DBMS Project

Online Retail Store Database Design

End Project Evaluation

By Group 46

2020023

2020060

2020117

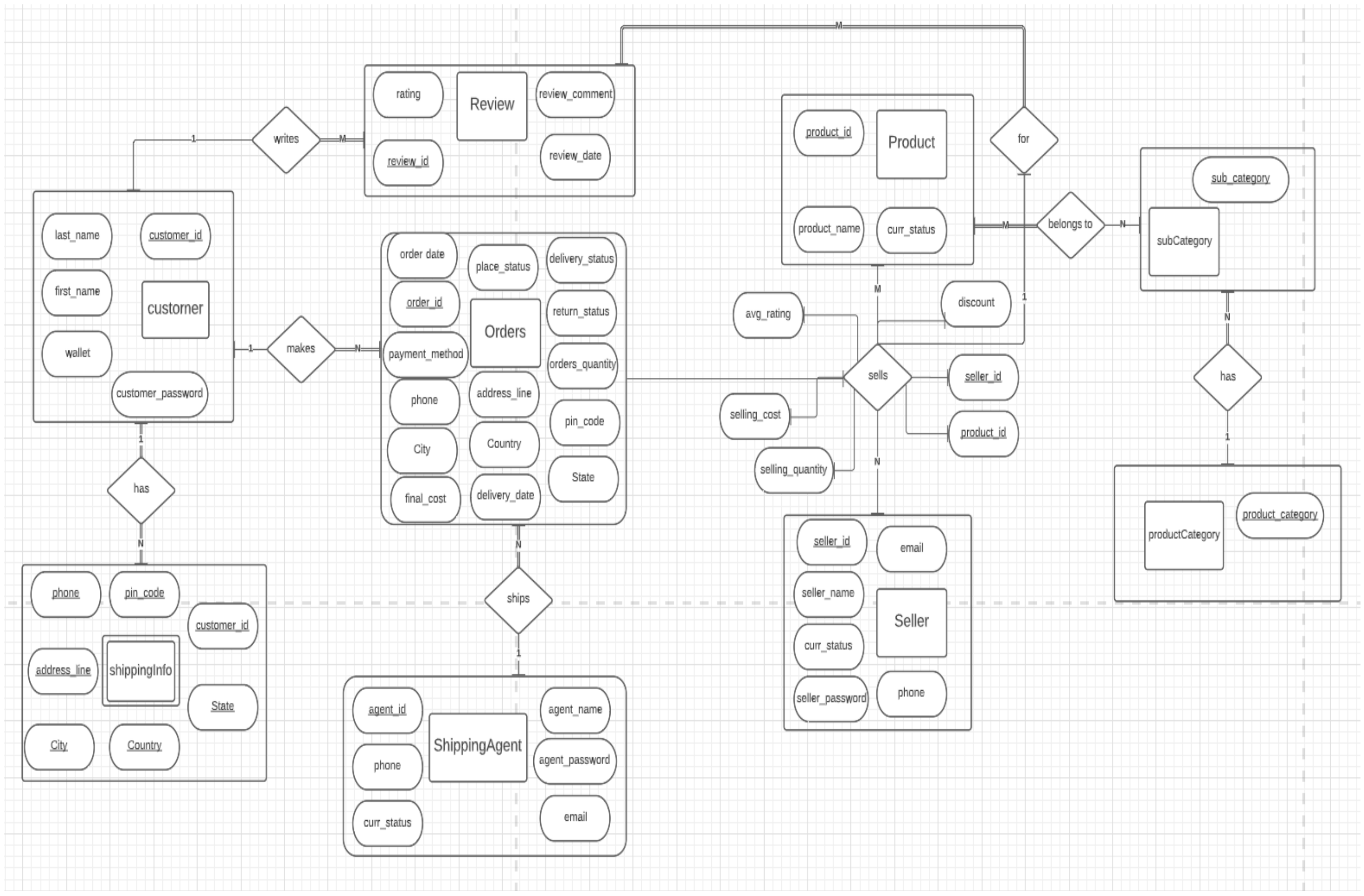
2020031

Description :

Developing an End 2 End DataBase Application, having primary focus
on the design of a back-end database of the applications requiring an extensive use of data entities selection and relationship between them, modeling of these data entities, relationships and constraints, populating the fictitious data in these data tables, database access and data manipulation.

Updated Scope and Schema :

As Discussed with the TA in the Mid Project Evaluation we were having issue with the RelationShip between the Orders , Product , Sellers Table ; So we have made a Ternary RelationShip between the Product , Seller and Order.



SQL QUERIES :

```
## 1. Rank the products according to their no. of entries in orders table
SELECT something.product_id, something.product_name, counts, DENSE_RANK() OVER
(ORDER BY counts DESC) AS ranks
FROM
        (SELECT product_temp.product_id, product_temp.product_name,
COUNT(orders_temp.orders_id) AS counts FROM
                (SELECT product_id, product_name FROM product) AS product_temp
                        INNER JOIN (SELECT product_id, orders_id FROM orders) AS
orders_temp
                                ON (product_temp.product_id = orders_temp.product_id)
        GROUP BY product_temp.product_name) AS something;
```

```
##  2. Selects the alternate shippingInfo of a customer
SELECT * FROM (
        SELECT s.*,
        row_number() over (partition by customer_id) as rn
        FROM shippingInfo s)  x
WHERE x.rn >1;
```

```
##  3. Selects the first two orders with different payment_method and delivery_status = “true”
SELECT * FROM (
        SELECT o.*,
        row_number() over (partition by payment_method order by orders_id) as rn
        FROM orders o)  x
WHERE x.rn < 3 and delivery_status = "true";
```

```
##  4.Query selects  products with the two least costs with curr_status in_stock,
no_longer_available, out_of_stock.
SELECT * FROM (
        SELECT p. product_id, p. product_name, p. curr_status, s. selling_cost,
        DENSE_rank() over (partition by curr_status order by selling_cost) as rnk
        FROM product p, sells s)  x
WHERE x.rnk < 3;
```

```
##  5. Selects first three sellers with different curr_status
SELECT * FROM (
          SELECT s.*,
          row_number() over (partition by curr_status) as rn
          FROM seller s)  x
WHERE x.rn < 4;

##  6. Selecting all active listings of products which are electronics and agave an average rating
more than 1.0
SELECT seller.seller_name, product.product_name,
productCategory.product_category,sells.avg_rating
FROM seller JOIN sells JOIN product JOIN belongTo JOIN subCategory JOIN productCategory
where productCategory.product_category = "Electronics"
and sells.avg_rating >= 1.0;

##  7. All orders where shipping agents that are no longer working have delivered
select * from orders join shippingAgent where orders.agent_id = shippingAgent.agent_id and
shippingAgent.curr_status = "NOT_WORKING";

##  8. All of the orders that are food and have been returned successfully
select * from orders where orders_id = (Select orders_id from orders join product join belongTo
join subCategory join productCategory where product.product_id = belongTo.product_id and
subCategory.sub_category = belongTo.sub_category and subCategory.product_category =
productCategory.product_category and
belongTo.product_id = orders.product_id and productCategory.product_category = "Food" and
orders.return_status = "ACCEPTED");

## 9. For customers with no orders yet
select * from customer where not customer_id in (select customer_id from orders where
place_status = 'true');

## 10. For products out of stock or no longer selling
select * from product where curr_status = "OUT_OF_STOCK" or curr_status =
"NO_LONGER_AVAILABLE";
```

Embedded SQL queries (Also present in the Code):

## 1. For Displaying seller's ID after login as seller

```
"Select seller_id, seller_name, curr_status, phone, email from seller where seller_id = " +
str(seller_id)
```

## 2. For deleting a seller's account

```
"UPDATE seller SET curr_status = '{}' WHERE seller_id = '{}'".format("NOT_WORKING",
seller_id)
```

##3. For a seller to view their active listings

```
cursor.execute("Select * from view_listings where seller_id = " + str(seller_id))
```

##4. For a seller to view ratings and reviews on their listing

```
"select first_name, last_name, rating, review_comment, review_date from
reviews_for_seller_product where seller_id = " +
                str(seller_id) +" and product_id = " + str(product_id)
```

Triggers Supporting Data Management in the Application

##1. a new order is placed after adding to the cart (equivalently, the false place_status status is updated to true)

```
delimiter //
CREATE TRIGGER place_order
BEFORE UPDATE ON orders
FOR EACH ROW
BEGIN
	IF old.place_status = "false" AND new.place_status = "true" THEN
		UPDATE sells SET sells.selling_quantity = sells.selling_quantity -
new.orders_quantity WHERE sells.product_id = new.product_id AND sells.seller_id =
new.seller_id;
	END IF;
END; //
delimiter ;
```

##2. (order is in transit OR is delivered) and customer chooses cancel/return respectively

```
delimiter //
CREATE TRIGGER cancel_return
BEFORE UPDATE ON orders
FOR EACH ROW
BEGIN
	##if cancels the order in transit (the cancel request is accepted immediately and the
order is sent back to the seller)
	IF OLD.place_status = "true" AND OLD.delivery_status = "false" AND
OLD.return_statusplace_ordercancel_returnplace_order = NULL AND NEW.return_status =
"ACCEPTED" THEN
		UPDATE sells SET selling_quantity = selling_quantity + NEW.orders_quantity;
	##if returns the order after delivery and the return request is accepted(i.e. delivery boy
confirmed that the product is in good shape)
  ELSEIF OLD.place_status = "true" AND OLD.delivery_status = "true" AND OLD.return_status
= "ONGOING" AND NEW.return_status = "ACCEPTED" THEN
```

```
                    UPDATE sells SET selling_quantity = selling_quantity + NEW.orders_quantity
WHERE sells.product_id = NEW.product_id AND sells.seller_id = NEW.seller_id ;
          END IF;
END; //
delimiter ;


##3.  avg rating of a sells entry
delimiter //
CREATE TRIGGER update_avg_rating
AFTER INSERT ON review
FOR EACH ROW
BEGIN
          #SET @avg_value = (SELECT ROUND (AVG(rating), 1) AS avg_rat FROM review WHERE
review.seller_id = NEW.seller_id AND review.product_id = NEW.product_id);
   UPDATE sells SET avg_rating = ((SELECT ROUND (AVG(rating), 1) AS avg_rat FROM review
WHERE review.seller_id = NEW.seller_id AND review.product_id = NEW.product_id)) WHERE
sells.product_id = NEW.product_id AND sells.seller_id = NEW.seller_id;
END; //
delimiter ;

##4. update product_curr status
delimiter //
CREATE TRIGGER update_product_status
AFTER UPDATE ON sells
FOR EACH ROW
BEGIN
          IF        (SELECT SUM(selling_quantity) as product_stock
                    FROM sells
                    WHERE sells.product_id = NEW.product_id) = 0 THEN
      UPDATE product SET curr_status = "OUT_OF_STOCK" WHERE product_id =
NEW.product_id;
          END IF;
END; //
delimiter ;
```

Index Tables

##1. Index of first names for customer table(index_name)

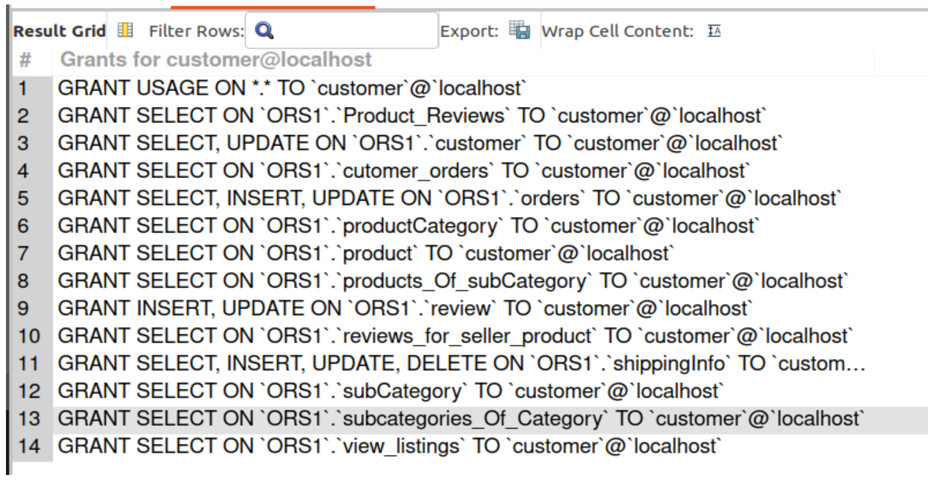
Create index index_name on customer (first_name);

##2. Index of agent names for shipping agent table(index_name)

create index index_name on shippingagent (agent_name);

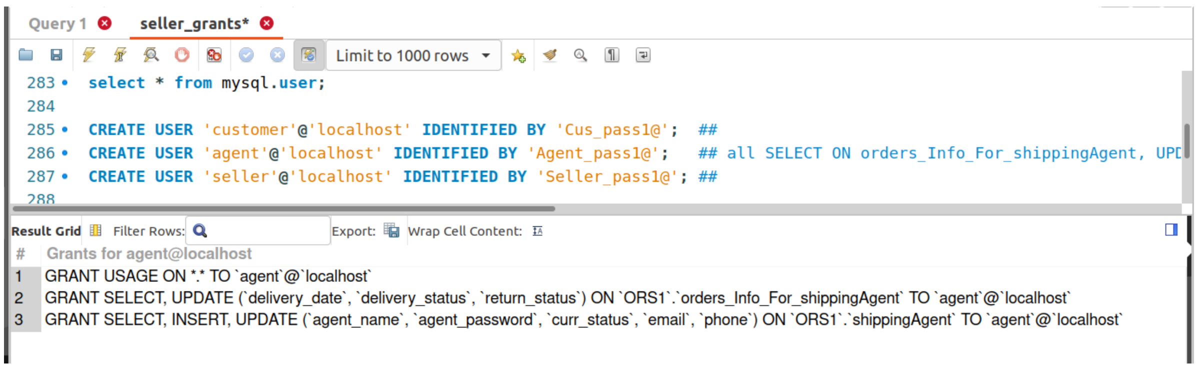
Views And Grants

Customer



#	Grants for customer@localhost
1	GRANT USAGE ON *.* TO \`customer\`@\`localhost\`
2	GRANT SELECT ON \`ORS1\`.\`Product_Reviews\` TO \`customer\`@\`localhost\`
3	GRANT SELECT, UPDATE ON \`ORS1\`.\`customer\` TO \`customer\`@\`localhost\`
4	GRANT SELECT ON \`ORS1\`.\`cutomer_orders\` TO \`customer\`@\`localhost\`
5	GRANT SELECT, INSERT, UPDATE ON \`ORS1\`.\`orders\` TO \`customer\`@\`localhost\`
6	GRANT SELECT ON \`ORS1\`.\`productCategory\` TO \`customer\`@\`localhost\`
7	GRANT SELECT ON \`ORS1\`.\`product\` TO \`customer\`@\`localhost\`
8	GRANT SELECT ON \`ORS1\`.\`products_Of_subCategory\` TO \`customer\`@\`localhost\`
9	GRANT INSERT, UPDATE ON \`ORS1\`.\`review\` TO \`customer\`@\`localhost\`
10	GRANT SELECT ON \`ORS1\`.\`reviews_for_seller_product\` TO \`customer\`@\`localhost\`
11	GRANT SELECT, INSERT, UPDATE, DELETE ON \`ORS1\`.\`shippingInfo\` TO \`custom...
12	GRANT SELECT ON \`ORS1\`.\`subCategory\` TO \`customer\`@\`localhost\`
13	GRANT SELECT ON \`ORS1\`.\`subcategories_Of_Category\` TO \`customer\`@\`localhost\`
14	GRANT SELECT ON \`ORS1\`.\`view_listings\` TO \`customer\`@\`localhost\`

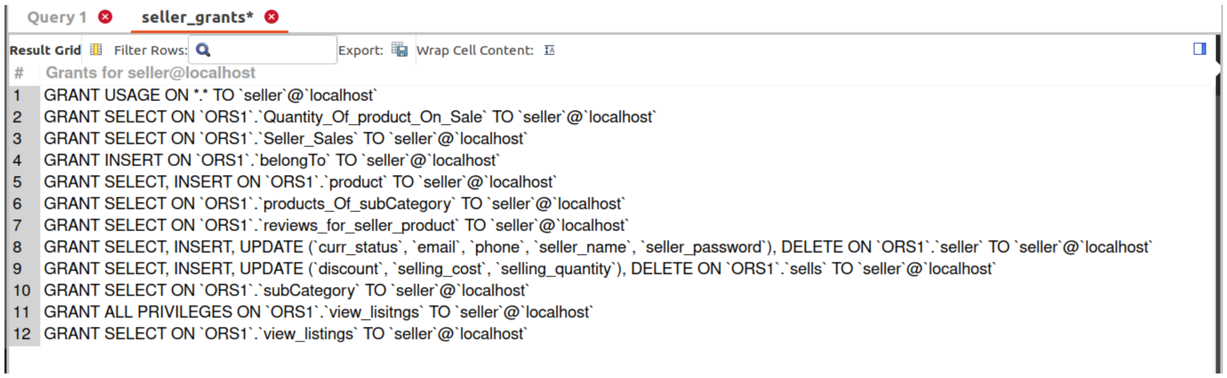
Agent



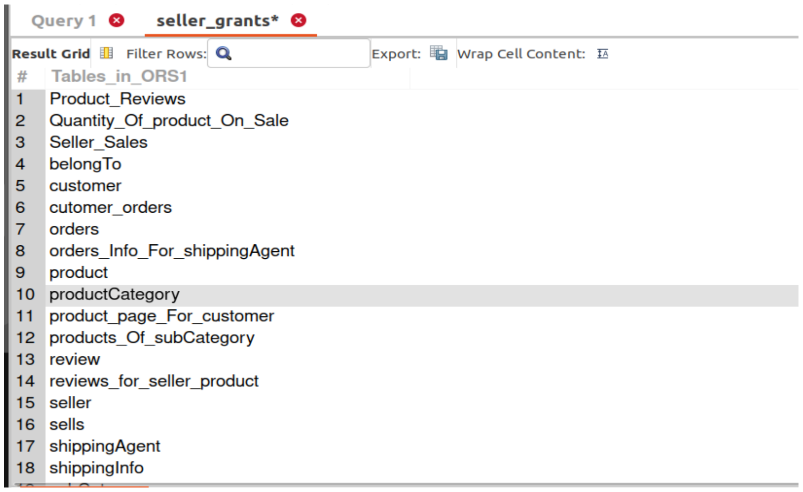
```sql
select * from mysql.user;

CREATE USER 'customer'@'localhost' IDENTIFIED BY 'Cus_pass1@';  ##
CREATE USER 'agent'@'localhost' IDENTIFIED BY 'Agent_pass1@';   ## all SELECT ON orders_Info_For_shippingAgent, UPD
CREATE USER 'seller'@'localhost' IDENTIFIED BY 'Seller_pass1@'; ##
```

```
#  Grants for agent@localhost
1  GRANT USAGE ON *.* TO `agent`@`localhost`
2  GRANT SELECT, UPDATE (`delivery_date`, `delivery_status`, `return_status`) ON `ORS1`.`orders_Info_For_shippingAgent` TO `agent`@`localhost`
3  GRANT SELECT, INSERT, UPDATE (`agent_name`, `agent_password`, `curr_status`, `email`, `phone`) ON `ORS1`.`shippingAgent` TO `agent`@`localhost`
```

Seller



```
#   Grants for seller@localhost
1   GRANT USAGE ON *.* TO `seller`@`localhost`
2   GRANT SELECT ON `ORS1`.`Quantity_Of_product_On_Sale` TO `seller`@`localhost`
3   GRANT SELECT ON `ORS1`.`Seller_Sales` TO `seller`@`localhost`
4   GRANT INSERT ON `ORS1`.`belongTo` TO `seller`@`localhost`
5   GRANT SELECT, INSERT ON `ORS1`.`product` TO `seller`@`localhost`
6   GRANT SELECT ON `ORS1`.`products_Of_subCategory` TO `seller`@`localhost`
7   GRANT SELECT ON `ORS1`.`reviews_for_seller_product` TO `seller`@`localhost`
8   GRANT SELECT, INSERT, UPDATE (`curr_status`, `email`, `phone`, `seller_name`, `seller_password`), DELETE ON `ORS1`.`seller` TO `seller`@`localhost`
9   GRANT SELECT, INSERT, UPDATE (`discount`, `selling_cost`, `selling_quantity`), DELETE ON `ORS1`.`sells` TO `seller`@`localhost`
10  GRANT SELECT ON `ORS1`.`subCategory` TO `seller`@`localhost`
11  GRANT ALL PRIVILEGES ON `ORS1`.`view_lisitngs` TO `seller`@`localhost`
12  GRANT SELECT ON `ORS1`.`view_listings` TO `seller`@`localhost`
```



```
#   Tables_in_ORS1
1   Product_Reviews
2   Quantity_Of_product_On_Sale
3   Seller_Sales
4   belongTo
5   customer
6   cutomer_orders
7   orders
8   orders_Info_For_shippingAgent
9   product
10  productCategory
11  product_page_For_customer
12  products_Of_subCategory
13  review
14  reviews_for_seller_product
15  seller
16  sells
17  shippingAgent
18  shippingInfo
```

Member contribution

SQL:

1. Views: Shahzan
2. Grants: Anas
3. Triggers:Shahzan
4. Index tables: Divyansh
5. Queries: Everyone submitted 2-3 queries

Python and embed queries :

Embedded queries: Divyansh and Shahzan

GUI: Shahzan and Anas