DBMS PROJECT

ER diagram:

https://lucid.app/lucidchart/0f591174-d6d1-42d6-aa45-0bb797e02e88/edit?beaconFlowId=F9ED06F2DF35DAEC&page=0_0&invitationId=inv_0bb08913-ee95-407f-bbd2-e5170b4043a7#

Relational Schemas:

1) **Customer** (customer_id, first_name, , last_name, emaill, c_password, phone, c_type)
PK: Customer_ID

2) **c_Order** (Order_ID, order_Status, Shipping_Price, Total_Price, Delivery_Date, Order_Date, Delivery_address_id, shipper_ID,Customer_ID)
PK: Order_ID
FK: shipper_ID,Customer_ID

3) **Orders**(Product_Id, Order_ID,quantity,price)
PK : Product_Id, Order_ID
FK : Product_Id, Order_ID

4) **Shipper**(Shipper_id,Shipper_Phone, Shipper_email, Shipper_details)
PK: Shipper_id

5)**Transanction**(Transanction_id,Customer_id,Transanction_Date,Type_Of_Transanction,Amount,Status_Type)
PK: Transanction_id
FK: CustomerID

6) **Shopping_cart**(Total_Price,Customer_id)
FK: Customer_id

7) **goes_to2**(Product_id,Customer__id, quantity)
PK: Product_id,Customer_id

8) **Product**(Product_ID, category_id, Price, Rating, Name, Description, Quantity, Discount_percentage, Seller_Id)
PK: Product_ID
FK: category_id, Seller_Id

9) **Category**(category_name, category_id)
PK: category_id

10) **Wishlist**(Name,date_of_creation,Customer_id)
FK : CustomerID

11) **goes_to1**(Wishlist_name,Customer_ID, Product_ID)
PK: wishlist_name, Customer_ID,Product_ID

12) **Address**(Address_id,CustomerID,Street1,Street2,City,State,Country,Pincode,Phone)
FK : CustomerID

13) **Seller**(Id, seller_name, Rating, Contact)
PK : Id

Queries:

1) List all the transactions done by all particular customers which are not passed.

```
SELECT t.Transanction_id,  c.customer_id, t.Transanction_Date, t.Type_Of_Transanction, t.amount
FROM transanction as t
JOIN customer as c
USING (customer_id)
WHERE t.status_type != "pass";
```

2) **Customer with given id wants to view the product details and wishlist details present in the wishlist**

```
SELECT p.product_id, p.description, p.rating, p.price, g.wishlist_name, w.date_of_creation
FROM goes_to1 g
JOIN product p
USING (product_id)
JOIN wishlist w
USING (wishlist_name,customer_id)
WHERE customer_id = 1;
```

3) A view to display details all the orders placed by customers along with the seller details

```
CREATE VIEW product_details AS
SELECT co.order_id, co.order_status, co.total_price, co.delivery_date, p.product_id, p.price,
p.discount_percent, s.seller_name, s.rating
FROM c_order co
JOIN orders o
USING (order_id)
JOIN product p
ON (p.product_id = o.product_id)
JOIN seller s
ON (p.seller_id = s.seller_id);
```

4) Find customers ids ,customers name and orders_id, order_status whose transactions were placed between 2021-05-01 and 2021-12-01

```
select c.customer_id,c.first_name,c.last_name, o.order_id, o.order_status from customer c,
c_order o  where  c.customer_id in
(select customer_id from transanction where transanction_date >'2021-05-01' and
transanction_date<'2021-12-01')
and o.customer_id = c.customer_id ;
```

5) Find the names of sellers along with the quantity and total amount of products sold by them.

```
select product.seller_id as seller_id, seller.seller_name as seller_name,
sum(orders.quantity) as quantity_sold, sum(orders.price * orders.quantity)
as total_amount from orders, product, seller where orders.product_id = product.product_id
and seller.seller_id = product.seller_id   group by product.seller_id;
```

6) Finding the total price in the cart for customer id =5

Select sum(goes_to2.Quantity * price * (100-Discount_percent)/100)
as total_price from goes_to2 **inner join** product on goes_to2.product_id =
product.product_id **where** goes_to2.customer_id = 5;

7) Accessing items from the cart for customer id = 1

Select product.product_id, Name, Description, goes_to2.Quantity,Rating,
(goes_to2.Quantity * price * (100-product.Discount_percent)/100)
as price from product, goes_to2 where customer_id = 1 **and** product.product_id =
goes_to2.product_id;

8)Finding category _id, category name which has a max number of product init.

Select category_name, category_id from category where category_id = (select category_id
from product group by category_id **order by** (quantity) **desc limit 1**);

9)Finding details of the seller having a maximum count of total products quantity

Select seller_id, seller_name, contact from seller where seller_id in (select seller_id from
product where quantity = (select quantity from product group by seller_id **order by**
quantity **desc LIMIT** 1) **group by** seller_id);

10)Finding the product which has the best rating in a category equals 1

select * from product **where** rating = (select **max**(rating) from product where category_id =
1) and category_id =1 ;

11) Finding names of sellers who sells more than 4 categories of products.

Select product.seller_id, seller.seller_name from seller, product
where product.seller_id = seller.seller_id **group by** product.seller_id having **count**(**distinct**
product.category_id) > 4;

Data generated using: https://www.mockaroo.com/

- Indexing
- Grant
- Queries
- Embedded query
- Trigger
- Views

Indexing

Code:

-> create index prod_name_and_categoryid_and_seller
on product (name, category_id,seller_id)

create index cid_and_c_password_idx
on customer(customer_id,c_password);

create index seller_id_and_seller_phone
on seller(seller_id, contact);

create index shipper_id_and_shipper_phone
on shipper (shipper_id,shipper_phone)

—---

Grants

Code:

-> GRANT select ON mydatabase.* TO 'shipper';
GRANT select ON mydatabase.* TO 'seller';

GRANT UPDATE, INSERT, DELETE ON mydatabase.address to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.c_order to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.customer to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.goes_to1 to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.goes_to2 to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.orders to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.product to 'customer';
GRANT UPDATE, INSERT, DELETE ON mydatabase.transanction to 'customer';

GRANT ALL ON mydatabase.orders to 'shipper';
GRANT ALL ON mydatabase.c_orders to 'shipper';
GRANT ALL ON mydatabase.product to 'shipper';
GRANT ALL ON mydatabase.customer to 'shipper';
GRANT ALL ON mydatabase.address to 'shipper';
GRANT ALL ON mydatabase.shipper to 'shipper';

GRANT ALL ON mydatabase.seller to 'seller';
GRANT ALL ON mydatabase.product to 'seller';
—----------

Aggregation Queries

-> **Average transaction amount for given past days**

```
String query = "select t.transanction_date,
avg(t.total_amount_for_day) over (order by t.transanction_date rows
"+past_days+" preceding) as average_amount from  " +
            "(select transanction_date , sum(amount) as
total_amount_for_day from transanction  where status_type = 'Pass'
group by transanction_date) as t;";
```

-> **Rank Customers based on total trancasction amount**

```
String query = "select customer_id, c.first_name, amount_sum,
dense_rank()  over(order by amount_sum desc) as rankk " +
            "from customer c join" +
            "(select  customer_id, sum(amount)  as amount_sum
from transanction group by customer_id) t " +
            "using(customer_id) limit " + limit + " ;";
```

-> **Rank bestselling products.**

```
select a.product_id , rank() over (order by a.sum desc ) as
bestsellerProduct  from
(select product_id,sum(quantity) as sum from orders group by
product_id ) as a order by bestsellerProduct asc;
```

-> **Finding Cumulative distribution of the orders table on the basis of total quantity sold for certain product.**

```
SELECT product_id, A.sum_quantity,
ROW_NUMBER() OVER (ORDER BY A.sum_quantity) row_num,
CUME_DIST() OVER (ORDER BY A.sum_quantity) cume_dist_val
FROM (select product_id, sum(quantity) as sum_quantity from
orders group by product_id) as A ;
```

Queries

-> A seller is updating some attribute of its product

```
String query = "update product set ";
if (Description.isSelected())
   query += " description = " + "\"" + newValueLabel.getText() + "\"";
else if (prodName.isSelected())
   query += " name = " + "\"" + newValueLabel.getText() + "\"";
else if (price.isSelected())
   query += " price = " + newValueLabel.getText();
else if (quantity.isSelected())
   query += " quantity = " + newValueLabel.getText();
else if (categoryID.isSelected())
   query += " category_id = " + newValueLabel.getText();
else if (sellerID.isSelected())
   query += " seller_id = " + newValueLabel.getText();
else if (Rating.isSelected())
   query += " rating = " + newValueLabel.getText();
else if (discount.isSelected())
   query += " discount_percent = " + newValueLabel.getText();
else {
   ok = false;
}

query += " where product_id = " + productID;
```

-> A new product is being added by a particular seller

```
String query = "insert into product values("+ id+", "+pprice+", "+prating+",
"+"\""+pname+"\""+", "+"\""+pdescription+"\""+", "+pquantity+", "+pdiscount+",
"+cid+", "+sellerid+");";
```

-> **Display wishlist details and product details of a particular product**

```
String query = "
    SELECT p.name, p.description, p.price
        FROM goes_to1 g
        JOIN product p
        USING (product_id)
        JOIN wishlist w
        USING (wishlist_name,customer_id)
        WHERE customer_id = 1";
```

-> **Get Details for products based on seller and category :**

Select product.*, category.category_name, seller.seller_name from product, category, seller where product.seller_id = seller.seller_id and product.category_id = category.category_id;

-> **Deleting cart details after an order has been placed by a user**

String query = "delete from goes_to2 where customer_id = " + cust_id + " and product_id = " + prod_id;

-> **Calculating total price after applying a discount of a cart of a particular customer**

String query = "Select sum(quantity * price * (100-discount)/100)" +

" as total_price from shoppingCartView where customer_id = " + customer_id+";";

-> **Getting product details, cart details, and the final price of a particular customer which is present in the cart**

String query1 = "Select g.product_id, g.customer_id, g.quantity,p.price as price_1u from goes_to2 g join product p using(product_id) where customer_id = "+customer_id +" ;";

-> **Getting order details of particular customer**
select order_id, total_price, order_date, order_status from c_order where customer_id = 1;
-> **Display products ordered under particular order id**
select name, orders.quantity, orders.price from orders, product where orders.product_id = product.product_id and orders.order_id = 1;

-> **Find customers ids ,customers name, and orders_id, order_status whose transactions were placed between 2021-05-01 and 2021-12-01**

select c.customer_id,c.first_name,c.last_name, o.order_id, o.order_status from customer c, c_order o where c.customer_id in (select customer_id from transanction where **transanction_date >'2021-05-01' and transanction_date<'2021-12-01**') and **o.customer_id = c.customer_id** ;
—----------

Trigger

-> This trigger reduces the quantity of certain products whenever an order is placed

```
delimiter ;
drop Trigger if exists productQuantity;
 delimiter |
Create Trigger productQuantity after insert on mydatabase.product
for each row
        begin
                update product set product.quantity  = product.quantity -
        new.quantity where new.Product_id = product_id;
        end;
|
delimiter ;
```

-> This trigger updates the c_order table whenever an order has been placed

```
use mydatabase;
drop Trigger if exists bookOrder;
delimiter |
Create Trigger bookOrder before insert on mydatabase.c_order
for each row
begin
```

```
set new.total_price = (select total_price from shopping_cart where
shopping_cart.Customer_id  = Customer_id);
end;
|
delimiter ;--
```

—----------

Views

Views :

-> (View to display order details to the seller)

```
create view displayOrderDetails as select orders.order_id, name, orders.quantity,
orders.price from orders, product where orders.product_id = product.product_id ;
select name, quantity, price from displayOrderDetails where order_id = 1;
```

-> (View to show shopping cart details to customer)

```
Select  goes_to2.customer_id, goes_to2.product_id, product.Name,
goes_to2.Quantity,product.price, product.discount_percent as discount from
product, goes_to2
```

—----------