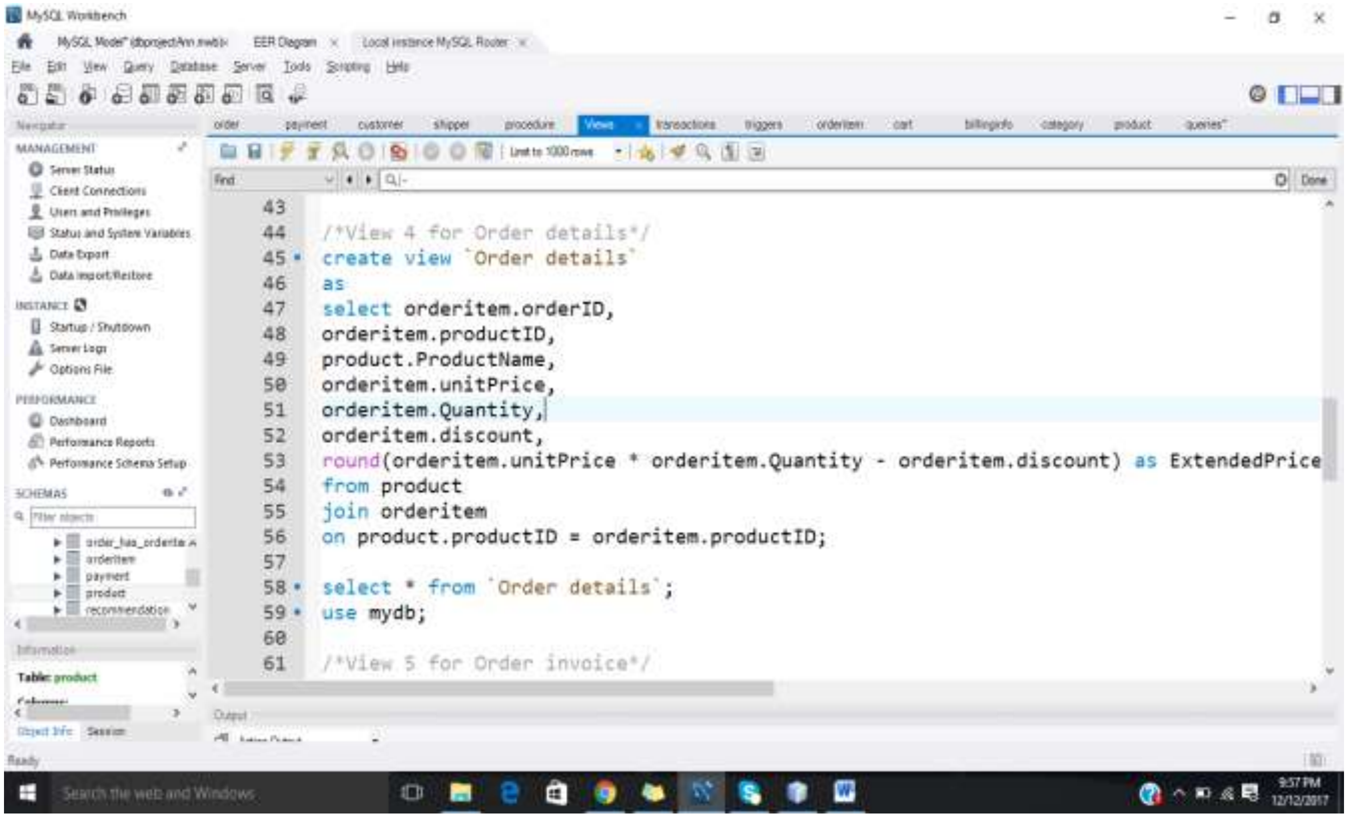
4. Sales by category



```
/*View 3 for Sales by Category*/
create view `Sales by Category`
as
select category.CategoryID,
category.CategoryName,
product.ProductName,
sum(round(orderitem.unitPrice * orderitem.Quantity - orderitem.discount)) as ProductS
from category
join product on category.CategoryID = product.Category_CategoryID
join `Order details` on product.productID = `Order details`.productID
join mydb.order on mydb.order.OrderID = `Order details`.orderID
join orderitem on mydb.order.OrderID = orderitem.orderID
where mydb.order.OrderID between '2015-01-01' and '2017-12-10'
group by
category.CategoryID,
category.CategoryName;

select * from `Sales by Category`;
```

5. Order details

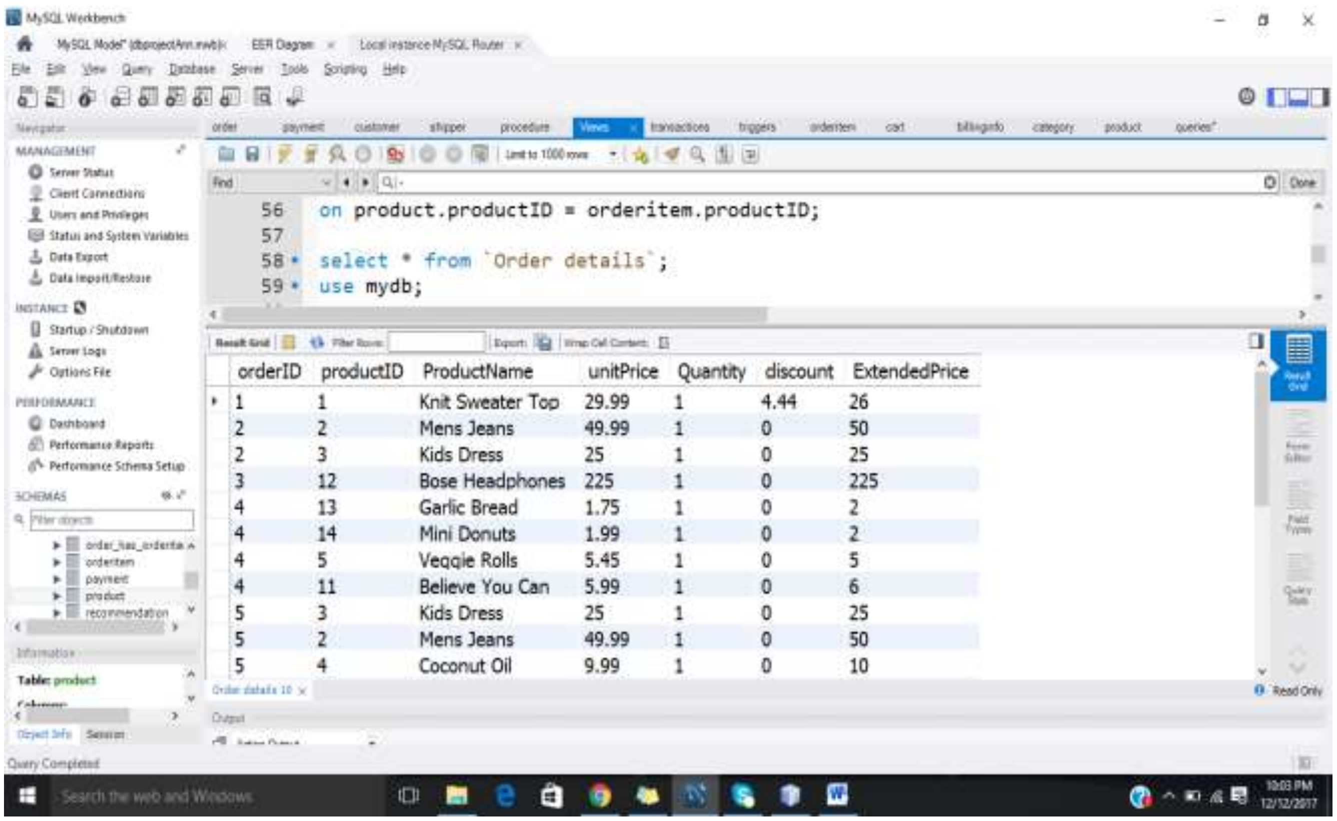


```

/*View 4 for Order details*/
create view `Order details`
as
select orderitem.orderID,
orderitem.productID,
product.ProductName,
orderitem.unitPrice,
orderitem.Quantity,
orderitem.discount,
round(orderitem.unitPrice * orderitem.Quantity - orderitem.discount) as ExtendedPrice
from product
join orderitem
on product.productID = orderitem.productID;

select * from `Order details`;
use mydb;

/*View 5 for Order invoice*/
```

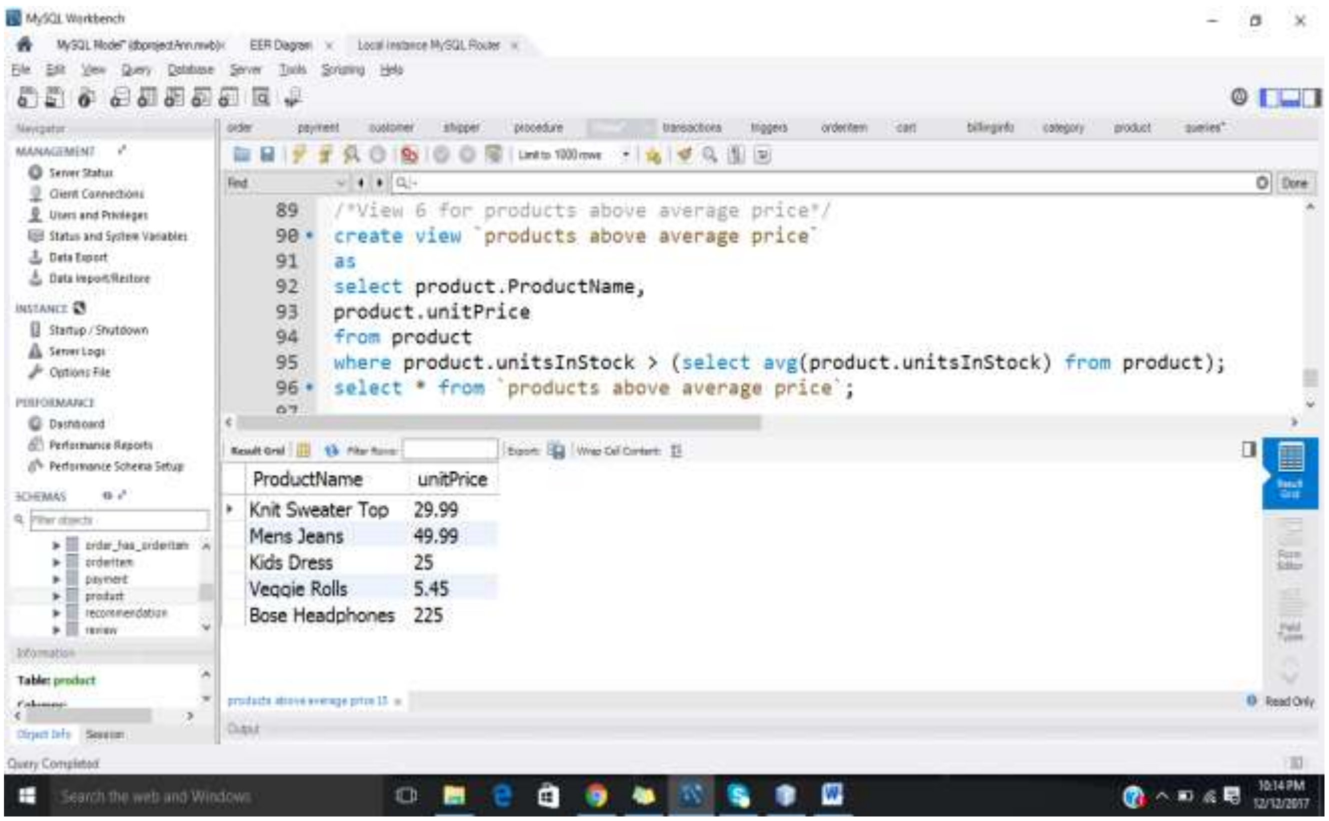


```sql
on product.productID = orderitem.productID;

select * from `Order details`;
use mydb;
```

orderID	productID	ProductName	unitPrice	Quantity	discount	ExtendedPrice
1	1	Knit Sweater Top	29.99	1	4.44	26
2	2	Mens Jeans	49.99	1	0	50
2	3	Kids Dress	25	1	0	25
3	12	Bose Headphones	225	1	0	225
4	13	Garlic Bread	1.75	1	0	2
4	14	Mini Donuts	1.99	1	0	2
4	5	Veggie Rolls	5.45	1	0	5
4	11	Believe You Can	5.99	1	0	6
5	3	Kids Dress	25	1	0	25
5	2	Mens Jeans	49.99	1	0	50
5	4	Coconut Oil	9.99	1	0	10

6. Product above average price

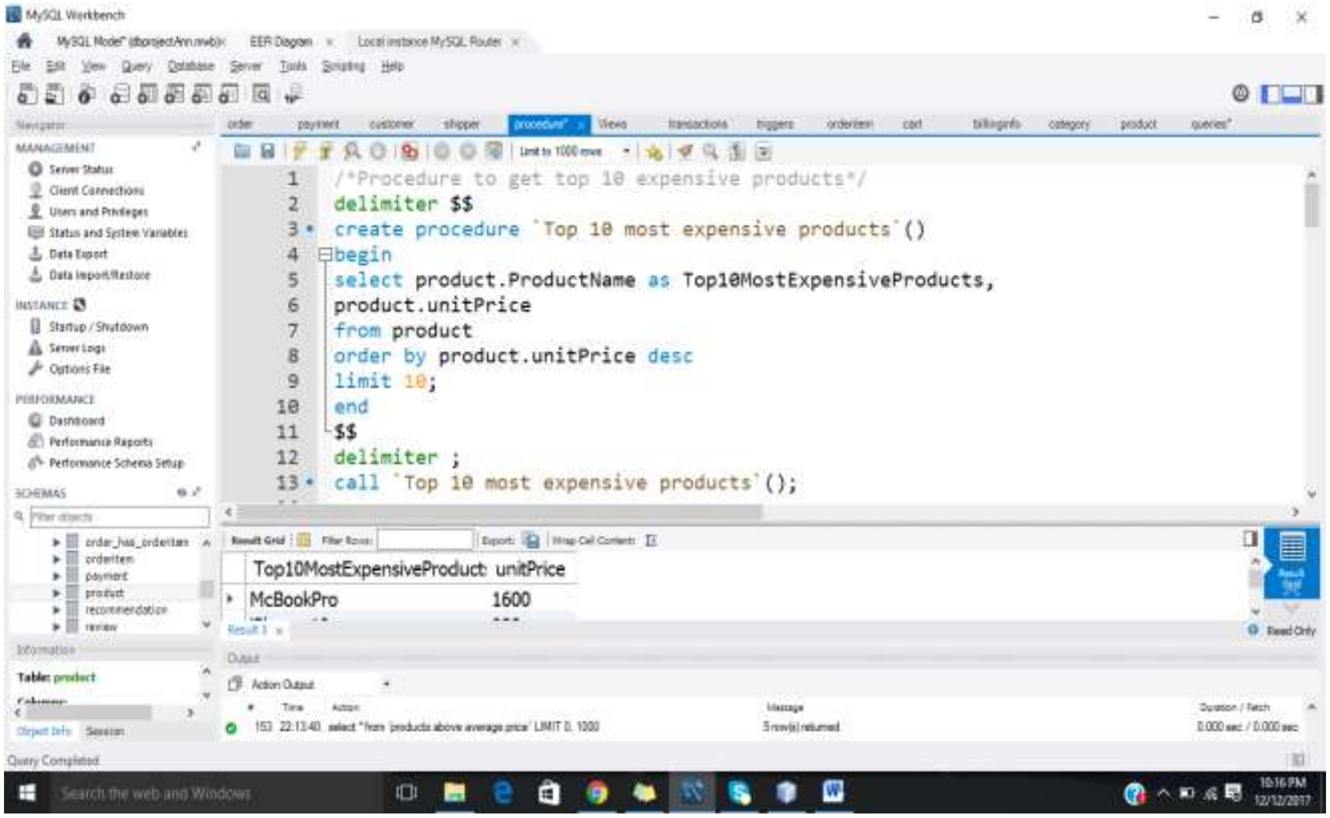


```sql
/*View 6 for products above average price*/
create view `products above average price`
as
select product.ProductName,
product.unitPrice
from product
where product.unitsInStock > (select avg(product.unitsInStock) from product);
select * from `products above average price`;
```

ProductName	unitPrice
Knit Sweater Top	29.99
Mens Jeans	49.99
Kids Dress	25
Veggie Rolls	5.45
Bose Headphones	225

STORED PROCEDURES

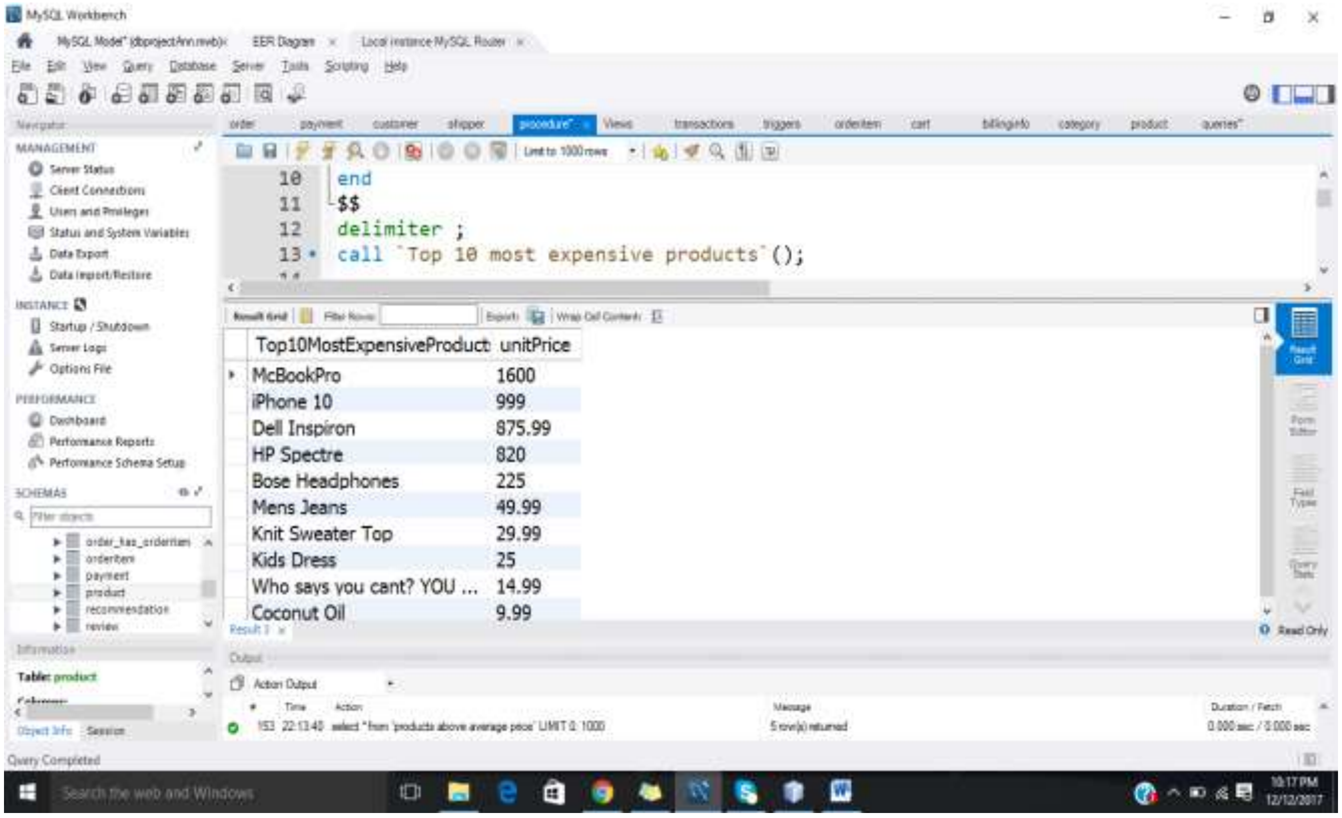
1. Procedure for Top 10 most expensive products



```
/*Procedure to get top 10 expensive products*/
delimiter $$
create procedure `Top 10 most expensive products`()
begin
select product.ProductName as Top10MostExpensiveProducts,
product.unitPrice
from product
order by product.unitPrice desc
limit 10;
end
$$
delimiter ;
call `Top 10 most expensive products`();
```

```
Top10MostExpensiveProduct  unitPrice
McBookPro                  1600
```

```
153  22:13:40  select * from `products above average price` LIMIT 0, 1000    5 row(s) returned    0.000 sec / 0.000 sec
```

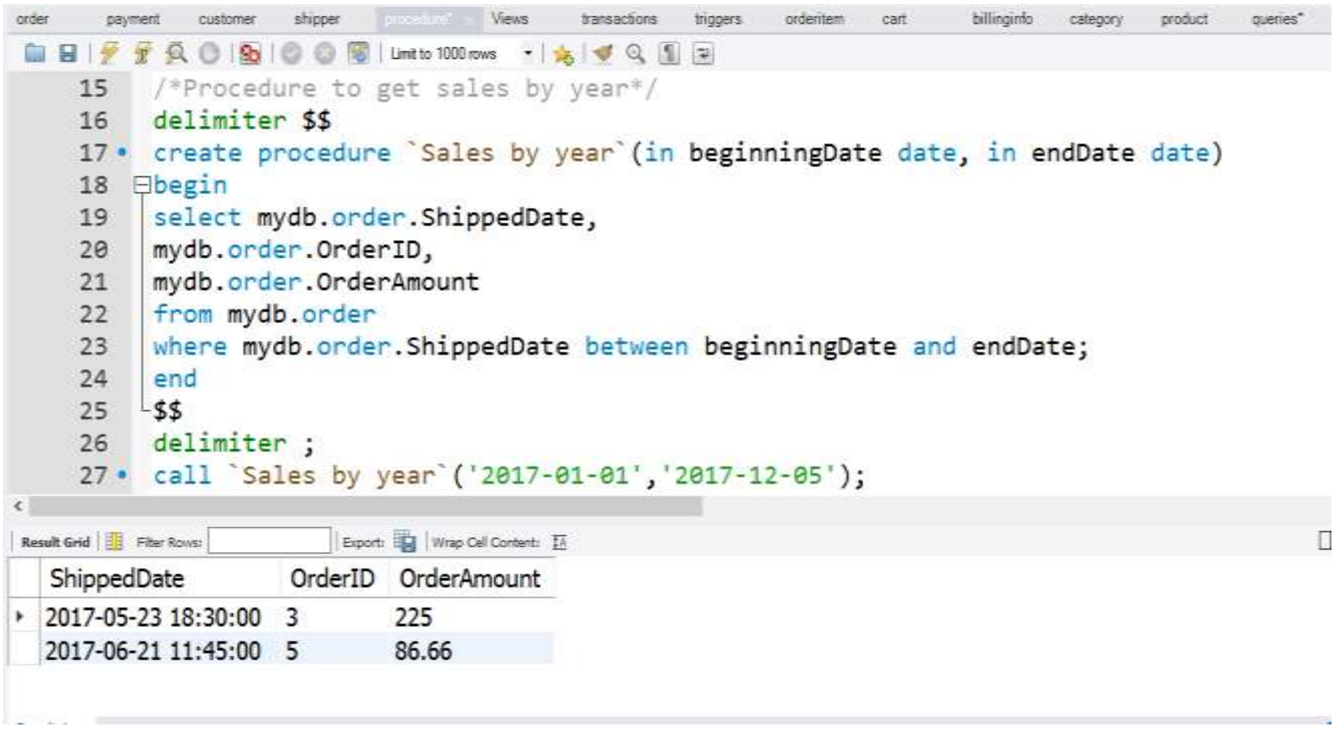


```
end
$$
delimiter ;
call `Top 10 most expensive products`();
```

Top10MostExpensiveProduct	unitPrice
McBookPro	1600
iPhone 10	999
Dell Inspiron	875.99
HP Spectre	820
Bose Headphones	225
Mens Jeans	49.99
Knit Sweater Top	29.99
Kids Dress	25
Who says you cant? YOU ...	14.99
Coconut Oil	9.99

```
153  22:13:40  select * from `products above average price` LIMIT 0, 1000    5 row(s) returned    0.000 sec / 0.000 sec
```

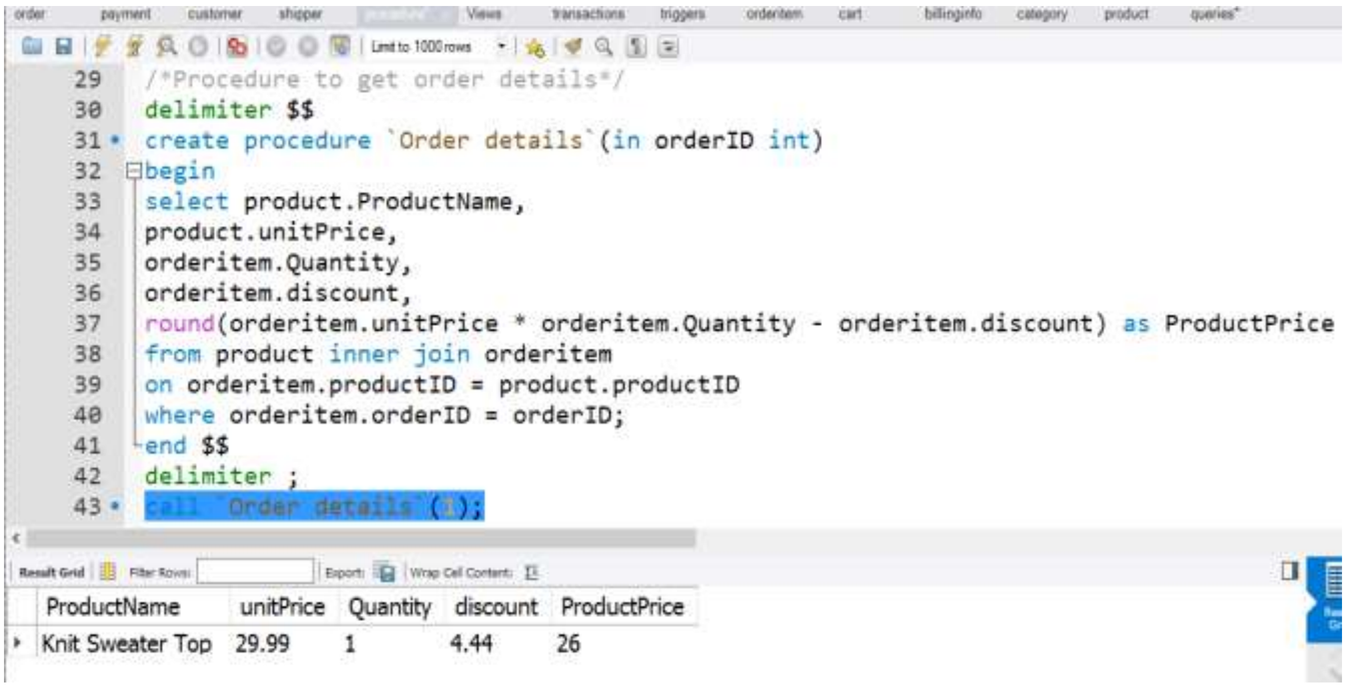
2. Procedure to get Sales by year



```
/*Procedure to get sales by year*/
delimiter $$
create procedure `Sales by year`(in beginningDate date, in endDate date)
begin
select mydb.order.ShippedDate,
mydb.order.OrderID,
mydb.order.OrderAmount
from mydb.order
where mydb.order.ShippedDate between beginningDate and endDate;
end
$$
delimiter ;
call `Sales by year`('2017-01-01','2017-12-05');
```

ShippedDate	OrderID	OrderAmount
2017-05-23 18:30:00	3	225
2017-06-21 11:45:00	5	86.66

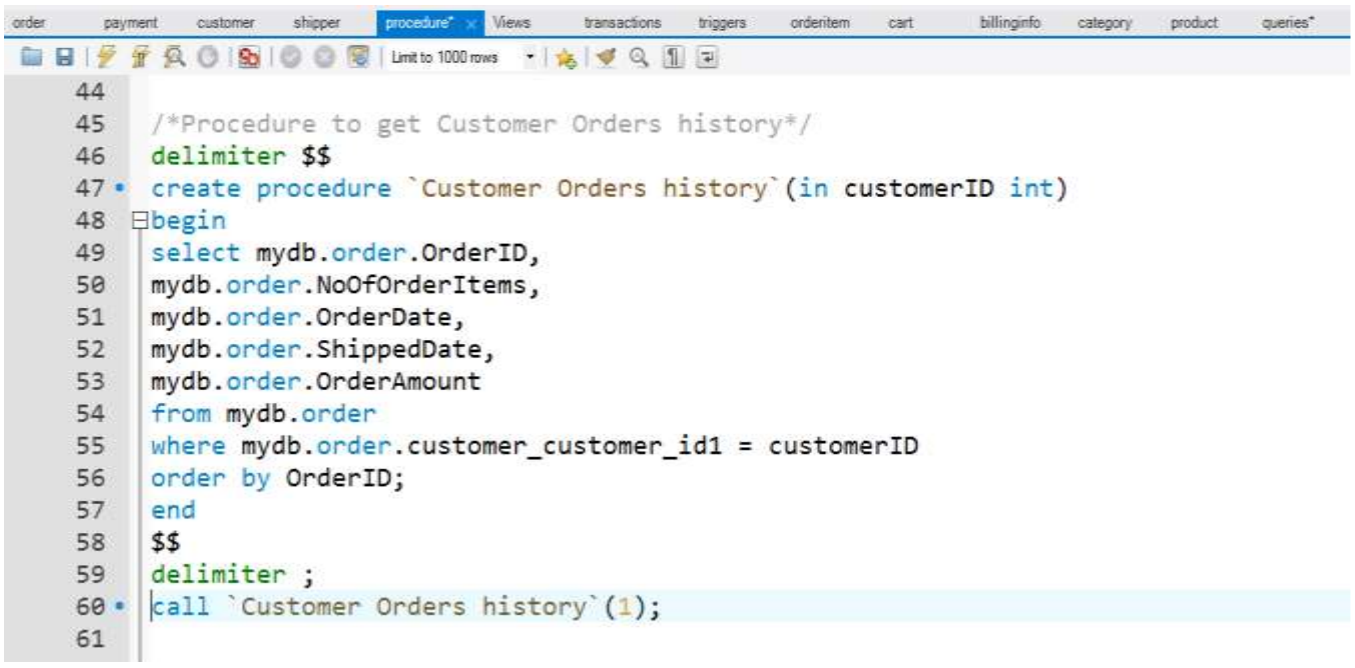
3. Procedure to get order details



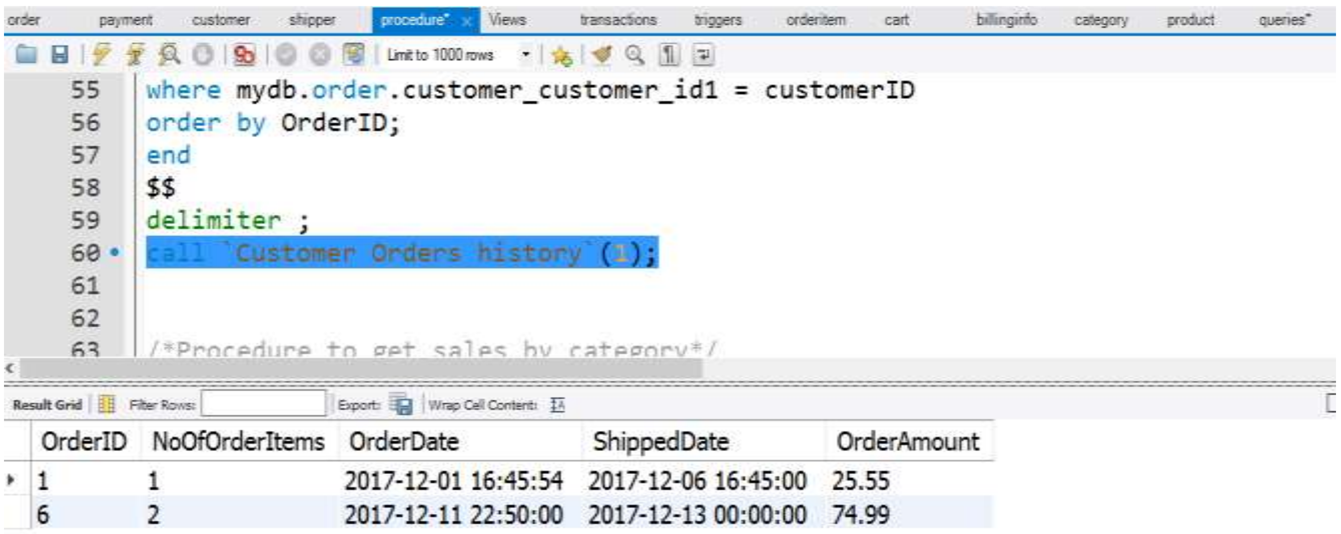
```
/*Procedure to get order details*/
delimiter $$
create procedure `Order details`(in orderID int)
begin
select product.ProductName,
product.unitPrice,
orderitem.Quantity,
orderitem.discount,
round(orderitem.unitPrice * orderitem.Quantity - orderitem.discount) as ProductPrice
from product inner join orderitem
on orderitem.productID = product.productID
where orderitem.orderID = orderID;
end $$
delimiter ;
call `Order details`(1);
```

ProductName	unitPrice	Quantity	discount	ProductPrice
Knit Sweater Top	29.99	1	4.44	26

4. Procedure to get customer order history



```
/*Procedure to get Customer Orders history*/
delimiter $$
create procedure `Customer Orders history`(in customerID int)
begin
select mydb.order.OrderID,
mydb.order.NoOfOrderItems,
mydb.order.OrderDate,
mydb.order.ShippedDate,
mydb.order.OrderAmount
from mydb.order
where mydb.order.customer_customer_id1 = customerID
order by OrderID;
end
$$
delimiter ;
call `Customer Orders history`(1);
```

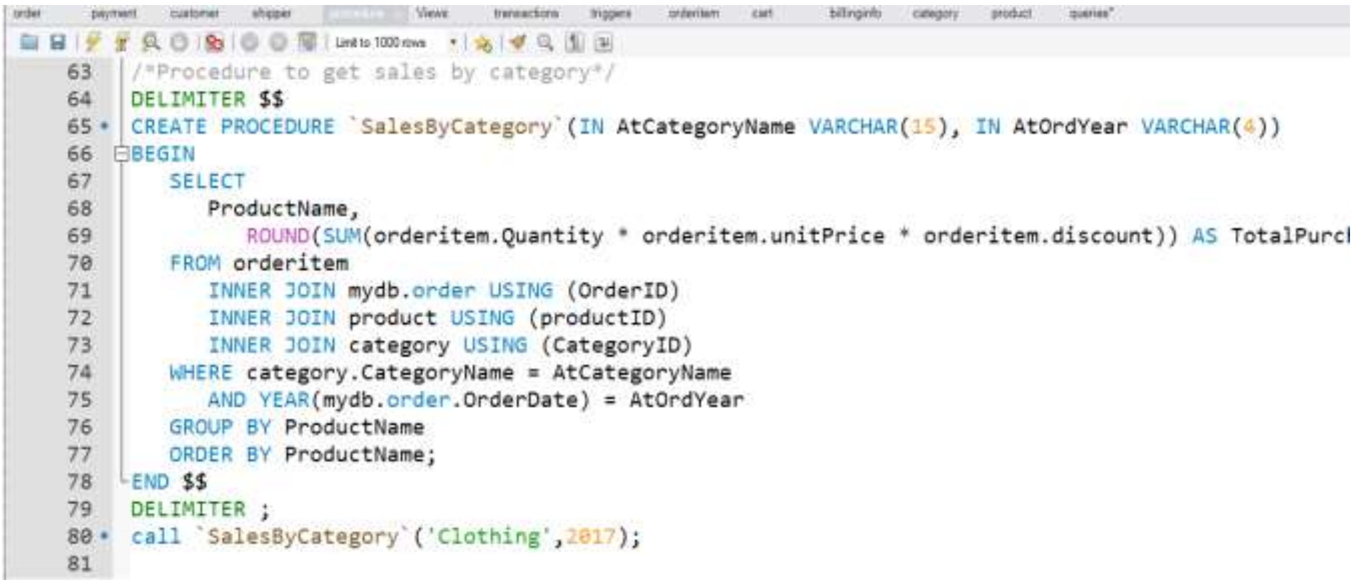


```
where mydb.order.customer_customer_id1 = customerID
order by OrderID;
end
$$
delimiter ;
call `Customer Orders history`(1);


/*Procedure to get sales by category*/
```

OrderID	NoOfOrderItems	OrderDate	ShippedDate	OrderAmount
1	1	2017-12-01 16:45:54	2017-12-06 16:45:00	25.55
6	2	2017-12-11 22:50:00	2017-12-13 00:00:00	74.99

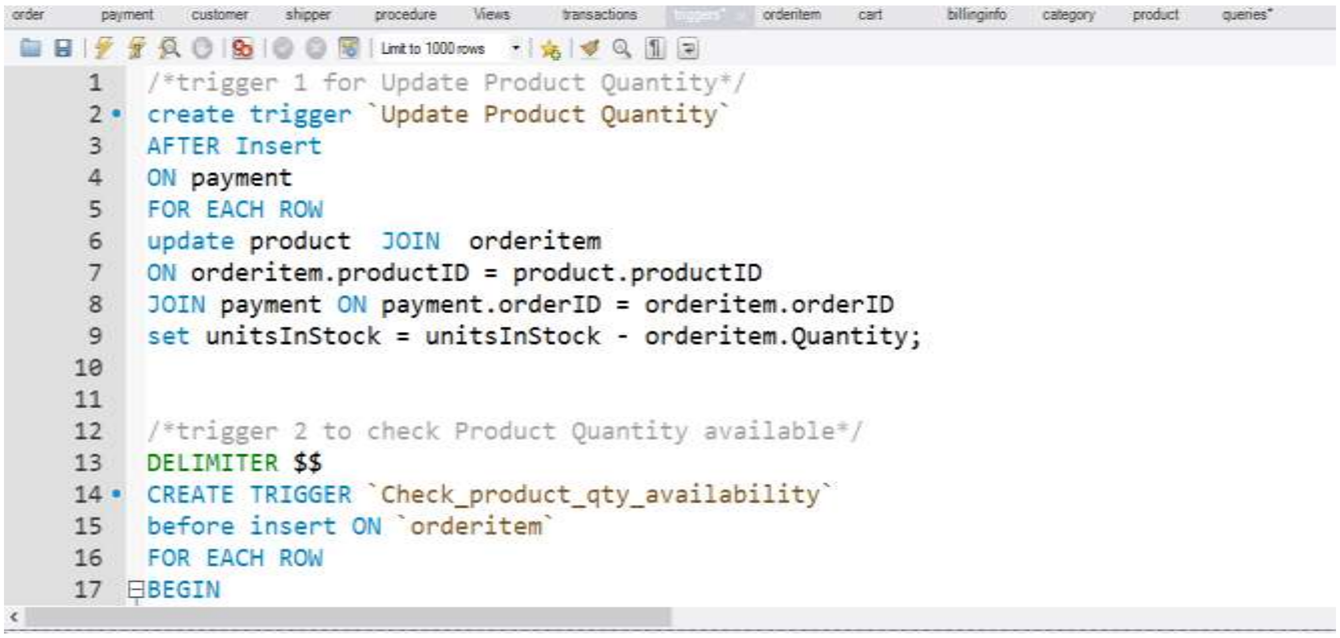
5. Sales by category



```sql
/*Procedure to get sales by category*/
DELIMITER $$
CREATE PROCEDURE `SalesByCategory`(IN AtCategoryName VARCHAR(15), IN AtOrdYear VARCHAR(4))
BEGIN
    SELECT
        ProductName,
            ROUND(SUM(orderitem.Quantity * orderitem.unitPrice * orderitem.discount)) AS TotalPurc
    FROM orderitem
        INNER JOIN mydb.order USING (OrderID)
        INNER JOIN product USING (productID)
        INNER JOIN category USING (CategoryID)
    WHERE category.CategoryName = AtCategoryName
        AND YEAR(mydb.order.OrderDate) = AtOrdYear
    GROUP BY ProductName
    ORDER BY ProductName;
END $$
DELIMITER ;
call `SalesByCategory`('Clothing',2017);
```

TRIGGERS

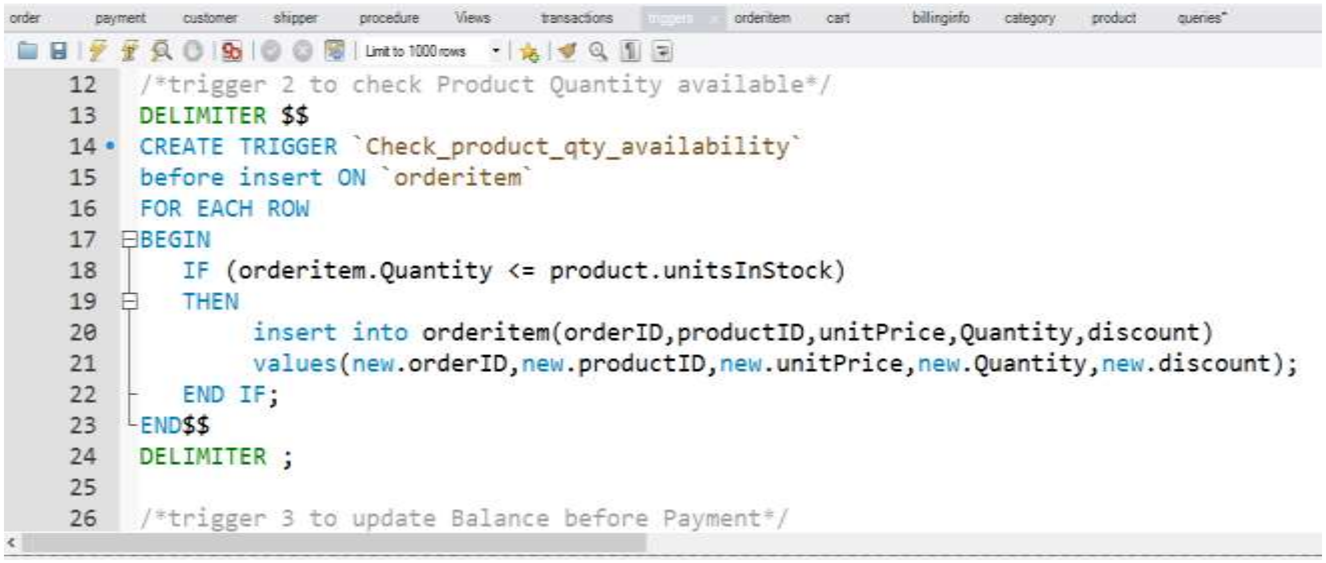
1. **Trigger to Update product Quantity**



```
/*trigger 1 for Update Product Quantity*/
create trigger `Update Product Quantity`
AFTER Insert
ON payment
FOR EACH ROW
update product  JOIN  orderitem
ON orderitem.productID = product.productID
JOIN payment ON payment.orderID = orderitem.orderID
set unitsInStock = unitsInStock - orderitem.Quantity;


/*trigger 2 to check Product Quantity available*/
DELIMITER $$
CREATE TRIGGER `Check_product_qty_availability`
before insert ON `orderitem`
FOR EACH ROW
BEGIN
```

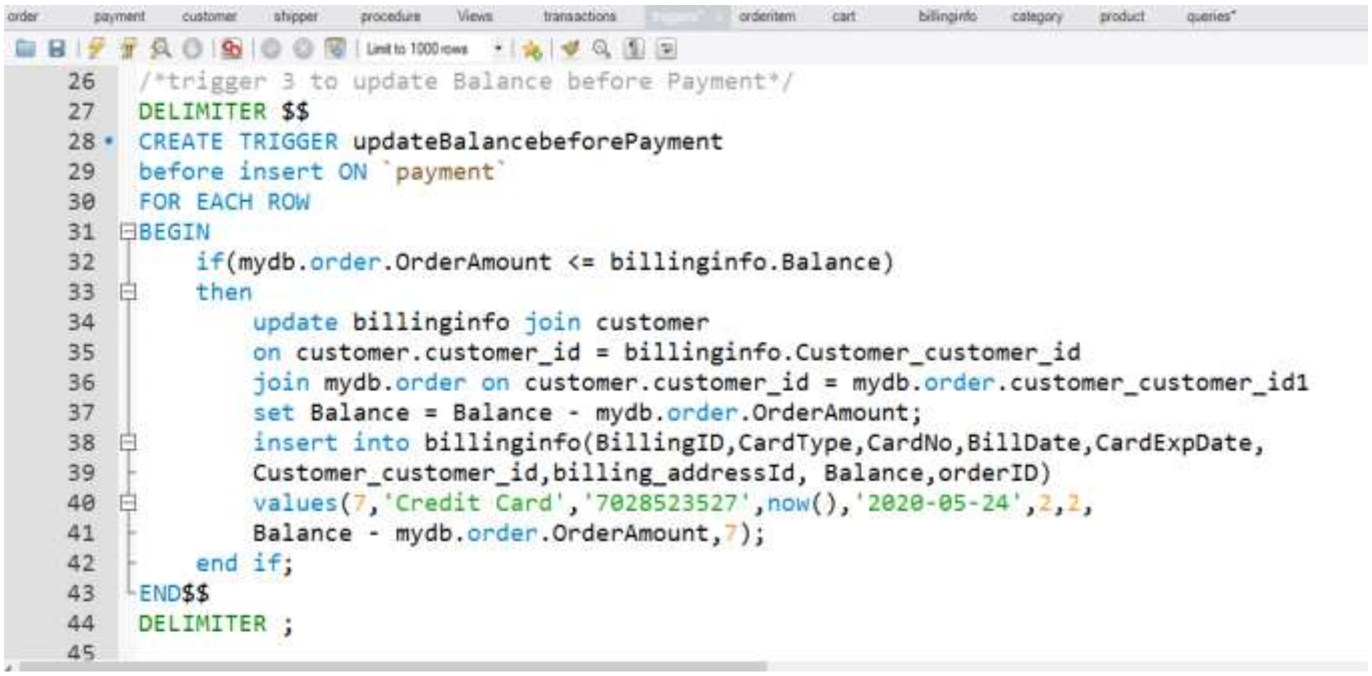
2. **Trigger to check product quantity availability**



```
/*trigger 2 to check Product Quantity available*/
DELIMITER $$
CREATE TRIGGER `Check_product_qty_availability`
before insert ON `orderitem`
FOR EACH ROW
BEGIN
   IF (orderitem.Quantity <= product.unitsInStock)
   THEN
        insert into orderitem(orderID,productID,unitPrice,Quantity,discount)
        values(new.orderID,new.productID,new.unitPrice,new.Quantity,new.discount);
   END IF;
END$$
DELIMITER ;

/*trigger 3 to update Balance before Payment*/
```

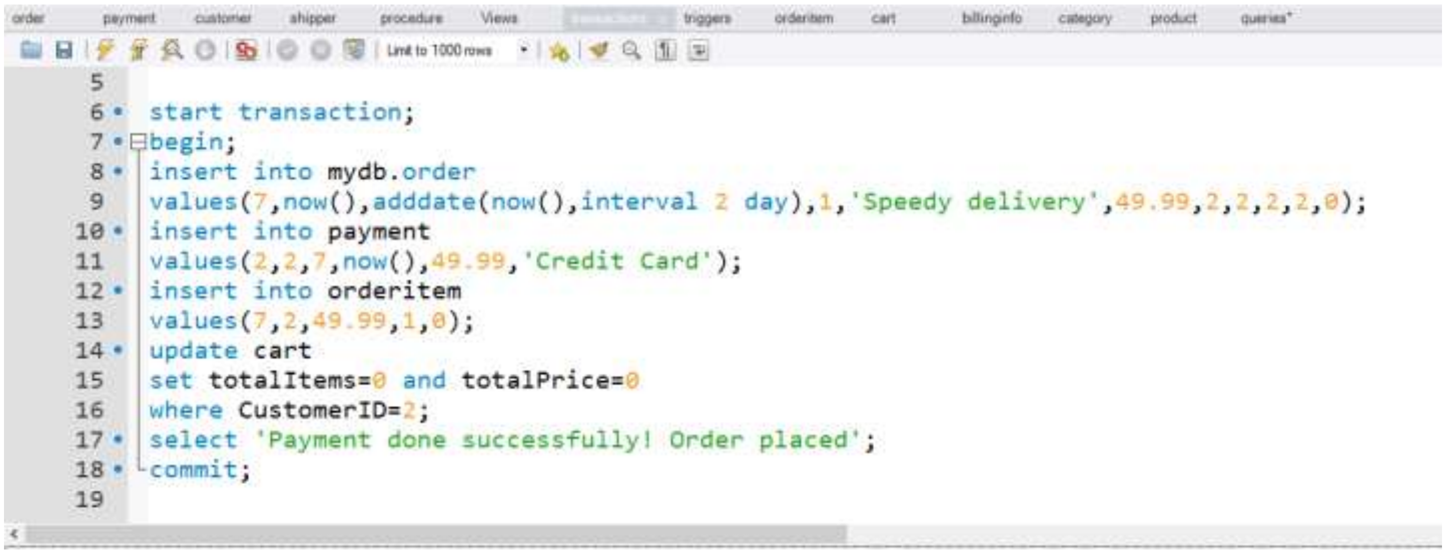
3. Trigger to deduct amount from account of customer before payment



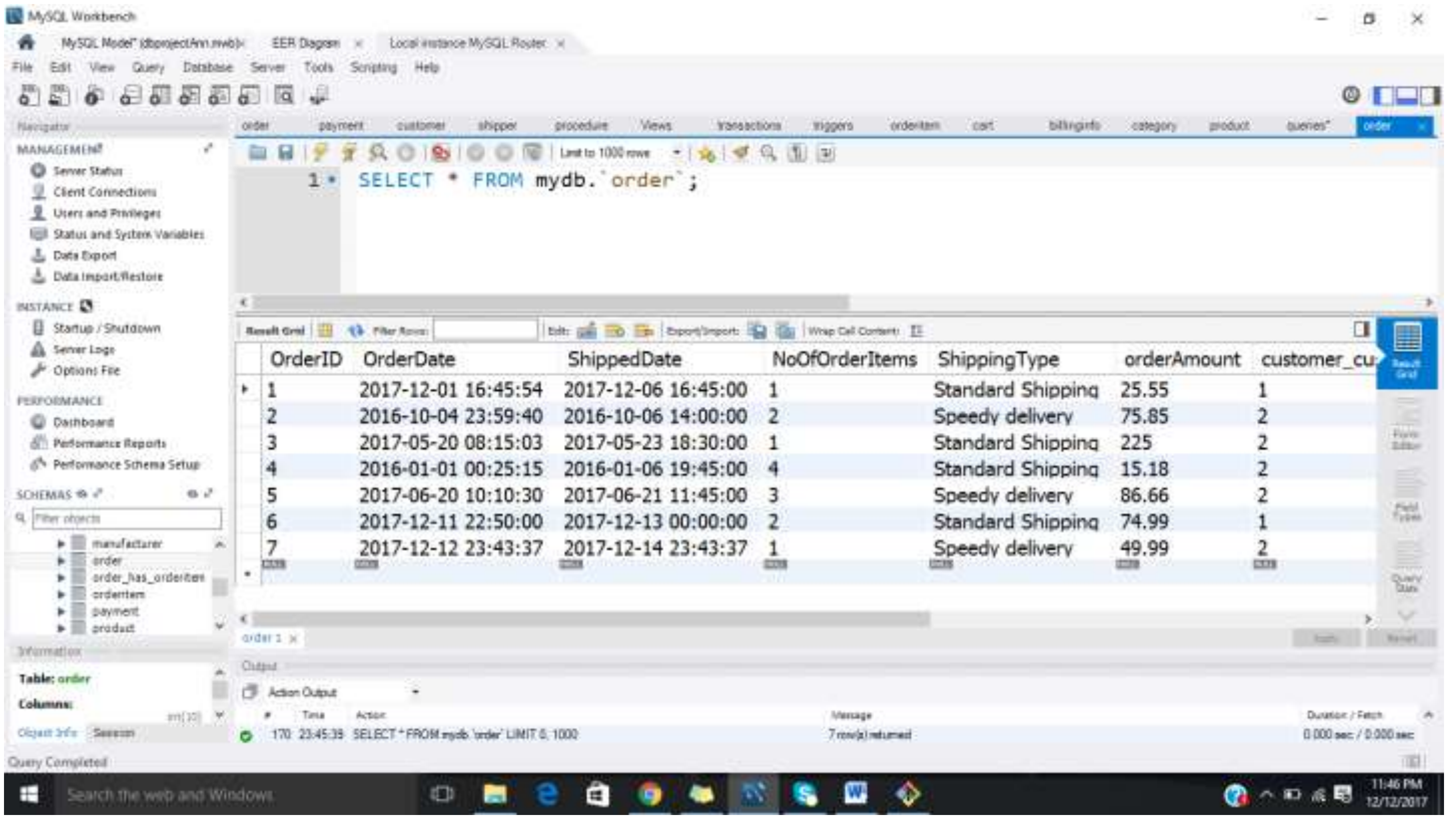
```
/*trigger 3 to update Balance before Payment*/
DELIMITER $$
CREATE TRIGGER updateBalancebeforePayment
before insert ON `payment`
FOR EACH ROW
BEGIN
    if(mydb.order.OrderAmount <= billinginfo.Balance)
    then
        update billinginfo join customer
        on customer.customer_id = billinginfo.Customer_customer_id
        join mydb.order on customer.customer_id = mydb.order.customer_customer_id1
        set Balance = Balance - mydb.order.OrderAmount;
        insert into billinginfo(BillingID,CardType,CardNo,BillDate,CardExpDate,
        Customer_customer_id,billing_addressId, Balance,orderID)
        values(7,'Credit Card','7028523527',now(),'2020-05-24',2,2,
        Balance - mydb.order.OrderAmount,7);
    end if;
END$$
DELIMITER ;
```

TRANSACTIONS

When the transaction is executed, the insert on order and payment is done if the transaction is committed.
A trigger is used to check availability before insert into order and a trigger is used to update product's unitInStock and unitsOnOrder.
A trigger is also used to update balance of the card of the customer.



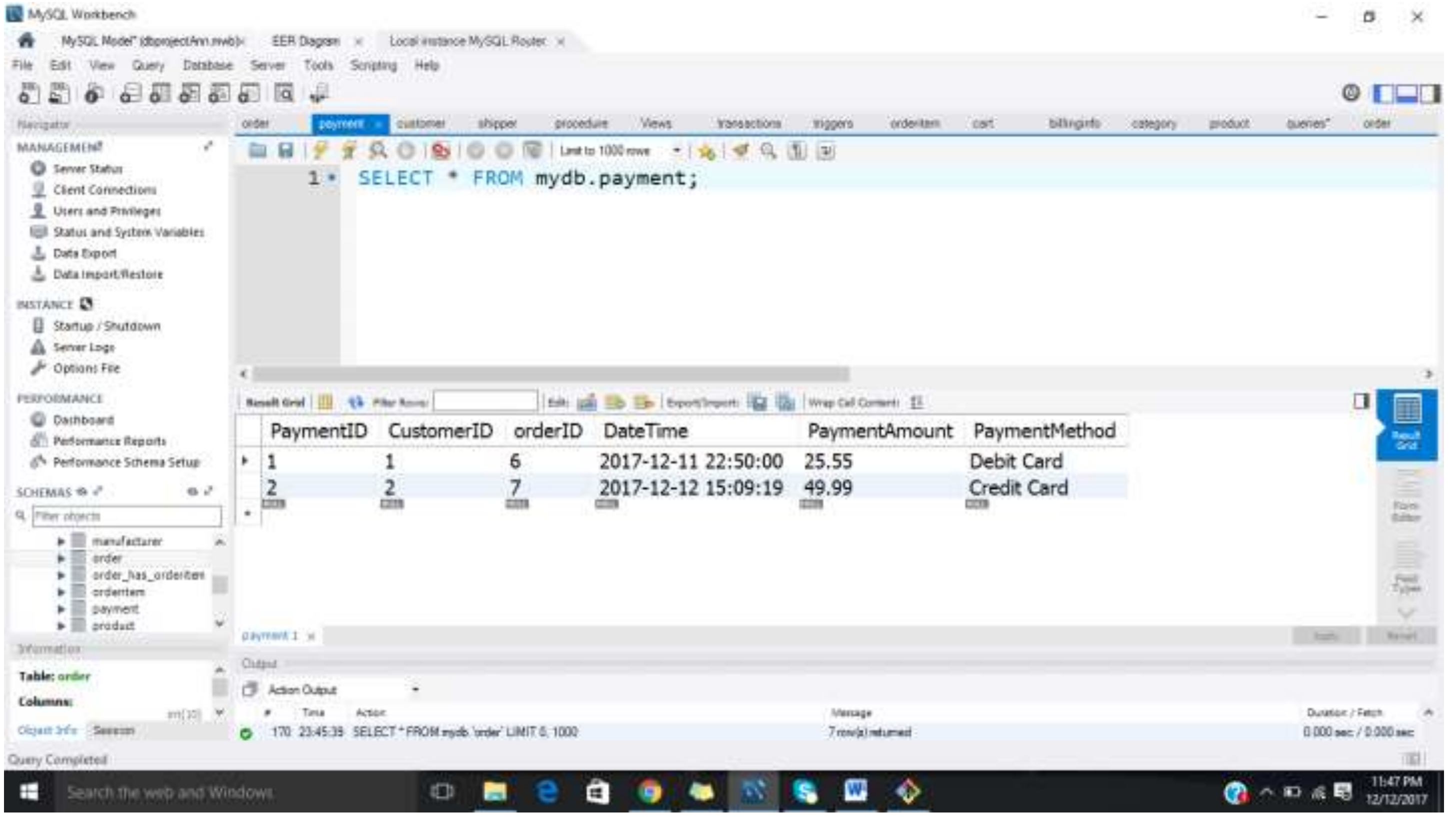
```
start transaction;
begin;
insert into mydb.order
values(7,now(),adddate(now(),interval 2 day),1,'Speedy delivery',49.99,2,2,2,2,0);
insert into payment
values(2,2,7,now(),49.99,'Credit Card');
insert into orderitem
values(7,2,49.99,1,0);
update cart
set totalItems=0 and totalPrice=0
where CustomerID=2;
select 'Payment done successfully! Order placed';
commit;
```



```
SELECT * FROM mydb.`order`;
```

OrderID	OrderDate	ShippedDate	NoOfOrderItems	ShippingType	orderAmount	customer_cu
1	2017-12-01 16:45:54	2017-12-06 16:45:00	1	Standard Shipping	25.55	1
2	2016-10-04 23:59:40	2016-10-06 14:00:00	2	Speedy delivery	75.85	2
3	2017-05-20 08:15:03	2017-05-23 18:30:00	1	Standard Shipping	225	2
4	2016-01-01 00:25:15	2016-01-06 19:45:00	4	Standard Shipping	15.18	2
5	2017-06-20 10:10:30	2017-06-21 11:45:00	3	Speedy delivery	86.66	2
6	2017-12-11 22:50:00	2017-12-13 00:00:00	2	Standard Shipping	74.99	1
7	2017-12-12 23:43:37	2017-12-14 23:43:37	1	Speedy delivery	49.99	2

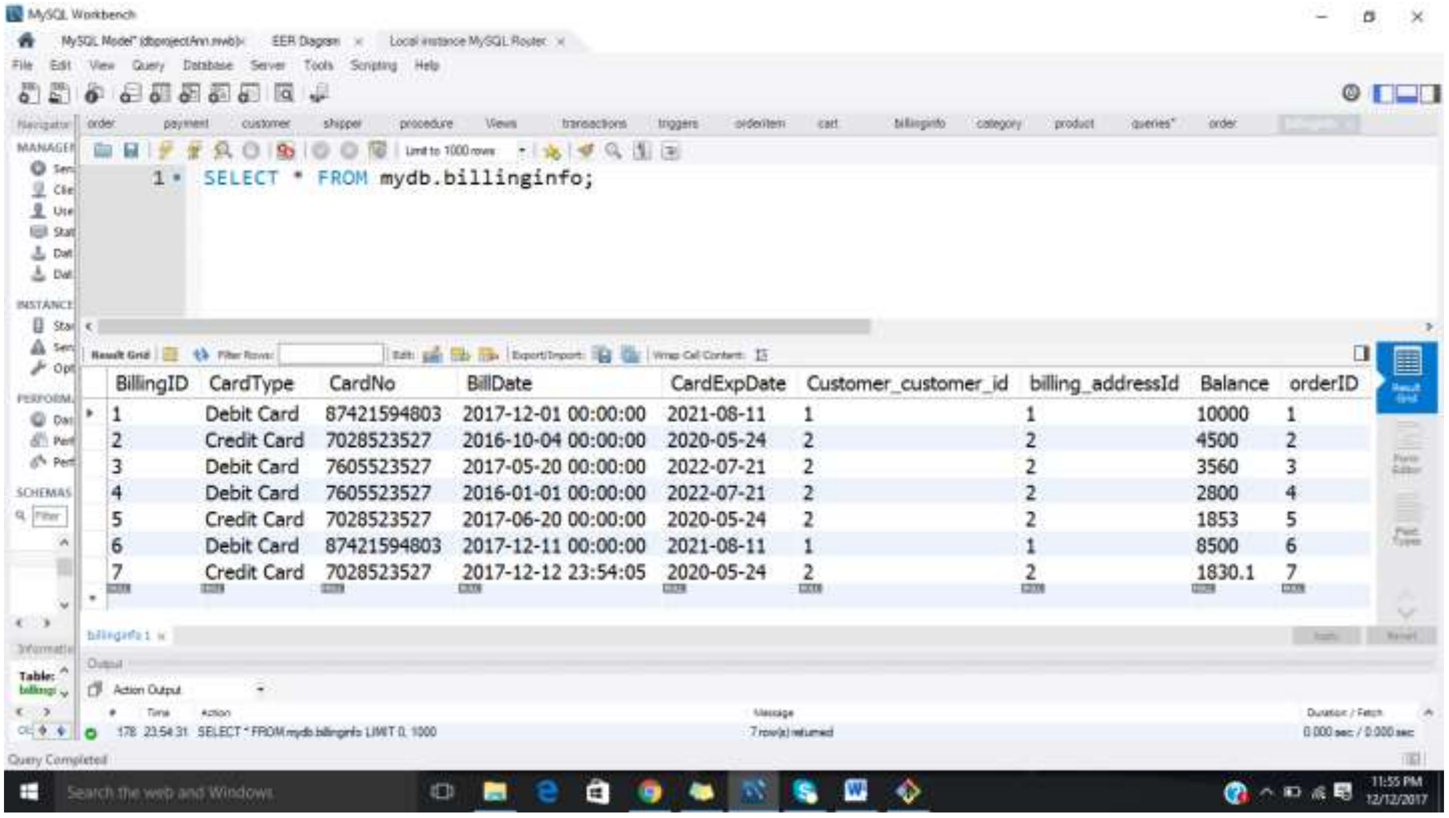
```
170  23:45:39  SELECT * FROM mydb.`order` LIMIT 0, 1000    7 row(s) returned    0.000 sec / 0.000 sec
```



```sql
SELECT * FROM mydb.payment;
```

PaymentID	CustomerID	orderID	DateTime	PaymentAmount	PaymentMethod
1	1	6	2017-12-11 22:50:00	25.55	Debit Card
2	2	7	2017-12-12 15:09:19	49.99	Credit Card

```
170  23:45:39  SELECT * FROM mydb.`order` LIMIT 0, 1000    7 row(s) returned    0.000 sec / 0.000 sec
```



```sql
SELECT * FROM mydb.billinginfo;
```

BillingID	CardType	CardNo	BillDate	CardExpDate	Customer_customer_id	billing_addressId	Balance	orderID
1	Debit Card	87421594803	2017-12-01 00:00:00	2021-08-11	1	1	10000	1
2	Credit Card	7028523527	2016-10-04 00:00:00	2020-05-24	2	2	4500	2
3	Debit Card	7605523527	2017-05-20 00:00:00	2022-07-21	2	2	3560	3
4	Debit Card	7605523527	2016-01-01 00:00:00	2022-07-21	2	2	2800	4
5	Credit Card	7028523527	2017-06-20 00:00:00	2020-05-24	2	2	1853	5
6	Debit Card	87421594803	2017-12-11 00:00:00	2021-08-11	1	1	8500	6
7	Credit Card	7028523527	2017-12-12 23:54:05	2020-05-24	2	2	1830.1	7

```
178  23:54:31  SELECT * FROM mydb.billinginfo LIMIT 0, 1000    7 row(s) returned    0.000 sec / 0.000 sec
```

Hence, the following has been used in order to perform different operations on the database:

1. **Analytical queries**
2. **Joins**
3. **Views**
4. **Stored procedures**
5. **Date functions**
6. **Triggers**
7. **Transaction**

Thank you.