Project Report

Online Retail Store System

Group 11
Anshak Goel - 2020283
Deeptorshi Mondal - 2020294
Pritish Poswal-2020321
Vibhor Agarwal-2020349

INTRODUCTION TO THE FINAL PROJECT

In the final project we have incorporated the database in the form of an application where we have tried to resemble it into an E-commerce website which is in a running form where users,sellers, and Administrators can work as they want in a real-life Platform. We have made the Front end to enable the User Interface and User Experience to the best of our abilities. Here our users can actually interact with our website and do changes as they want to according to their scope in the project. It ranges from selling and buying for users to make changes for Administrators and for Sellers for selling their products. It also enables the site to work with all the necessary constraints to put up an actual E-commerce site. We have connected the Database to our actual website which maintains the recording accordingly. if we do any basic change some change happens in or database.

We have used Flask to handle the backend which ranges from passing the data from the different stakeholders and putting it in the database as well as handling the data while traveling through different websites. Front End is developed through HTML and CSS. We have populated enough data to check how our website works.

CHANGES MADE AS SUGGESTED BY THE TEACHER ASSISTANT IN BRIEF

We made our project on E-Commerce and after completing it submitted the same as Mid semester evaluation. After the Evaluation we got some feedback form our teacher assistant where he suggested quite a few changes into our projects. We have made the changes as suggested by our teacher to make the project more fruitful.The following were the changes suggested:-

1. To include an offers section such that we can give different offers according to the amount value which is an attribute in the cart entity.
2. To include another stakeholder where in a delivery boy is also taken care of. It was suggested to take it as an entity and add certain attributes. For every order a delivery boy would be assigned.
3. We also made a ternary relationship between delivery boy, order and customer as suggested by the teacher assistant. This is the ternary relationship where the rating can be given by the customer to the prodcut as well as the delivery boy.

IMPORTANT NOTE- ALL THE CHANGES IN THE RELATONSHIP SCHEMA AND VARIOUS OTHER HEADING WOULD BE HIGHLIGHTED IN YELLOW PLEASE DO CHECK.

SCOPE OF PROJECT

Our Project deals with the making and managing of an online retail store. In this project our main objective is to make such a system which would help in the functioning of the online retail system amongst the stakeholders involved in such a system. Our scope of the project is mainly aiming to bridge the gap between people involved such as a seller,customer , the person administering it and the delivery boys. We would have different people to play different roles.

Here the Admin,Customer and the Seller all would be able to login to a page.

Specifically here we first take the Admin who would handle many requests and would ensure the proper working amongst people other than the organization being involved in the system. He would have all his details where he would have a name,ID,and a password. He would be able to add sellers giving products in the market and adding those products in the cart with their attributes. He would also be able to view the product. The Admin can also add a delivery boy to add products and also has the access to add offers for different orders as per their eligibility.

Then similarly a seller having their name, place of operation,password,email and phone no would be able easily be able to sell the products.

The product sold would also have various attributes where the admin and customer would be able to see it's price,name, brand, measurement,unitand would also have an ID.

Next we would have a customer who would have a name, password, email, mobile number and also an ID who would have the liberty to give the feedback on the product and is associated with the category and select the category. The customer now has the power to give a rating to the delivery boy correspondimng to a order..

The Feedback would have a body to write the details as well as contain an ID to store it. It would contain the date to be added and also the score to measure it.

We would have Categories containing the ID and the name to differentiate and the products would also be added to a cart having the total cost and the value added with it's ID.

Finally the cart makes the orders so that it can finally go for purchasing where the order would have it's ID, the address it is being delivered to, the mode of payment and the

amount to be given. It would have the order date and also it's time. The cart would also have a final value as an attribute which is there after a offer is applied to it. Now offers can be applied in the cart as well.The orders section would laos be assigned to a particular delivery boy.
We would have a offers section as well which is mainly used to apply special discounts to all the products that have been added in the product. The offer would contain a promo code, an offer id,a maximum discount, minimum value and also a percentage discount.

Next at last we have a deliver boy who carries out different orders throughout the process. The delivery would have his password email id, phone number, as well as his average rating. He would have an ID and also a name with first and last name. He would be assigned to a order.

So all in all a person would be able sell his products to the customer where the process would be managed by the admin and the order would be carried out by the delivery boy. It would have all the added features to make the process more efficient and smooth. We have also included Git for version control to ensure all members of the team have the latest version of the database with them. We regularly update dumps of database on Github.

Our final scope lies in identifying the products,users,admin ,orders ,sellers and delivery boy and maintaining a proper cycle and interaction between all.

STAKEHOLDERS OF THIS PROJECT

There are multiple Stakeholders in this project mainly the project is centered around customers and sellers along with it we also have admin and delivery boy as two more of our stakeholders. Customers are important stakeholders who perform operations like viewing products category wise, adding the products to cart, placing orders and giving feedback on products. They can also add rating to the delivery boy corresponding to a particular order.
Our other stakeholder seller sells various products which customers buy.

One other stakeholder here is the admin who maintains and adds data to our database for online retail store like adding products, adding sellers along with that he can view the orders customers have placed for processing them. The Admin also adds the delivery boy while the delivery boy always gets assigned to a order when it is made.

ASSUMPTIONS IN THE PROJECT

We have taken the following assumptions in our project.

1. First of all we have assumed that all orders will go through the cart and every order will have a unique cart with unique cart ID and would be associated with only one single customer. Also every customer will have one cart at a time to place order.
2. Only a customer can add product feedback to a product by giving review and rating to the products.
3. Each product will belong to some unique category and there are no categories which don't have any product.
4. Customer can view product directly also and can view it by selecting that particular category also.
5. There can be multiple admins and each of them has the power to add products,category and sellers.
6. Only admins can view details of all the orders placed by different customers while a customer can view only his order
7. Each product can be sold by any of the multiple sellers and each seller can sell multiple products
8. Seller can only know no of products of each type they have sold
9. We have also assumed each cart is associated with single order only and every order is associated with single cart only
10. ~~It is assumed in this project that all the products are sold at MRP and there is no option of discount or coupon code or cashback.~~ ASSUMPTION REMOVED.
11. We have made sure that offers are made according to a minimum price.It would have a maximum discount and would have a percentage discount.
12. A cart can have a single offer only. But a single offer can be made to multiple carts.
13. Only a single admin can add multiple offers.
14. Only a single admin can add multiple delivery boys.
15. A single delivery boy can be assigned to multiple orders
16. Multiple customers can give a particular rating value to multiple deliver boys corresponding to multiple orders..
17. A customer can add a rating corresponding to its own order only.

GRANTS IN THE PROJECT

1 GRANT FOR ADMIN

```
grant select,insert,update,alter
on category
to 'Admin1';

grant select,insert,update,alter
on delivery_boy
to 'Admin1';

grant select,insert,update,alter
on offer
to 'Admin1';

grant select,insert,update,alter
on product
to 'Admin1';

grant select,insert,update,alter
on seller
to 'Admin1';

grant select
on orders
to 'Admin1';

grant select
on customer
to 'Admin1';
```

EXPLAINATION-

This is the grant that has been created for the Admin. The Admin username is “Admin1” as shown in the above SQL query.
As seen above in this grant we have given the administrator who is “Admin1” the privilege to select,update,insert and alter the category entity as per his choice.
Now similarly the admin has the privelege to select,update,insert and alter the delivery_boy,offer,product and seller as per his choice.
On orders the admin is has the privilege to only select the orders as given by the customer.
The Admin has the privilege to only select a customer as well.

2 GRANT FOR SELLER

```
grant select,insert,update
on product
to 'Seller1','Seller2';

grant select
on orders
to 'Seller1','Seller2';

grant select
on offer
to 'Seller1','Seller2';

grant select
on sells
to 'Seller1','Seller2';

grant select
on category
to 'Seller1','Seller2';
```

EXPLAINATION-

This is the grant that has been created for two sellers.There are two usernames for two sellers which are “Seller1” and “Seller2” respectively.
As seen above in this grant we have given both the sellers who are “Seller1” and “Seller2” the privilege to select,update,insert and alter the product entity as per his choice.
On the contrary the “Seller1” and “Seller2 have the privilege to only select the orders,offer and category entity as given in the database.
Both the seller can also select the relationship table sells present in our database.

2 GRANT FOR DELIVERY BOY

```
grant select
on product
to 'Delivery_boy1','Delivery_boy2';

grant select
on orders
to 'Delivery_boy1','Delivery_boy2';
```

```
grant select
on offer
to 'Delivery_boy1','Delivery_boy2';

grant select
on product_feedback
to 'Delivery_boy1','Delivery_boy2';

grant select
on rates_order_delivery
to 'Delivery_boy1','Delivery_boy2';
```

EXPLAINATION-

This is the grant that has been created for two delivery boys.There are two usernames for two delivery boys are “Delivery_boy1” and “Delivery_boy2” respectively.
As seen above in this grant we have given both the delivery boys who are “Delivery_boy1” and “Delivery_boy2” the privilege to ONLY select product feedback,offer,orders and product.
Both the delivery boys can also select the relationship table rates_order_delivery present in our database.

ENTITIES WITH THEIR PRIMARY KEYS

There are numerous entities in the project and following are there keys:-

Customer:- It is the main person involved in the buying of the products who is able to select the category and also gives the product feedback. He is also associated with the cart for making the orders. The Customer has a name, an email, a phone number and also has been given a password. The customer can give a rating value to a delivery boy corresponding to a particular order Foreign Key- NA **PRIMARY KEY-**Customer_ID

Product:- This the element that is added by the admin,sold by the seller and finally selected by the customer through the user. It would have the name, the ID and the price. It would also have the brand,unit and the measurement. It is associated with cart and has a category. Foreign Key- Category_ID, **PRIMARY KEY-**Product_ID

Admin:- It is the work of the administrator to add the products and the person selling them. The Admin can also view what the orders are. The Admin can add offers as well as delivery boys. The Admin has a name,ID and a password. Foreign Key- Category_ID, **PRIMARY KEY-**Admin_ID

Category:- It is basically the section where the product belongs to. It would have the category name as well as the category ID. The Admin can also view what the orders are. It would have the name as well as the ID. Foreign Key- NA, **PRIMARY KEY-**Category_ID

Seller:-The person involved in selling the products in the store. The seller is added by the admin. The seller would have his name and an ID. The seller's contact details have been stored and his place of operation as well. A password also has been provided. Foreign Key- Admin_ID, **PRIMARY KEY-**Seller_ID.

Product Feedback[WEAK ENTITY]:-It is the review and feedback given by the customer. Every product would have feedback.The Feedback would have an ID. It would also contain the date when it is added. Product would also contain the rating and the body to determine what our review basically is. Foreign Key-NA, **PRIMARY KEY-**Review_ID ∪ Product_ID

Orders:-This is one of the main entities where the main order takes place. Here the products which are confirmed are finally put in the system. Here the cart would finally make the orders. The order would contain the ID, the address from where it is coming. It would also have the time and date of its delivery. One would also have the option to choose the mode of payment and it's amount. The order would be assigned to a delivery boy while with regards to a particular order a delivery boy would be given a rating.Foreign Key- Cart_ID, **PRIMARY KEY-**Order_ID.
Cart:- This would all the orders that are being stored by the customer. The cart would make the orders. Cart would have the ID, the total no of products as well as the total units each product has. The cart would also have the final amount as an attribute after applying a particular offer. The offer would also be applied to the cart.Foreign Key-NA , **PRIMARY KEY-**Admin_ID

Offer:- These will be the offers which will be applied on the basis of certain condition. It can be on the basis of a minimum price applied to the order. These offers would give the users some amount thereby decreasing the cost of the cart. The offer will be applied to the cart only. The offer will also have a promo code. So the attributes would be promo

code,offer ID, maximum discount,percentage discount and a min order value.Foreign Key-Admin_ID , **PRIMARY KEY-**Offer_ID

Delivery_boy:- This is the person who would deliver the orders to the customer. He would added by the admin and would also be liable to a rating by the customer with regards to a particular order. He/she would be assigned to a order. The delivery boy would have a first name, last name. In the form of contact the delivery boy would have a email address and a mobile number. An ID would be assigned and a average rating and password would be given.Foreign Key-Admin_ID , **PRIMARY KEY-**Delivery_Boy_ID

RELATIONSHIPS ESTABLISHED,ENTITY PARTICIPATION AND RELATIONSHIP TYPES

We have built many relationships between entities here and a brief description of each relationship along with participation and type is given below:-

1. **customer selects category :-** Here the relationship between customer and category is M to N as each customer can select multiple categories and each category can be selected by multiple customers. Here both customer and category has partial participation as it is not necessary that every customer selects some category and also it is not necessary for every category to be selected by some customer so the participation is partial.
2. **customer gives product_feedback :-** Here the relationship between customer and product_feedback is 1 to N as we know that a customer can give product feedback for multiple products but we know that a single product feedback can be given by only one customer. Here customer has partial participation as it is not necessary for every customer to give product feedback but it is necessary for every product feedback to given by some customer so product feedback has complete participation. Here our product feedback is weak entity and the relationship type is between the weak and strong entity so it was necessary for weak entity to have complete participation.
3. **category has product :-** Here the relationship between category and product is 1 to N as every category has multiple products but every each product belongs to only one category. Here there is complete participation from both category and product as each product belong to some category and also here we have assumed that each category has atleast one product.
4. **product has product feedback :-** Here the relationship between product and product feedback is 1 to N as every product has multiple feedbacks but one product feedback belongs to only one product. Here there is partial

participation from product side as not every product has product feedback but there is complete participation from product feedback side as every product feedback is associated with some product.

5. **seller sells product :-** Here the relationship between seller and product is m to n as every product can be sold by multiple seller and also every seller can sell multiple products. Here there is complete participation from seller as well as product because every seller sells some product and every product is sold by some seller.

6. **admin adds seller :-** Here the relationship between admin and seller is 1 to N. As 1 admin can add multiple seller but 1 seller can be added by one admin only only. Here there is partial participation by admin because we have assumed not every admin adds seller also there is complete participation by seller as every seller is added by some admin.
7. **admin adds product :-** Here the relationship between admin and product is 1 to N. As one admin can add multiple products but one product is added by single admin only. There is complete participation from product because every product is added by some admin but there is partial participation from admin because not every admin adds some product.
8. **cart makes order :-** Here the relationship between cart and order is 1 to 1. As we have assumed that one order can be made by single cart only and one cart makes only single order. Here there is partial participation from the cart because not every cart makes a order as some customer may leave some items inc art and may not place a order but there is complete participation from order because every order has been made by some cart.
9. **admin admin_views orders :-** Here the relationship between admin and order is M to N. It is M to N as each order can be viewed by multiple admins also each admin can view multiple orders. Here there is partial participation from both admin and orders as not every admin views some order also not every order is viewed by some admin.

10. **customer is associated with product and cart :-** This is a ternery relationship in our online retail store.
    Talking about these one by one we have taken some assumptions here
    1. Between customer and cart the relation is one to one as we have assumed and implemented in such a way that one customer will have only cart and one cart will be related to only one customer and here there is

partial participation from customer as we have assumed not every customer has a cart and there is complete participation from cart as every cart is associated with some customer.

2. Between product and cart the relation is N to 1 as we have assumed one product can be added to only single cart at a time for one user and one cart can have multiple products. There is partial participation from product as not every product may be added to cart but there is complete participation from cart as there is 1 unique cart for one user so it will always have product and participate.
3. Between customer and product the relationship is 1 to N as we have assumed one person can add multiple products to cart but one product can be added to cart by only single person at a time. There is partial participation from both customer and product as not every customer adds product to cart and not every product is added to cart by an customer according to our implementation.

11. Offer is Applied to a Cart

Here the relationship between offer and cart is 1 to N. As we have assumed that one order can be applied to mutiple carts only and only 1 cart can have only 1 offer only. Here there is partial participation from the cart because not every cart would have a offer as some offer may not be applied to a cart and hence leave some offers and there is partial participation from offer because not all offers can be applied to carts as it is not necessary that offer can be made.

12. Admin adds offer :- Here the relationship between admin and offer is 1 to N. As one admin can add multiple offers but one offer is added by single admin only. There is complete participation from product because every offer is added by some admin but there is partial participation from admin because not every admin adds some offer.

13. Admin adds delivery boys :- Here the relationship between admin and delivery boy is 1 to N. As one admin can add multiple delivery boys but one delivery boy is added by single admin only. There is complete participation from deliver boy because every delivery boy is added by some admin but there is partial participation from admin because not every admin adds some delivery boy.

14. delivery boy is assigned to orders :- Here the relationship between delivery boy and order is 1 to N. As one delivery boy can be assigned to multiple orders but one orderis can be assigned to a single delivery boy only.

There is complete participation from order because every order is assigned to some order but there is partial participation from delivery boy because not every delivery boy is assigned to a order.

15. Customer rates_order_delivery with respect to the order and delivery boy:- This is another ternery relationship in our online retail store.

Talking about these one by one we have taken some assumptions here

1. Between customer and delivery boy the relation is N to 1 as we have assumed and implemented in such a way that M customers can give multiple ratings to 1 delivery boy with respect to multiple orders and here there is partial participation from customer as we have assumed not every customer will give a rating to delivery boy and there is partial participation from delivery boy as not every delivery boy will be rated by some customer.
2. ** Between delivery boy and order the relation is 1 to N as we have assumed that multiple customers can rate 1 delivery boy with respect to mutiple orders.There is partial participation from delivery boy as not every delivery boy can be assigned to an order and there is partial participation from order as it is not necessary that every order has to be assigned to a delivery boy.
3. Between customer and order the relationship is M to N as we have assumed multiple customers can rate multiple orders say N with respect to a particular delivery boy. There is partial participation from both customer and order as not every customer adds rating with respect to order and not every order is rated by an customer according to our implementation.

IDENTIFICATION OF WEAK ENTITY AND REASONS

Product Feedback-

Product Feedback is that entity which is basically used to give reviews about a product. Here it would have a Review ID,Review Body Rating and the date it has been added.

Both Customer ID and Review ID would together form the primary key also known as composite key. The Product ID here is the foreign key.

WHY IS IT A WEAK ENTITY

Weak entities are those entities which completely rely on some other entity. In this case product feedback depends completely on the product. Here the product feedback would not exist without a product or without having a product, the product feedback would have no meaning. We just cannot have product feedback just like that . All in all product feedback is not identified by its entities alone. This would also have an identifying relationship where product feedback is uniquely identified when paired with a=the product entity. PRODUCT FEEDBACK WILL NOT EXIST ON IT'S OWN AND ONLY EXISTS IN THE CONTEXT OF A PRODUCT. The product entity must also have a complete participation with the product entity where each feedback must correspond to a product.

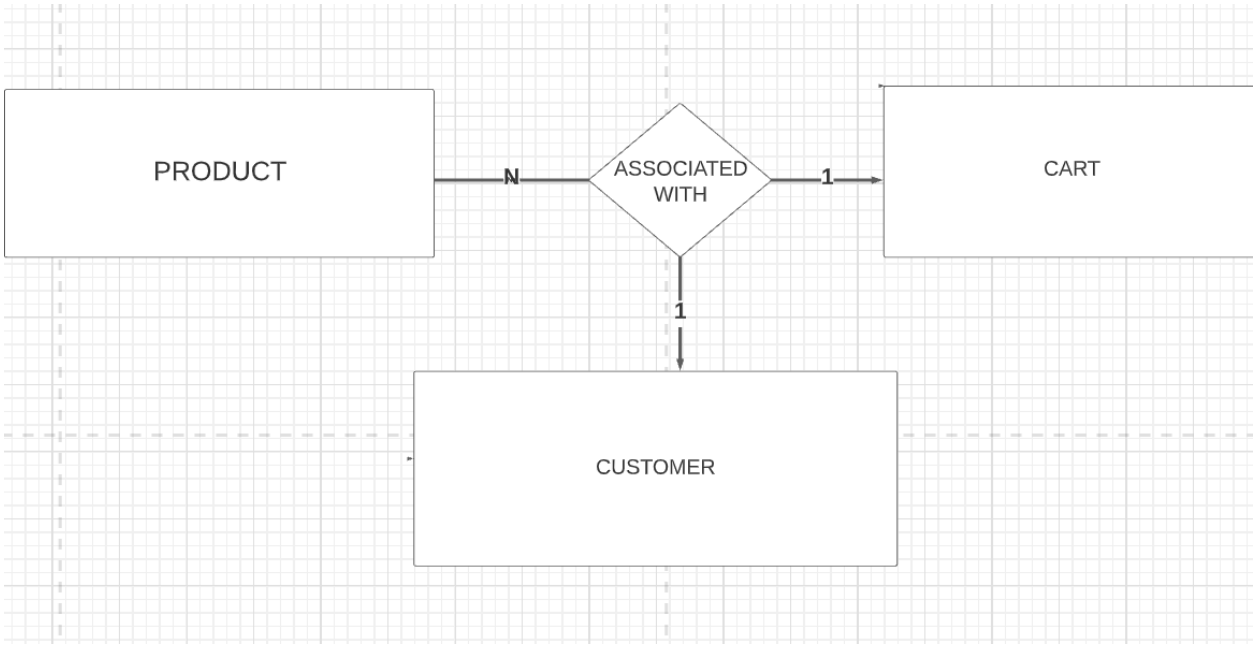
TERNARY RELATIONSHIP AND REASONS

1. **associated_with-** It is a ternary relationship between three different entities. Relationship denotes how entities are related to each other. They denote how tables are connected with each other in a manner so that different attributes are used together. Here 3 different entities take part in the relationship which are the cart, the customer and the product hence the degree is also 3. Let us analyze the different entities involved which are:-

1) Product- It is the main entity around which our relationship revolves where the product is put into the cart.
2) Cart- The cart as an entity is used by the the customer to put the products
3) Customer- It is one of the main entities which adds the product to the cart.

Product, Cart and Customer will participate simultaneously in a relationship. Because of this fact, when we consider cardinality we need to consider it in the context of two entities simultaneously relative to the third entity.

THE CARDINALITY



We will analyze the cardinalities of all the possible entities involved:-

1. **Relationship of Customer with Cart and Product**- Here the cardinality of the customer is 1 with respect to the Cart and Product. It is so because **at one time 1 Customer can add multiple say N products at 1 cart.** Hence the Cardinality is 1.

2. **Relationship of Cart with Customer and Product**- Here the cardinality of the cart is 1 with respect to the Customer and Product. It is so because **at one time 1 cart can be used by 1 customer to add multiple say N products.** Hence the Cardinality is 1.

3. **Relationship of Product with Customer and Cart**- Here the cardinality of the product is N with respect to the Customer and Cart. It is so because **at one time multiple products say(N) can be added into only 1 cart by only 1 customer.** Hence the Cardinality is N.

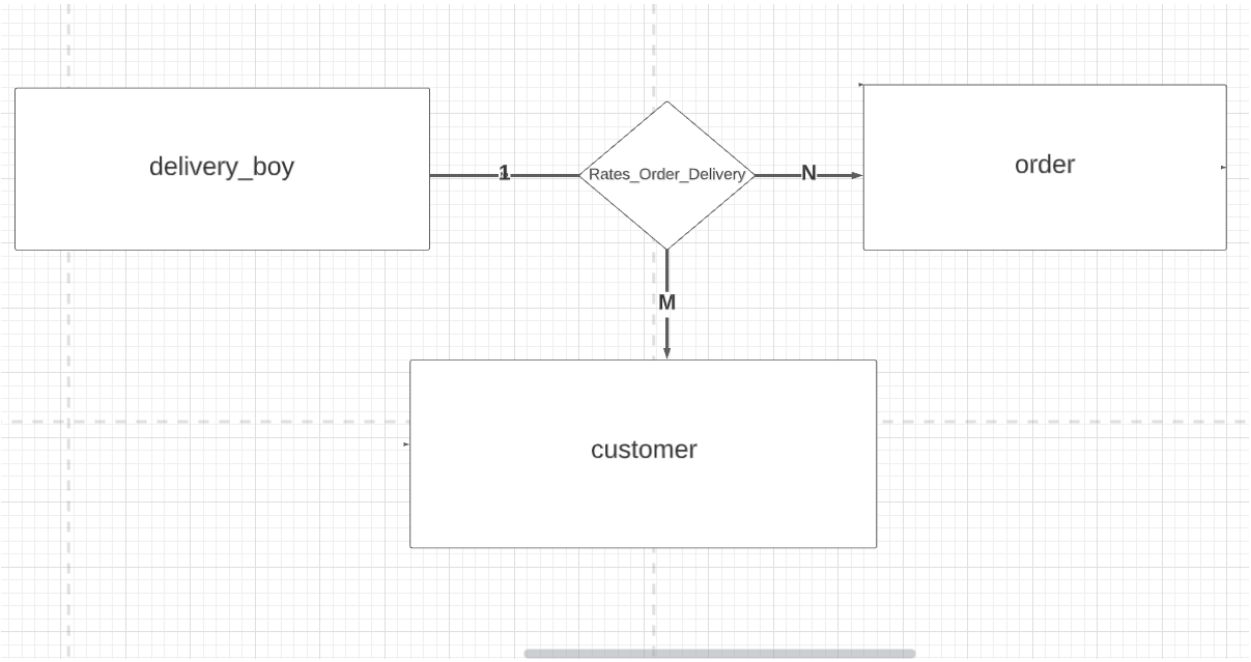
2. **Rates_order_delivery-** It is a ternary relationship between three different entities. Relationship denotes how entities are related to each other. They denote how tables are connected with each other in a manner so that different attributes are used together. Here 3 different entities take part in the relationship which are the order, the customer and the delivery boy hence the degree is also 3. Let us analyze the different entities involved which are:-

4) Customer- It is the main entity around which our relationship revolves where the customer rates the delivery boy.
5) Order- The order is an entity with respect to which the customer would give a rating to a delivery boy.

6) Delivery_Boy- It is one of the main entities where the delivery boy is rated by the customer

Order, the customer and the delivery boywill participate simultaneously in a relationship. Because of this fact, when we consider cardinality we need to consider it in the context of two entities simultaneously relative to the third entity.

THE CARDINALITY



We will analyze the cardinalities of all the possible entities involved:-

1. **Relationship of Customer with delivery_boy and order**- Here the cardinality of the customer is M with respect to the delivery_boy and order. It is so because **at one time multiple say M Customers can rate multiple delivery boys say 1 for mutiple order. Hence the Cardinality is M.**

2. **Relationship of order with delivery_boy and customer-**Here the cardinality of the order is N with respect to the Customer and delivery_boy. It is so because **at one time mutiple orders say N can be rated by mutiple customers say M customers for a particular delivery_boy which is 1 delivery_boy. Hence the Cardinality is 1.**

3. **Relationship of delivery_boy with Customer and order**- Here the cardinality of the delivery_boy is N with respect to the Customer and Cart. It is so because **at one time 1 delivery boy can be rated by**

multiple customers say M for multiple orders say(N) . Hence the Cardinality is 1.

Converting E-R into Relational Schemas

We converted E-R into relationship schemas as follows by following the rules related to including foreign key where required and creating extra tables in schema for M to N relationships and also for relationships with attributes. The schema we made is as follows:-

product(Product_ID,Name,Price,Brand,Measurement,Unit,Admin_ID,Category_ID)

customer(Customer_ID,First_Name,Last_Name,Email,Mobile No, Password)

category(Category_ID,Category_Name)

admin(Admin_ID, First Name, Last Name, Admin_Password)

seller(Seller_ID, First_Name, Last_Name, Email, Phone_Number, Password, Place_Of_Operation,Admin ID)

cart(Cart ID,Total_Value, Total_Count,Final_Amount,Offer_ID)

orders(Order ID, Mode, Amount, Order_Time, State, City, House_Flat_No, Pincode, Cart_ID, Date, Delivery_Boy_ID)

product_feedback(Review ID,Rating, Review_Body,Product_ID,Customer_ID,Review_Date)

sells(Seller_ID,Product_ID, No_of_Product_Sold)

admin_views(Admin_ID, Order_ID, No_Of_Orders_Viewed)

selects(Customer ID,Category ID)

associated_With(Customer_ID, Cart_ID,Product_ID)

delivery_boy(Delivery_Boy_ID,First_Name,Last_Name,Password,Mobile_No,Email,Average_Rating,Admin_ID)

rates_order_delivery(Order_ID,Delivery_Boy_ID,Customer_ID,Rating_Given)

offer(Offer_ID,Promo_Code,Percentage_Discount,Min_OrderValue,Max_Discount,Admin_ID

EXPLANATION OF DDL

DDL is actually defined as a data definition language. They are SQL commands dealing with descriptions of database schema used to create/modify structure of database objects. They are used to create,modify and delete database structure, normally not used by general user. Eg ALTER ,DELETE etc.
Now we will visit different tables formed due to entities and see the data types formed of many attributes.

TABLE ADMIN

This table mainly houses all the details about the admin and their information. It is used to store people responsible for adding sellers selling the products. The Admin also adds the product and is able to view them. The Admin adds different offers as well as adds offers according to requirement. The main job of the admin is to manage the communication between the seller, products and customers.
Attributes and their data types

1. Admin_ID(Stores the ID of the admin when they register into to keep track of the admin and store them easily and help in for further uses)
   a) Data type-int This is done as it is easier for us to look at data in the form of integers rather than alphabets such that we can easily differentiate between the data. Alphabets may cause problems where figuring may be difficult. Identification becomes easier.

b) Primary Key- It is the primary key as it will be different for all the people. It might happen that names may be the same. Password also not set as it may be of huge length. It becomes easier to search through the ID which is of the form of characters and only through int one can differentiate. Tracking becomes easier through this.
c) NOT NULL- The ID can never be NULL. It is not possible to add an Admin where the primary key is null. Keeping track would become a big issue. Here the person would not be able to register without using the primary key. The data itself would not be there. Null data would make other data obsolete.

2. First_Name(Stores the first name to track the admin)
   i) Data Type- Varchar(15) The first name can not be in integers and only would contain characters. The length is set to 15 as per the length requirements.
   ii) NOT NULL- The first name can not be null otherwise identifying the records becomes difficult. A null person would be there adding datas where keeping the data would be difficult. Here there is no option for the user but to give it.

3. Last_Name(stores the last name of the admin to keep track of the person)
   i) Data Type- Varchar(15) The last name can not be in integers and only would contain characters. The length is set to 15 as per the length requirements.
   ii) NULL- The last name can easily be null. It might happen that the person does not have a last name or does not want to mention it. This NULL data also will not cause any problem during putting of any data as records. It is set to NULL as the data is not so important where the user may or may not give.

4. Admin_Password(stores the password of the person so that he can keep his account secure)
   i) Data Type- Varchar(15) We have set the password to varchar as it can not be full integer otherwise the password is not strong and may be identified very easily. Hence it is set to varchar with the length being set to 15.
   ii) NOT NULL-The password can not be null because without the password login would not be possible. Here the tracking of the admin would become tough. The password is needed to check about the conditions and to make the person enter. It helps to keep the person secure. The admin may become obsolete without the password.

TABLE CART

This table is used to store all the products selected into the cart. The cart has an association with the customer and the product. It mainly maintains the total value of the

products and as well as keeps the total count of the product. The cart would also contain the final amount after applying the offers through the offer ID passed. The cart becomes helpful for the user in storing the products and thereby remembering them.It makes the orders as well.The main job is making the orders.

Attributes and their data types

1. Cart_ID(Stores the ID of the cart so that each customer can keep track and keep updating their products. One person would also be able to keep the orders and finally make the orders)
   i) a) Data type-int It is int because it is the ID which has to be stored in int. With the alphabets it is difficult to keep track of the same. Alphabets will cause issues where one person would not be able to keep track of the same. It also may cause issues while putting.
   b) Primary Key- It has been made the primary key as it would easily differentiate between different carts. One would also be able to keep the track of the records and easily identify different records. The data would not be repeated and unique identification would be possible. It is also of integer type which helps.The total value and total count would not be correct as they might be same as well. The int data type removes many complexities
   c) NOT NULL- The ID can never be NULL as if the ID is null the primary key itself would not even exist. One would not be able to keep track of the records. It would cause issues while taking out queries. The data would get deleted just like that,Without the ID the cart would not even exist. It would make it obsolete as well
   d) Auto_increment- The auto increment automatically increments the primary key where we need not again put the values and it automatically puts the values in the ID. By the load and the time is reduced. We need not add the data again and again. The value is unique and thus is not repeated at all.

2. Total_Value(It tells about the total price the products kept would have. The customer would be able to keep an estimate about the price their and that would add all of the products that it would have)
   a) Data Type int- The data type would be int as the price can never be in any alphabet form. It is always in the numerical form.
   b) NULL- It might happen that the customer has not bought anything. There at times the price may be 0 as well. So even if something does not have the value it would not impact the tracking of any records. So null value will not affect anything as such.

3. Total_Count(helps to keep the count on the number of the products taken in. It tells the customer about the no of products

a) DATA TYPE int- Here the data type has to be integer as alphabets can not be put into count
b) NULL- It might happen that the data might be null because no products have been sold till now and in that case the data might be null. In this case too data will not be troubled even if it is null.

4. Offer_ID-(Stores the ID of the offer when they register into to keep track of the offers and store them easily and help in for further uses)

a) Data type-int This is done as it is easier for us to look at data in the form of integers rather than alphabets such that we can easily differentiate between the data. Alphabets may cause problems where figuring may be difficult. Identification becomes easier.
b) NULL- It might happen that no oofer has applied to the cart. There at times the offer may not ne there only. So even if something does not have the value it would not impact the tracking of any records. So null value will not affect anything as such.

5. Final_Amount- It denotes the total amount after the offer has been applied to the order. The offer would naturally reduce the cost upon applying the discount. The percentage or the total amount would be subtracted from the cost. The final amount tells the customer the actual cost

1) Data type-int This is done as it is easier for us to look at data in the form of integers rather than alphabets such that we can easily differentiate between the data. Alphabets may cause problems where figuring may be difficult. Identification becomes easier.
2) NOT NULL- The data by default can not be null as if a cart exists then there must be some amount that must be there. It is quite improbable and illogical to have a cart and then not have a price. The price must be there if there is some product added into the cart. It is also required to kep a track wether the products are actually being paid in. Otherwise no mount would be there.

6. Foreign Key- **Offer ID-** According to the relationship our offer is being applied to the cart hence it becomes essential to connect it with the Offer ID such that we can connect the two tables. Also it has a relation such that we can identify the offer a applying to the carttherebyhelping us to determine the a offer has been activated.
ON DELETE CASCADE-Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here we want to delete the rows of the offer if the cart itself gets deleted.One would not be able to keep track of the records. It would cause issues while taking out queries. The data would get deleted just like

that,Without the final amount the cart would not even exist. It would make it obsolete as well
ON UPDATE CASCADE- - We have made the foreign key as an offer ID key. Now if the offerr is updated then the cart will also be updated.

TABLE CATEGORY-

This is the table that has been used to categorize the products into parts. It has also been seen that categorizing products makes it easy to use for the user. It is selected by the customer. The category mainly categorizes the products. The category has the products as well.

1. Category_ID(A number is given to the category to keep a track of it such that one can easily identify with it. The category ID gives an identity,)
    a) Data Type int- The data type has to be int as alphabets would make it difficult to recognize. The integer helps us to define them properly. The track is also easily kept using the integer variable
    b) Primary Key- This is the only key that can be kept unique altogether. The category name may be the same and would not be able to uniquely identify the data properly. It is not alphabetical too. The tracking of records would be easily done.
    c) Auto_Increment- The keys would automatically increment on the entry and we need to always increase. This reduces the time and labor put into and it would be done automatically. It is based on logic where the value is incremented as it is a primary key and it is unique as well. The different data types get stored as well
2. Category Name(It is basically name of the category which can be of different types)
    a) Data type- Varchar(15)- The name can not be in numbers, It is of character type and it is set as a maximum of 15 characters.
    b) NOT NULL- A data type should have a value so that the product can be identified. Without any name the category would not exist. The existing records might get affected. Without the name it might become difficult.

TABLE CUSTOMER

This is one of the main tables where the details about the customer are stored. Here the customer is allowed to make orders through the cart. They are allowed to give reviews as well as select the category. The Customer also gives specific ratings to delivery boys for some order.The Customer is mainly associated with the cart and the product.

1. Customer ID(Stores the ID of the customer when they register into so that they can easily buy and for other help in for further uses)
    a) Data Type- int- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.
    b) PRIMARY KEY- The customer ID is taken as the primary key as it helps to differentiate the data properly where we can keep the track of the records. The customer ID can easily store the data for each customer uniquely. The other data types are passwords which is not an appropriate way to have a primary key due to its constraints and it is not something which is easy to store. Similarly first name and last name are both character types which first of all may not be unique as well as not the correct way to store the ID. The email ID is also very long and will be difficult to use while using queries. Same goes for the contact numbers where the person will not be able to store it properly due to its length and storage. It is also not repeated too, hence the primary key. Tracking will be easy and as it is of int type it will be stored easily.
    c) Auto_increment- The auto increment automatically increments the primary key where we need not again put the values and it automatically puts the values in the ID. By the load and the time is reduced. We need not add the data again and again. The value is unique and thus is not repeated at all.

2. First Name- It is used to store the first name of the customer, which gives an identity and helps us to keep the track record and identify the user.
    a) Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.
    b) NOT NULL-IThe first Name has to be there as otherwise keeping the records and identifying the person would be very difficult. So if we keep it major problems would be encountered.
3. Last Name- It is used to store the last name of the customer, which gives an identity and helps us to keep the track record and identify the user.
    a) Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.
    b)NULL-It is kept null as a person may not want to write the name and that is kept in mind. The null value would not affect anything during the query or while maintaining the record.

4. Email_ID- It is mainly used to store the Email-ID which helps in maintaining the record of the person while the person buys the product. It helps to keep the extra information.

a) Data Type- Varchar(35)- This data type is used as the email-id can not be stored using numbers. The length has been kept at a maximum length of 35.

b)NOT NULL-It is kept null as a person may not want to write the email and that is kept in mind. The null value would not affect anything during the query or while maintaining the record.

5. Password(stores the password of the person so that he can keep his account secure)

i) Data Type- Varchar(20) We have set the password to varchar as it can not be full integer otherwise the password is not strong and may be identified very easily. Hence it is set to varchar with the length being set to 20.

ii) NOT NULL-The password can not be null because without the password login would not be possible. Here the tracking of the customer would become tough. The password is needed to check about the conditions and to make the person enter. It helps to keep the person secure. The customer may become obsolete without the password.

6. Mobile_No(Stores the mobile no of the person so that another extra information can be stored)

a) Data Type- Varchar(14) - The Mobile No has been take in the form of characters just to ease the entry of data into the table
b) NOT NULL- The mobile no becomes important as the person needs to be contacted hence it is not a NULL data. We can not make it a NULL data.

TABLE ORDERS- This Table mainly used to carry out orders. These will have all the details about the orders which are described in the attributes. Here the orders are taken from the customer and then delivered accordingly. It will carry all such values required for specifying the orders. The cart would do the same for placing the orders. The Admin can also regulate by viewing them separately.

c) Order_ID(Stores the ID of the order when the customer buys the product)
    a) Data type-int This is done as it is easier for us to look at data in the form of integers rather than alphabets such that we can easily differentiate between the data. Alphabets may cause problems where figuring may be difficult. Identification becomes easier.
    b) Primary Key- It is the primary key as it will be different for all the orders. It might happen that the mode of payment,amount,Pincode,order time,flat

no,city ,state may be the same for many orders. The value needs to be different and unique otherwise there is no such use of the primary key.It becomes easier to search through the ID which is of the form of characters and only through int one can differentiate. Tracking becomes easier through this.

c) NOT NULL- The ID can never be NULL. It is not possible to add an order ID where the primary key is null. Keeping track would become a big issue.An order without an ID will not ensure the correct taking of the data and all other information would become impossible to handle. The data itself would not be there. Null data would make other data obsolete.

2. Mode- It denotes the Mode of Payment by which the order is being made.It allows the user to choose freely how he wants to pay. It basically allows the user to choose the form of payment they want to.

a) Data Type- Varchar(15)- This data type is used as the mode can not be stored using numbers. The length has been kept at a maximum length of 15.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

3. Amount- It is the amount the customer will have to pay where it would sum the amount of all the possible values. The net money to be paid by the user is shown.

a) Data Type- Int- This data type is used as the number can not be stored using alphabets.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

4. City- It denotes the city where the order needs to be delivered. The place of the customer from where he is ordering.

c) Data Type- Varchar(15)- This data type is used as the city can not be stored using numbers. The length has been kept at a maximum length of 15.
d) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

5. State- It denotes the state where the order needs to be delivered. The place of the customer from where he is ordering.

a) Data Type- Varchar(20)- This data type is used as the state can not be stored using numbers. The length has been kept at a maximum length of 20.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

6. House_Flat_No-It denotes the state where the House_Flat_No needs to be delivered. The place of the customer from where he is ordering.
a) Data Type- Varchar(30)- This data type is used as the state can not be stored using numbers. The length has been kept at a maximum length of 30.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

7. Order_Time- It denotes the time at which the order was made. It would tell the admin at the specific time the user is making the order.
a) Data Type- Varchar(20)- The time format has been stored in a specific format which would need alphanumeric data. It was our way of taking care of this data type.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.
8. Pincode- It denotes the pincode where the order needs to be delivered. The place of the customer from where he is ordering.
a) Data Type- Varchar(20)- The pincode format has been stored in a specific format for convenience. It was our way of taking care of this data type.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.
9. Date- It denotes the time at which the order was made. It would tell the admin at the specific time the user is making the order.
a) Data Type- Varchar(20)- The date format has been stored in a specific format which would need alphanumeric data. It was our way of taking care of this data type.
b) NULL-It is kept null as an order may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.
10. **FOREIGN KEY- DELIVERY_BOY_ ID-** It is the ID of the delivery boy. According to the relationship our delivery boy is supplying the order hence it becomes essential to

connect it with the delivery boy ID such that we can connect the two tables. Also it has a relation such that we can identify the delivery boy while making the order thereby helping us to determine the delivery boy is supplying the order.

11. **FOREIGN KEY- CART ID-** It is the ID of the cart. According to the relationship our cart is making the order hence it becomes essential to connect it with the cart ID such that we can connect the two tables. Also it has a relation such that we can identify the cart making the order thereby helping us to determine the customer making the order.

a) ON DELETE SET NULL- As it is foreign key, and if by chance the cart id is removed then the cart would be null now as the cart might have been deleted. Hence there is no point in storing the data. So that data is set to NULL so that there is no issue in identifying it.

TABLE PRODUCT- This is one of the main tables where all the products are stored. From the product table only the customer is associated such that they can see and select the products. Every detail here is present here. Products would have a category. It would be added by the admin and then sold by the seller. The products form the crux which interacts with the admin,customer and seller. It has a ternary relationship.

1. Product_ID(Stores the ID of the product which the customer buys or the seller sells or the admin views)

a) Data Type- int- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.

b) Primary Key- It is the primary key as it will be different for all the products. It might happen that the mode of name,price,brand,measurement may be the same for many orders. The value needs to be different and unique otherwise there is no such use of the primary key. It becomes easier to search through the ID which is of the form of characters and only through int one can differentiate. Tracking becomes easier through this.

c) NOT NULL- The ID can never be NULL. It is not possible to add a product ID where the primary key is null. Keeping track would become a big issue.An product without an ID will not ensure the correct taking of the data and all other information would become impossible to handle. The data itself would not be there. Null data would make other data obsolete.

2. Name- The name of the product would be there. It denotes the identification of the product.

a) Data Type- Varchar(25)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 25.

b) NOT NULL- The data by default can not be null as if a name exists then there must be some name that must be there. It is quite improbable and illogical to have a product and then not have a name. The name must be there if there is some product added into the cart. It is also required to kep a track wether the products are actually being paid in. Otherwise no amount would be there.

3. Price- Here the price of the product is mentioned which appears as a parameter towards determining how our product will move and also appears in many other tables. Without the price one can not actually get the total amount. Following are the data types

DATA TYPE

a) INT- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.
b) NOT NULL- The data by default can not be null as if a cart exists then there must be some amount that must be there. It is quite improbable and illogical to have a cart and then not have a price. The price must be there if there is some product added into the cart. It is also required to kep a track wether the products are actually being paid in. Otherwise no amount would be there.

4. Brand- Here the product's brand name is mentioned so that the person can make the choices after seeing the brand name. The brand name would stand out and would result in easy identification. Brand makes the user easily select the product.

a) Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.
b) NULL-It is kept null as a product may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

5. Measurement- It would denote how a particular product is measured by a person. The measurement would give an idea about what the person wants to purchase. It would tell in which form the data would be present. The units may be in kg,litre etc. It mainly gives an idea about the quantity.

a) Data Type- Varchar(15)- This data type is used as the measurement can not be stored using numbers. It has to be alphanumeric to store data like 1 kg etc.The length has been kept at a maximum length of 15.
b) NULL-It is kept null as a product may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

6. Unit- It denotes the line of action for the measurement. It would give the total amount of product what is given. It would give us an idea about what all is actually present in the

product that we have bought and denotes the amount which is available. Their measurement wil be done in Kilogram,litre etc.It would help set the amount for the product

a) Data Type- Varchar(15)- This data type is used as the measurement can not be stored using numbers. It has to be alphanumeric to store data like 1 kg etc.The length has been kept at a maximum length of 15.
b) NOT NULL- The value can not be null because some unit has to be there otherwise the product can not be defined. Without the value of the unit the product can not be measurement. There will be major measurement issues pertaining to the unit. A product without unit would not signify anything and it is important. The data would be useless hence it is set to NOT NULL.

7. **FOREIGN KEY- ADMIN_ID-** It is the ID of the admin. According to the relationship, our admin is adding to the products,it becomes essential to connect it with the Admin_ID such that we can connect the two tables. Also it has a relation such that we can identify the Admin adding which product thereby helping us to determine the admin who is adding the product.

ON DELETE SET NULL- As it is foreign key, and if by chance the foreign id is removed then the admin would be null now as the admin might have been deleted. Hence there is no point in storing the data. So that data is set to NULL so that there is no issue in identifying it.

8. **FOREIGN KEY- CATEGORY_ID-** It is the ID of the category. According to the relationship with the product, the product is being categorized into categories,it becomes essential to connect it with the Category_ID such that we can connect the two tables. Also it has a relation such that we can identify which category the product belongs to.

a) ON DELETE SET NULL- As it is foreign key, and if by chance the foreign id is removed then the cart would be null now as the cart might have been deleted. Hence there is no point in storing the data. So that data is set to NULL so that there is no issue in identifying it.

TABLE SELLER- This table contains all the details about the seller. The seller is that person who is the one selling the products in this system. He is essential as he is the one selling all the products and supplying them.The seller is being added by the admin and they help in adding the products as well.

1. Seller_ID(Stores the ID of the seller when they register into so that they can easily sell and for other help in for further uses)
    a) Data Type- int- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.
    b) Primary Key- The Seller ID is taken as the primary key as it helps to differentiate the data properly where we can keep the track of the records. The seller ID can easily store the data for each seller uniquely. The other data types are passwords which is not an appropriate way to have a primary key due to its constraints and it is not something which is easy to store. Similarly first name and last name are both character types which first of all may not be unique as well as not the correct way to store the ID. The email ID is also very long and will be difficult to use while using queries. Same goes for the contact numbers where the person will not be able to store it properly due to its length and storage. The Place of Operation also may be the same for different ID's. It is also not repeated too, hence the primary key. Tracking will be easy and as it is of int type it will be stored easily.
    c) NOT NULL- The ID can never be NULL. It is not possible to add a seller where the primary key is null. Keeping track would become a big issue. Here the person would not be able to register without using the primary key. The data itself would not be there. Null data would make other data obsolete.

2. First Name- It is used to store the first name of the seller, which gives an identity and helps us to keep the track record and identify the user.

1. Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.
2. NOT NULL- The first Name has to be there as otherwise keeping the records and identifying the person would be very difficult. So if we keep it, major problems would be encountered.

3. Last Name- It is used to store the last name of the seller, which gives an identity and helps us to keep the track record and identify the user.

3. Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.

4. NULL-It is kept null as a person may not want to write the name and that is kept in mind. The null value would not affect anything during the query or while maintaining the record.

4. Password(stores the password of the person so that he can keep his account secure)

i) Data Type- Varchar(20) We have set the password to varchar as it can not be full integer otherwise the password is not strong and may be identified very easily. Hence it is set to varchar with the length being set to 20.

ii) NOT NULL-The password can not be null because without the password login would not be possible. Here the tracking of the seller would become tough. The password is needed to check about the conditions and to make the person enter. It helps to keep the person secure. The seller may become obsolete without the password.

5. Phone_Number(Stores the mobile no of the person so that another extra information can be stored)

5. Data Type- Varchar(14) - The Phone No has been take in the form of characters just to ease the entry of data into the table
6. NOT NULL- The phone no now becomes important as the person needs to be contacted hence it is not a NULL data. We can not make it a NULL data.

6. Email_ID- It is mainly used to store the Email-ID which helps in maintaining the record of the person while the person sells the product. It helps to keep the extra information.

a) Data Type- Varchar(35)- This data type is used as the email-id can not be stored using numbers. The length has been kept at a maximum length of 35.

b)NOT NULL- The email id now becomes important as the person needs to be contacted hence it is not a NULL data. We can not make it a NULL data.

7. Place_of_Operation - The seller would definitely operate from some place.The place of operation becomes important as one might want to know from where the products are actually coming from. Here the Admin gets to know from where it comes through which one can easily tell that time of processing it would take. It basically tells from where the product is coming from.

a)Data Type- Varchar(15)- This data type is used as the Place_of_Operation can not be stored using numbers. The length has been kept at a maximum length of 15.

b)NULL-It is kept null as a seller may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

8. FOREIGN KEY- ADMIN_ID- It is the ID of the admin. According to the relationship our admin is adding the sellers,it becomes essential to connect it with the Admin_ID such that we can connect the two tables. Also it has a relation such that we can identify the Admin adding which sellert thereby helping us to determine the admin who is adding the seller.

ON DELETE SET NULL- As it is foreign key, and if by chance the foreign id is removed then the admin would be null now as the admin might have been deleted. Hence there is no point in storing the data. So that data is set to NULL so that there is no issue in identifying it.\

TABLE PRODUCT_FEEDBACK

This table is mainly used to take feedback about a certain product in the form of a particular attribute. Through the feedback a proper evaluation can be done and the future decision about the products can be taken. We can take care of how one can deal with that product. By this the proper segregation happens about what one should do with the product. A person would also be able to determine which type of customers are liking and what background does that person have.A perfect evaluation of products is also done. We get to know what is the market value of the product and what is its future and likeability among the customers. The Product Feedback is a weak entity which would not exist without the product table. Without the product the feedback ceases to exist. The feedback is always given by the customer.

1. Review_ID(Stores the ID of the review which the customer would give when they register into)
    a) Data type-int This is done as it is easier for us to look at data in the form of integers rather than alphabets such that we can easily differentiate between the data. Alphabets may cause problems where figuring may be difficult. Identification becomes easier.
    b) NOT NULL- The ID can never be NULL. It is not possible to add a review ID where the primary key is null. Keeping track would become a big issue.An review without an ID will not ensure the correct taking of the data and all other information would become impossible to handle. The data itself would not be there. Null data would make other data obsolete.
    c) Auto_increment- The auto increment automatically increments the primary key where we need not again put the values and it automatically puts the values in the ID. By the load and the time is reduced. We need not add the data again and again. The value is unique and thus is not repeated at all.

2. Rating- It is the value or parameter by which the customer would judge a product or service. It helps in judging how good or bad the product is. Rating would just be a value to judge a product from other customers to see when they buy that particular product and also serves as a parameter for the admin as well as to whether a particular product be allowed to sell.

a) Data Type - int- The rating has been set to integer as giving numbers makes it easier to tell about how the product actually is. It is easy to maintain and keep a track of. Alphabets would not be to the point determining the product and would not give a definite result. The user may type anything. It is better to keep it in a range say of 1-5. Hence the integer.
b) NULL- It might be the case where the person may not give the rating. It is not necessary that one has to give a rating hence it is set to null. Hence we have kept the option for that.

3. Review_Body- It is mainly given for the user to give a more detailed description about the review. It might happen that the rating is not enough and the user would like to describe more on it. This allows the user to express everything and also gives a detailed description to the one selecting the product. It might happen that the admin or some other person would have to see more details. There has been a window for the person to enter the data.

a) Data Type- Varchar(50)-his data type is used as the review body can not be stored using numbers. The length has been kept at a maximum length of 15.
b) NULL- It might be the case where the person may not give the rating. It is not necessary that one has to give a review body hence it is set to null. Hence we have kept the option for that.

4. Review_Date- It denotes the date at which the review was made. It would tell the date for the review.

c) Data Type- Varchar(20)- The date format has been stored in a specific format which would need alphanumeric data. It was our way of taking care of this data type.
d) NULL-It is kept null as a review may be deleted if wanted to so in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

5. **FOREIGN KEY- CUSTOMER_ID-** It is the ID of the customer. According to the relationship with the customer, the review is given by the customer,it becomes essential to connect it with the Customer_ID such that we can connect the two tables. Also it has a relation such that we can identify which customer the review belongs to.

b) ON DELETE SET NULL- As it is foreign key, and if by chance the foreign id is removed then the customer would be null now as the customer might have been deleted. Hence there is no point in storing the data. So that data is set to NULL so that there is no issue in identifying it.

6.PRIMARY KEY- A COMPOSITE KEY COMBINING BOTH THE REVIEW ID AND PRODUCT ID)

Here we have taken both the product id with the review id because product feedback already is a weak entity where the feedback can not exist without the product. The product and the feedback are related hence we need to store it together to show an entry. They both exist with each other. Otherwise only 1 key would not have any meaning.

ON DELETE CASCADE-Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here we want to delete the rows of the review if the product itself gets deleted. The same reasoning of the weak entity comes that of the product itself is not there then the review row should also not be there.

TABLE OFFERS

This Table mainly used to store the offers. These will have all the details about the offers which are described in the attributes. Here the offers would be applied to the cart. It will carry all such values required for specifying the offers.We would keep on applying the offers as required. The Admin will be able to add the offers. The offers would basically reduce the price of the product as bought by the user. It would give the offer as per the terms and conditions be it maximum amount to be bought or storing the promo code. It would take the minimum discount and percentage discount. It has a ternary relationship with rates_order_delivery.

1.Offer_ID(Stores the ID of the offer which id applied to the cart and can be added by the admin)

a)Data Type- int- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.

b) Primary Key- It is the primary key as it will be different for all the offers. It might happen that the total discount or the percentage or the minimum value given would be same. The value needs to be different and unique otherwise there is no such use of the primary key. It becomes easier to search through the ID which is of the form of characters and only through int one can differentiate. Tracking becomes easier through this.

c)NOT NULL- The ID can never be NULL. It is not possible to add a offer ID where the primary key is null. Keeping track would become a big issue.An offer without an ID will not ensure the correct taking of the data and all other

information would become impossible to handle. The data itself would not be there. Null data would make other data obsolete.

d) Auto_increment- The auto increment automatically increments the primary key where we need not again put the values and it automatically puts the values in the ID. By the load and the time is reduced. We need not add the data again and again. The value is unique and thus is not repeated at all.

2. Promo Code- The promo Code is the code required to activate a particular offer. The user has to enter that particular promo code then only that offer would get activated. It would have a particular so that one can protect the originality of code such that any code written is not accepted. A proper check is seen after then only the discount is applied.

A) Data Type- Varchar(20)- The promo code format has been stored in a specific format which would need alphanumeric data. It was our way of taking care of this data type.
B) NOT NULL- The data by default can not be null as if a offer exists then there must be some promo code that must be there. It is quite improbable and illogical to have a offer and then not have a promo code. The promo code must be there if there is some product added into the cart. It is also required to keep a track wether the offers are being made according to a specific rule and not being given just like that. If no promo code then the offer would cease to exist.

3. Min_OrderValue- It is the minimum order value which is required to actually get the discount. Without the minimum value the offers may be applied just like that so the discount have to be given in a controlled manner. So we have given a minimum order forom which the discount may start to put a cap and carry out proper business.

a) INT- The data can be int only a ,value can be nothing other than a integer.
b) NOT NULL- The value can not be null as clearly without the minimum order value the offer will not get validated as there must be some minimum value so that the order can be capped and the discount is not applied to any amount possible otherwise it would non uniformity. Hence the value should exist as without the offer would be obsolete.

4. Max_Discount- It is the maximum discount value which can be provide to a particular order. Without the maximum discount the offers may go to a p. So we have given a minimum order forom which the discount may start to put a cap and carry out proper business.

c) INT- The data can be int only, a ,value can be nothing other than a integer.

d) NOT NULL- The value can not be null as clearly without the maximum discount the offer will not get validated as there must be some maximum discount so that the order can be capped with a maximum value of discount so that the profit can me maximised at most. Without the maximum discount it may happen that unimaginable discount with respect to percentage will be offered which would affect the market. Hence the value is NOT NULL.

5. Percentage_Discount- It is the percentage of discount that is applied on our product. The discount percentage would stay till a particular value. It would apply discount with (discount value)/100 in value. We will subtract this value to get the final discount from the product. Now this percentage after some maximum value is useless. If this pertains the value keeps on increasing and it would affect the company, so cap is applied on the discount. If 50% discount is applied on a product with high pricing it would affect the economic so a max value is set as defined above. Also the percentage discount keeps track of the total discount and gives a breather to the customer too.

INT- The data can be int only ,a value can be nothing other than a integer. The percentage can not be measured in alphabets. So it is integer only.

NOT NULL- The value can not be null as clearly without the discount percentage the offer will not get validated as there must be some discount percentage so that the order can be capped with a percentage discount so that the profit can me maximised at most. Without the percentage discount it the total discount for all the stakeholders becomes difficult to check which affects the market. Hence the value is NOT NULL.

6. FOREIGN KEY- ADMIN ID- It is the ID of the admin. According to the relationship our admin is adding the offers,it becomes essential to connect it with the Admin_ID such that we can connect the two tables. Also it has a relation such that we can identify the Admin adding which offer thereby helping us to determine the admin who is adding the offer.

ON UPDATE CASCADE- As it is foreign key, and if by chance the foreign id is updated then the admin would be updated now as the admin might have been updated. So if the admin is deleted then the whole offer row would get deleted

TABLE DELIVERY BOY

This table contains all the details about the delivery boy. The delivery boy is that person who is the one delivering the products in this system. He is essential as he is the one delivering all the products and supplying them.The delivery boy is being added by the admin and they help in delivering the products as well. The delivery boy is being rated by a customer with respect to a order as well. He ill be assigned to a order as well. It is participating in a Ternary Relationship.

2. Delivery_boy_ID(Stores the ID of the delivery boy when they register into so that they can easily deliverl and for other help in for further uses)
    d) Data Type- int- The data type is int as it helps in easily identifying the attribute. The ID is always in the form of int as alphabets would cause issues in the identification. The tracking of records would be difficult as well.
    e) Primary Key- The Delivery_boy_ID is taken as the primary key as it helps to differentiate the data properly where we can keep the track of the records. The Delivery_boy_IDcan easily store the data for each delivery boy uniquely. The other data types are passwords which is not an appropriate way to have a primary key due to its constraints and it is not something which is easy to store. Similarly first name and last name are both character types which first of all may not be unique as well as not the correct way to store the ID. The email ID is also very long and will be difficult to use while using queries. Same goes for the contact numbers where the person will not be able to store it properly due to its length and storage. The other attributes also may be the same for different ID's. It is also not repeated too, hence the primary key. Tracking will be easy and as it is of int type it will be stored easily.
    f) NOT NULL- The ID can never be NULL. It is not possible to add a delivery boy where the primary key is null. Keeping track would become a big issue. Here the person would not be able to register without using the primary key. The data itself would not be there. Null data would make other data obsolete.

2. First Name- It is used to store the first name of the delivery boy, which gives an identity and helps us to keep the track record and identify the user.

7. Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.
8. NOT NULL- The first Name has to be there as otherwise keeping the records and identifying the person would be very difficult. So if we keep it, major problems would be encountered.

3. Last Name- It is used to store the last name of the delivery boy, which gives an identity and helps us to keep the track record and identify the user.

9. Data Type- Varchar(15)- This data type is used as the name can not be stored using numbers. The length has been kept at a maximum length of 15.

10. NULL-It is kept null as a person may not want to write the name and that is kept in mind. The null value would not affect anything during the query or while maintaining the record.

4. Password(stores the password of the person so that he can keep his account secure)

i) Data Type- Varchar(20) We have set the password to varchar as it can not be full integer otherwise the password is not strong and may be identified very easily. Hence it is set to varchar with the length being set to 20.

ii) NOT NULL-The password can not be null because without the password login would not be possible. Here the tracking of the seller would become tough. The password is needed to check about the conditions and to make the person enter. It helps to keep the person secure. The seller may become obsolete without the password.

5. Phone_Number(Stores the mobile no of the person so that another extra information can be stored)

11. Data Type- Varchar(14) - The Phone No has been take in the form of characters just to ease the entry of data into the table
12. NOT NULL- The phone no now becomes important as the person needs to be contacted hence it is not a NULL data. We can not make it a NULL data.

6. Email_ID- It is mainly used to store the Email-ID which helps in maintaining the record of the person while the person delivering the product. It helps to keep the extra information.

a) Data Type- Varchar(35)- This data type is used as the email-id can not be stored using numbers. The length has been kept at a maximum length of 35.

b)NOT NULL- The email id now becomes important as the person needs to be contacted hence it is not a NULL data. We can not make it a NULL data.

7. Average Rating- The average rating denotes the total rating given by all in the scale of 1 to 5 by the total number of people who have given the rating.It is done so to ease the process where we would be able to handle in the database as well as the customer would find it easier as well to rate. The rating is given to improve the systems as a whole and ensure a efficient system overall. The rating helps to make the order delivery to happen in a prompt manner and also allows the customer to get to kmow about the person delivering where they can decide accordingly. It will make the system more efficient as well.The rating would be the way to judge how the delivery boy is working and would enable us to judge and supervise how the system is working. The rating forms an essential part in the working.

DECIMAL- It is clearly mentioned that the rating can only be saved in decimal as iwth numbers the work becomes difficult to judge and we do not store it alphanumeric form so that usern also does not give any input according to itself and any unwanted is rejected immediately. Keeping track of it also becomes difficult too. Rating in number becomes easier as well.

NULL- The value is null because it might happen some costumer has not given the rating to the deliver

6. FOREIGN KEY- ADMIN ID- It is the ID of the admin. According to the relationship our admin is adding the delivery boys,it becomes essential to connect it with the Admin_ID such that we can connect the two tables. Also it has a relation such that we can identify the Admin adding which offer thereby helping us to determine the admin who is adding the offer.

ON UPDATE CASCADE- As it is foreign key, and if by chance the foreign id is updated then the admin would be updated now as the admin might have been updated. So if the admin is deleted then the whole delivery boy row would get deleted

RELATIONSHIP TABLE SELLS This is the table which is made due to the relation of sell which is between seller and the product. It is a M-N relation due to which the table is made, where there would be multiple entries. Here multiple sellers can sell multiple products.

1. No of Products Sold(It denotes the total no of products sold by the seller)
    a) Data Type- Int-his data type is used as the no of products sold can not be stored using alphabets. Hence the numeric data used.
    b) NULL- It might happen that no product is sold and in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

2.**PRIMARY KEY- A COMPOSITE KEY COMBINING BOTH THE SELLER ID AND PRODUCT ID)**

Here we have taken both the product id and the seller id because the table can exist only if both the product and seller IDs are joined simultaneously. The product and the seller are related hence we need to store it together to show an entry. They both exist with each other. Otherwise only 1 key would not have any meaning.

ON DELETE CASCADE-
Here the foreign key is also the primary key.Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here rows of the sells get deleted if the product or seller get deleted.

3. **FOREIGN KEY- PRODUCT_ID-** It is the ID of the product. According to the relationship with the seller, the product is sold by ,so it becomes essential to connect it with the Product_ID such that we can connect the two tables. Also it has a relation such that we can identify which seller is selling which product.

a) ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the product is deleted then there is no meaning of the sells row hence it is deleted.

4. **FOREIGN KEY- SELLER_ID-** It is the ID of the seller. According to the relationship with the seller, the sells the product and hence it has been put into

a) ON DELETE CASCADE- - We have made the foreign key as a seller ID key. Now if the seller is deleted then there is no meaning of the sells row hence it is deleted.

RELATIONSHIP TABLE ADMIN_VIEWS- This is the table which is made due to the relation of admin_views which is between admin and the orders. It is a M-N relation due to which the table is made, where there would be multiple entries. Here multiple admins can view multiple orders.

1. No_of_Orders_viewed(It denotes the total no of orders viewed)
    a) Data Type- Int-his data type is used as the no of orders viewed can not be stored using alphabets. Hence the numeric data used.
    b) NULL- It might happen that no order is viewed and in that case it might be NULL. The null value would not affect anything during the query or while maintaining the record.

PRIMARY KEY- A COMPOSITE KEY COMBINING BOTH THE ORDER ID AND ADMIN ID)
Here we have taken both the order id and the admin id because the table can exist only if both the order and admin IDs are joined simultaneously. The order and the admin are related hence we need to store it together to show an entry. They both exist with each other. Otherwise only 1 key would not have any meaning.
ON DELETE CASCADE-

Here the foreign key is also the primary key.Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here the rows of the admin views will get deleted if the order or admin get deleted.

2. **FOREIGN KEY- ADMIN_ID-** It is the ID of the admin. According to the relationship with the admin_views the admin which views the order get's stored,so it becomes essential to connect it with the Admin_ID such that we can connect the two tables. Also it has a relation such that we can identify which Admin is viewing the order.

b) ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the admin is deleted then there is no meaning of the admin views row hence it is deleted.

3. **FOREIGN KEY- ORDER_ID-** It is the ID of the order. According to the relationship with the admin views the order hence the order becomes essential
ON UPDATE CASCADE- - We have made the foreign key as an order ID key. Now if the order is updated then the admin_views will also be updated.

<u>RELATIONSHIP TABLE SELECTS-</u>
This is the table which is made due to the relation of selects which is between customer and the categories. It is a M-N relation due to which the table is made, where there would be multiple entries. Here multiple customers can select multiple categories.

PRIMARY KEY- A COMPOSITE KEY COMBINING BOTH THE CATEGORY ID AND CUSTOMER ID)
Here we have taken both the category id and the customer id because the table can exist only if both the customer and category IDs are joined simultaneously. The customer and the category are related hence we need to store it together to show an entry. They both exist with each other. Otherwise only 1 key would not have any meaning.
ON DELETE CASCADE-
Here the foreign key is also the primary key.Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here the rows of the selects will get deleted if the category or customer get deleted.

2. **FOREIGN KEY- CUSTOMER_ID-** It is the ID of the customer. According to the relationship with the selects where the customer selects the category, it becomes essential to connect it with the Cuatomer_ID such that we can connect the two tables. Also it has a relation such that we can identify which Admin is viewing the order.

ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the customer is deleted then there is no meaning of the selects row hence it is deleted.

3. **FOREIGN KEY- CATEGORY_ID-** It is the ID of the Category. According to the relationship with selects where the category is selected by the customer, hence category id becomes essential

ON DELETE CASCADE- - We have made the foreign key as an order ID key. Now if the category is deleted then there is no meaning of the category row hence it is deleted.

TABLE ASSOCIATED_WITH[TERNARY RELATIONSHIP]-

Here we are making a ternary relationship between 3 entities: the product, the customer and the cart. They all are related with each other through this relationship and even if one is removed the whole relationship is removed. Here the three entities come together sharing different attributes.Product, Cart and Customer will participate simultaneously in a relationship. Because of this fact, when we consider cardinality we need to consider it in the context of two entities simultaneously relative to the third entity. The cardinalities have been explained above in detail.

_PRIMARY KEY- A COMPOSITE KEY COMBINING THE PRODUCT ID , CART ID AND CUSTOMER ID)

Here we have taken both the category id ,the customer id and the cart id because the table can exist only if the customer,cart,product IDs are joined simultaneously. The customer, cart and product are related hence we need to store it together to show an entry. They all exist with each other. Otherwise only 1 key would not have any meaning.

ON DELETE CASCADE-

Here the foreign key is also the primary key.Here we delete the rows from the child table automatically, when the rows from the parent table are deleted. Here the rows of the associated will get deleted if the cart or customer or product gets deleted.

1. **FOREIGN KEY- CUSTOMER_ID-** It is the ID of the customer. According to the relationship associated_with where the customer is associated with both the cart as well as product, it becomes essential to connect it with the Customer_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how the association works.

ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the customer is deleted then there is no meaning associated with the row hence the row itself is deleted.

2. **FOREIGN KEY- CART_ID-** It is the ID of the cart. According to the relationship associated_with where the cart is associated with both the customer as well as product, it becomes essential to connect it with the Cart_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how the association works.

ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the cart is deleted then there is no meaning associated with the row hence the row itself is deleted.

3. **FOREIGN KEY- PRODUCT_ID-** It is the ID of the product. According to the relationship associated_with where the product is associated with both the customer as well as cart, it becomes essential to connect it with the product_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how the association works.

ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the product is deleted then there is no meaning associated with the row hence the row itself is deleted.

TABLE RATES_ORDER_DELIVERY[TERNARY RELATIONSHIP]-

Here we are making a ternary relationship between 3 entities: the delivery boy, the customer and the order. They all are related with each other through this relationship and even if one is removed the whole relationship is removed. Here the three entities come together sharing different attributes. Delivery Boy, order and Customer will participate simultaneously in a relationship. Because of this fact, when we consider cardinality we need to consider it in the context of two entities simultaneously relative to the third entity. The cardinalities have been explained above in detail.

RATING GIVEN- It is basically the actual rating that has been given to the delivery boy. The rating would be given on a scale of 1 to 5. It is done so to ease the process where we would be able to handle in the database as well as the customer would find it easier as well to rate. The rating is given to improve the systems as a whole and ensure a efficient system overall. The rating helps to make the order delivery to happen in a prompt manner and also allows the customer to get to kmow about the person delivering where they can decide accordingly. It will make the system more efficient as well.The rating would be the way to judge how the delivery boy is working and would enable us to judge and supervise how the system is working. The rating forms an essential part in the working

INT- It is clearly mentioned that the rating can only be saved in numbers as iwth numbers the work becomes difficult to judge and we do not store it alphanumeric form so that usern also does not give any input according to itself and any unwanted is rejected immediately. Keeping track of it also becomes difficult too. Rating in number becomes easier as well.

NOT NULL- The value can not be null as without this value there would be no meaning of the rating only. We would not realize only what this table wanted and accordingly the deletion of data would not be correctly handled. The changes would take place wherever wanted which would affect our database. The value would not be null as the null value would delete the data and affect the database. The value can not be null otherwise the coherency would not be there and implementation of queries would not be there properly.

_PRIMARY KEY- (ORDER ID)
Here we have taken the order ID to be our primary key as on the basis of the order the rating would be given to a delivery boy by a customer. The order ID becomes the main factor around which the whole concept of rating would revolve around. The rating can only be given if order is taken into account. The rating is only given when the order is taken as a base. A rating on delivery can only be done on basis of a order. Without the order ID the rating is obsolete.

2. **FOREIGN KEY- CUSTOMER_ID-** It is the ID of the customer. According to the relationship rates_order_delivery where the customer is associated with both the order as well as delivery boy, it becomes essential to connect it with the Customer_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how the association works.
   ON UPDATE CASCADE- As it is foreign key, and if by chance the foreign id is updated then the rates_order_delivery would be updated now as the customer have been updated. So if the customer is deleted then the whole rates_order_delivery row would get deleted

2. **FOREIGN KEY- ORDER_ID-** It is the ID of the cart. According to the relationship rates_order_delivery where the order is associated with both the customer as well as delivery_boy, it becomes essential to connect it with the order_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how therating works..

ON DELETE CASCADE- We have made the foreign key as a primary key. Now if the order is deleted then there is no meaning associated with the row hence the row itself is deleted.

3. **FOREIGN KEY- DELIVERY_BOY_ID-** It is the ID of the delivery boy. According to the relationship where the order is associated with both the customer as well as order, it becomes essential to connect it with the delivery_boy_ID such that we can connect the 3 tables. Also it has a relation such that we can identify how the association works.

ON UPDATE CASCADE- As it is foreign key, and if by chance the foreign id is updated then the delivery boy would be updated now as the rates_order_delivery might have been updated. So if the delivery boy is deleted then the whole rates_order_delivery row would get deleted

DATA POPULATION IN ALL TABLES

In all the tables the data has been inserted manually by our group where we used the insert into values command to put data into the table. We inserted data based on familiar people and each and every data has been handpicked by our group only. We have also inserted the data of all the new tables that we have made in the form of rating,delivery boy and offers. We have made sure the data is coherently added to the tables. We inserted the data according to the constraints and type of data as defined in the DDL while creating those individual tables. We kept in mind the length, kind of data needed and ensured the variety of the data inserted is relevant to our online retail store. After inserting the data we wrote the query according to our tables and repeated the query multiple times to fill our tables.

While inserting the data and populating data we kept in mind all the constraints and kept the data in accordance with other tables like for example if we were inserting data for some table where product id is the foreign key we ensured that the inserted data which we are populating in this table has only those product id which are present in products table to maintain the integrity of data.

Also while inserting and populating data all the assumptions taken in the project are taken into consideration along with further functionality and data is populated in such a manner that we can test diverse set of queries with the data for example we have assumed each product can have multiple sellers so in the data populated we have ensured there are multiple sellers who are selling a particular product.

Other things that have been taken into consideration are how are offers are being added and how they are being applied to the cart, with regards to the promo code and

other things. Similarly delivery boys are also adjusted according to the order they are assigned to.

Also the amount of data in tables is kept close to the real world situation with tables like product having 100 entries, customers having 200 entries and admin having 5 entries. These are the few of the queries which we have used to populate the data in various tables

insert into admin (Admin_ID, First_Name, Last_Name, Admin_Password) values (5, 'Ted', 'Cann', 'ScAdH6Z');

insert into product (Product_ID, Name, Price, Brand, Measurement, Admin_ID, Category_ID) values (4, 'Napkin - Dinner, White', 196, 'Yoveo', 10, 1, 2);

insert into product_feedback (Review_ID, Rating, Review_Body, Product_ID, Customer_ID, Review_Date) values (10, 2, 'Vision-oriented', 51, 149, '15-05-2021');

insert into orders (Order_ID, Mode, Amount, City, State, Order_Time, House_Flat_No, Pincode, Cart_ID) values (13, 'Netbanking', 357, 'Mobile', 'Alabama', '19:42', '8 Jenifer Center', '36670', 28);

insert into customer (Customer_ID, First_Name, Last_Name, Email, Mobile_No, Password) values (12, 'Robin', 'Seabrooke', 'rseabrookeb@sciencedaily.com', '8534237274', 'HN6ZTLg9j');

insert into category (Category_ID, Category_Name) values (5, 'Clothing');

BACKEND FRAMEWORK-FLASK

We have used to make the backend using flask. Flask has enabled us to pass the data into the frontend of our webpage. We have also used quite a bit of **jinja** to pass all the values such that the values get easily get replaced whatever the user writes on the webpage . Flask acts as a connecting block for our frontend and backend to merge. The data gets updated into the database only due to flask. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

We have used to code flask by

```
@app.route('/adminOrder/<admin_id>',methods=['GET', 'POST'])
def adminViewOrder(admin_id):
    my_list =[]
```

```
    cur = my_sql.connection.cursor()
    order_list = cur.execute("SELECT * FROM orders")
    if order_list>0:
        order_all = cur.fetchall()
        for order in order_all:
            temp_dict = {}
            for index in range(11):
                if(index==0):
                    temp_dict['Order_ID']=order[0]
                elif(index==1):
                    temp_dict['Mode']=order[1]
                elif(index==2):
                    temp_dict['Amount']=order[2]
                elif(index==9):
                    temp_dict['Date']=order[9]
            my_list.append(temp_dict)
    if request.method=='POST':
        return redirect('/admin/'+str(admin_id))
    return render_template('viewOrder.html',list=my_list)
```

FRONTEND FRAMEWORK-HTML/CSS AND BOOTSTRAP

Here we have made the Front end by using HTML and CSS. By using HTML(Hyper Text Markup Language) and CSS(Cascading Style Sheets) we have tried to enhance the User Experience and User Interface.

We have mainly used bootstrap where we have taken templates from. The basic background and the interface was a result of using the interface given in bootstrap . It enable us to make the background better. The rest setup of the frontend was set such that the frontend was made to our needs was done by using HTML/CSS.

In the frontend Jinja has also been used for the basic pass of input from the backend to users. The rest beautification was mainly done using HTML and CSS where we used flexbox and grid as well as div elements. We mainly used the main formatting in the bootstrap file.

THIS IS OUR BASE FILE

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.
css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoR
xT2MZw1T" crossorigin="anonymous">
    <link rel="stylesheet" href="styles.css">

    <title>
        {% block title %}

        {% endblock %}
    </title>
  </head>
  <body>
    <nav class="navbar navbar-expand-lg  navbar-dark bg-primary">
        <a class="navbar-brand" href="#">Online Store </a>
        <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
              <a class="nav-link" href={{url_for('homePage')}}>Home <span
class="sr-only">(current)</span></a>
            </li>
          </ul>
          {% block navBar %}
          {% endblock %}
        </div>
      </nav>
```

```
      {% block flash %}
      <div class="container" style="margin-top: 5px;">
        {% for message in get_flashed_messages() %}
        <div class="alert alert-warning alert-dismissible fade show"
role="alert">
          <strong>{{message}}</strong>
          <button type="button" class="close" data-dismiss="alert"
aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        {% endfor %}

      {% endblock %}

    {% block content %}

    {% endblock %}

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE
1Pi6jizo" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86d
IHNDz0W1" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js
"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xI
M+B07jRM" crossorigin="anonymous"></script>
  </body>

  <style>
    body {
    background-color: #212121;
    color: white
```

```
    }
 </style>

</html>
```

EXPLANATION OF SQL QUERIES

1.

```
SELECT customer.Customer_ID, customer.First_Name, customer.Last_Name
FROM customer
WHERE customer.Customer_ID IN (SELECT associated_with.Customer_ID
                                    FROM associated_with
                                    WHERE associated_with.Cart_ID IN (SELECT orders.Cart_ID
                                                                        FROM orders
                                                                        WHERE orders.Amount>=300
                                                                            )
                                    );
```

Explanation:- In this query we display customer_ID,first name,last name of those customers whose associated carts have any order of amount greater than equal to 300. Here we use nested queries at multiple level to get the required information.

2.

```
CREATE VIEW Rating_Table AS
SELECT AVG(product_feedback.Rating) AS Average_Rating, product_feedback.Product_ID
FROM product_feedback
GROUP BY product_feedback.Product_ID;


SELECT product.Name, product.Brand, Category.Category_Name
FROM product
JOIN category
ON product.Category_ID = category.Category_ID
WHERE product.Product_ID IN (
                                SELECT rating_table.Product_ID
                                FROM rating_table
                                WHERE rating_table.Average_Rating>=3
                                 );
```

Explanation:- In this query we display Name, Brand and category name of all those products where average rating of the product is greater than equal to 3. Here we use the concept of view(Virtual table) which is used to calculate and store the average

ratings of all the products with their product IDs then after that we join product table with category table to finally display the required data taking help of our View average_Rating.

3.

```
CREATE VIEW deliverydata AS
SELECT cart.Cart_ID, cart.Final_Amount, orders.Order_ID, orders.Delivery_Boy_ID,
delivery_boy.First_Name, delivery_boy.Last_Name, cart.Total_Count
FROM
cart, orders, delivery_boy
WHERE cart.Cart_ID = orders.Cart_ID AND orders.Delivery_Boy_ID = delivery_boy.Delivery_Boy_ID;

SELECT deliverydata.Delivery_Boy_ID, deliverydata.First_Name, deliverydata.Last_Name ,
COUNT(deliverydata.Order_ID) AS No_of_unique_orders_delivered ,SUM(deliverydata.Total_Count) AS
No_of_products_delivered, (deliverydata.Final_Amount) AS Total_Value_Of_Orders_Delivered
FROM deliverydata
GROUP BY Delivery_Boy_ID;
```

Explanation:- In this query we have used create view to basically store the data that we have implemented using the query. It would store the query data such that we can use it afterwards for some other time. Here the create view has been done for the delivery data.
The query's use is to display some details of the delivery boy. The detail shows the delivery ID, number of unique deliveries he/she has done, the total no of products he has delivered and the total worth of money he has delivered .

4.

```
SELECT sells.Seller_ID,seller.First_Name,seller.Last_Name,count(product.Category_ID) AS
No_Of_Categories_Sold, SUM(sells.No_of_Product_Sold) as
Total_Products_Sold,SUM(Product.price*sells.No_of_Product_Sold) AS Total_Sales_Done,
AVG(Product.price*sells.No_of_Product_Sold) AS Average_Sale_Per_Order ,
rating_table.Average_Rating AS Average_Rating_of_Products_Sold
FROM sells
JOIN seller
JOIN product
JOIN rating_table
WHERE sells.Seller_ID=seller.Seller_ID AND product.Product_ID=sells.Product_ID AND
rating_table.Product_ID=sells.Product_ID
GROUP BY Seller_ID
ORDER BY Seller_ID;
```

Explanation:- In this query we display Seller ID, First_Name of Seller, Last_Name of Seller, the total no of categories of a product has sold(counts the no of categories a seller has sold), the total no of products that have been sold by the seller ,the total sale they have done till now(which has been implemented by multiplying the the price of the

product with the total no of product sold), Average sale per order and average rating of the products sold. This query also uses the join query to join product, seller and the rating table and then we display the output which is grouped by Seller IDs as well as ordered by seller ID's.

5.

```
CREATE VIEW productsoldstats AS
SELECT product.Product_ID, product.Name, COUNT(associated_with.Product_ID)*product.Price AS
total_sale_done_from_this_product
FROM product
JOIN associated_with
WHERE product.Product_ID = associated_with.Product_ID
GROUP BY Product_ID
ORDER BY Product_ID;

CREATE VIEW categorystats AS
SELECT product.Category_ID, category.Category_Name, product.Product_ID,
productsoldstats.total_sale_done_from_this_product AS sale_from_this_product
FROM category
JOIN product
JOIN rating_table
JOIN productsoldstats
WHERE product.Category_ID=category.Category_ID AND
productsoldstats.Product_ID=product.Product_ID
GROUP BY Product_ID
ORDER BY Product_ID;


CREATE VIEW totalSold AS
SELECT categorystats.Category_ID,
categorystats.Category_Name,COUNT(associated_with.Product_ID) AS No_of_products_sold
FROM categorystats
JOIN associated_with
WHERE associated_with.Product_ID=categorystats.Product_ID
GROUP BY Category_ID;

SELECT categorystats.Category_ID , categorystats.Category_Name ,
SUM(categorystats.sale_from_this_product) AS sale_from_this_category , totalsold.No_of_products_sold
   FROM categorystats
   JOIN totalsold
   ON categorystats.Category_ID = totalsold.Category_ID
   GROUP BY categorystats.Category_ID;
```

Explanation:- In this query we have used create view to basically store the data that we have implemented using the query. It would store the query data such that we can use it afterwards for some other time. The create view has been created for the stats for sold products as well as the total sold products and te category stats. Here the job of our query is to display the category ID, category Name, the total sale from a particular category(by using the sum function) and the total no of products sold. We have also used the join function to join the table which shows the sale of each category and grouped on the basis of the category ID.

6.

```
SELECT customer.Customer_ID, customer.First_Name, customer.Last_Name
FROM customer
WHERE customer.Password NOT REGEXP '[0-9]'
UNION
SELECT seller.Seller_ID, seller.First_Name, seller.Last_Name
FROM seller
WHERE seller.Password NOT REGEXP '[0-9]'
UNION
SELECT delivery_boy.Delivery_Boy_ID, delivery_boy.First_Name, delivery_boy.Last_Name
FROM delivery_boy
WHERE delivery_boy.Password NOT REGEXP '[0-9]';
```

Explanation:- In this query we display Customer Id, First Name and Last Name of all customers, Seller Id , First Name and Last Name of all Sellers along with Delivery Boy Id , First Name and Last Name of all those delivery boys whose Password doesn't contain any numerical digit and hence is a week password. We here use UNION operation as we are running the similar query on two different tables but in the output we are getting them together.

7.

```
SELECT COUNT(product.Product_ID) AS Total_Additions, product.Admin_ID, admin.First_Name AS
ForeName, admin.Last_Name AS SurName
FROM product
JOIN admin
ON product.Admin_ID = admin.Admin_ID
GROUP BY product.Admin_ID
ORDER BY count(product.Product_ID) DESC;
```

Explanation:- In this query we display Admin ID, First name and Last name of admin along with the total products each admin have added into database of our store. We here use JOIN between admin and product tables and we display the output in

descending order of total additions made with admin making the highest number of product additions coming on top.

8.

```
SELECT product.Product_ID, product.Name, product.Brand, product.Price
FROM product
WHERE NOT EXISTS (
                    SELECT associated_with.Cart_ID
                    FROM associated_with
                    WHERE associated_with.Product_ID=product.Product_ID
                    AND EXISTS (
                              SELECT orders.Order_ID
                              FROM orders
                              WHERE orders.Cart_ID=associated_with.Cart_ID
                              )
                    );
```

Explanation:- In this query we display Product ID,Product name, Product's brand and price of those products which have not been sold even a single time till now. We use nested query to look over all those products which have been sold and use NOT EXISTS to get all those products which have not been sold as required by us in the query.

9.

```
SELECT cart.Offer_ID, offer.Promo_Code,count(cart.Cart_ID) AS No_Of_Times_Offer_Applied
,sum(cart.Total_Value-cart.Final_Amount) AS Combined_Discount_Availed
FROM cart
JOIN offer
WHERE cart.Offer_ID = offer.Offer_ID
GROUP BY cart.Offer_ID;
```

Explanation:- In this query we display the offer ID, the promo code, the no of times an offer has been applied(using the sum function) and the total amount of discount that has been availed by that offer code till now. Then we will use the join command to join offer and group the offer ID accordingly.

10.

```
SELECT product_feedback.Review_Body, product_feedback.Rating,
product_feedback.Review_Date,product.Name, product.Brand,product.Price
FROM product_feedback
JOIN product
ON product_feedback.Product_ID = product.Product_ID
WHERE product_feedback.Rating=1 OR product_feedback.Rating=5
GROUP BY product_feedback.Product_ID
```

```
ORDER BY MIN(product_feedback.Rating) DESC;
```

Explanation:- In this query we display Review Body, Rating, Review Date, Product Name, Product Brand and product price of all reviews which have either a 1 star rating or 5 star rating given by the user to the product. Here also we join product and product_feedback tables and the output is in descending order of rating.

TRIGGERS

It basically tells about the the action that gets performed once there is a certain change in the database. These actions may be things where the person might have to insert or delete a row and then ask the trigger to do something.

1.

```
DELIMITER $$
CREATE
	TRIGGER update_average_delivery_rating_after_insert AFTER INSERT
	ON rates_order_delivery
	FOR EACH ROW
	BEGIN
		UPDATE delivery_boy
		SET average_rating = (SELECT AVG(Rating_Given)
					FROM rates_order_delivery
					WHERE
rates_order_delivery.Delivery_Boy_ID=NEW.Delivery_Boy_ID)
		WHERE Delivery_Boy_ID = NEW.Delivery_Boy_ID;
	END$$
DELIMITER ;
```

Explanation:- In this trigger the main job is that when we give the rating to a delivery boy in rates_order_delivery then in the table of the delivery_boy the average rating is once again recalculated and updated.

2.

```
DELIMITER $$
CREATE
	TRIGGER remove_customer_corresponding_product_feedback BEFORE
DELETE
	ON customer
	FOR EACH ROW
```

```
	BEGIN
		DELETE FROM product_feedback
		WHERE OLD.Customer_ID=Customer_ID;
	END$$
DELIMITER ;
```

Explanation:- In this trigger the main work is that when we remove a particular customer then the customer's feedback in the product feedback is also removed simultaneously.

3.

```
DELIMITER $$
CREATE
	TRIGGER change_delivery_boy BEFORE UPDATE
	ON orders
	FOR EACH ROW
	BEGIN
		UPDATE rates_order_delivery
		SET rates_order_delivery.Delivery_Boy_ID = NEW.Delivery_Boy_ID
		WHERE Order_ID = NEW.Order_ID;
	END$$
DELIMITER ;
DELIMITER $$
CREATE
	TRIGGER update_average_delivery_rating_after_update AFTER UPDATE
	ON rates_order_delivery
	FOR EACH ROW
	BEGIN
		UPDATE delivery_boy
		SET average_rating = (SELECT AVG(Rating_Given)
				FROM rates_order_delivery
				WHERE
delivery_boy.Delivery_Boy_ID=rates_order_delivery.Delivery_Boy_ID);
	END$$

DELIMITER ;
```

Explanation:- Here if the user changes the delivery boy for a particular order then the new delivery boy is set for that particular order as well now for the removed delivery boy the average rating will change and recalculated accordingly.

4.

```
DELIMITER $$
CREATE
    TRIGGER delete_product_from_category BEFORE DELETE
    ON category
    FOR EACH ROW
    BEGIN
        DELETE from product
        WHERE OLD.Category_ID=product.Category_ID;
    END$$
DELIMITER ;
```

Explanation:- If a particular category is removed then all the products from that particular category will be deleted.

Appropriate INDEXES Created

1. CREATE INDEX priceindex on product (price);

This index has been created as we are frequently looking for price in our queries and summing the price for total sales so indexing will make the query faster

2. CREATE INDEX totalcount on cart(Total_count);

Again we are using total count in cart multiple times in our complex queries for calculating total amount so we index it for faster running time

3. CREATE INDEX finalvalue on cart(Final_Amount);

The data from column Final_Amount has been used multiple times for computing coupon code data and finding carts with value greater than some particular amount so it is a good practice to index this column

4. CREATE INDEX customerpassword on customer(Password);

We have written a query which checks for weak customer password and we have large number of entries in Customer table so it is a good practice to index this column.

5. CREATE INDEX deliveryboyavgrating on delivery_boy(Average_Rating);

Average_Rating of delivery boy has been used multiple times in various queries so we index it

6. CREATE INDEX ordermode on orders(Mode);

There can be various orders in the data and accessing them by payment mode would be faster through this indexing

7. CREATE INDEX productreview on product_feedback(Rating,Product_ID);

Quickly searching rating given to particular product is essential for complex queries which takes average and count over large amount of such data so indexing is beneficial here

8. CREATE INDEX customerratingdeliveryboy on rates_order_delivery(Customer_ID,Rating_Given);

Here too it makes searching for data faster to know which customer has given what rating to know average rating given by that customer for further use and this is optimized by indexing this two columns together.

QUERY OPTIMIZATION

While writing all of our queries we have made sure they are optimized for performance and are not redundant and they are not unnecessarily complicating the running times and computation.
There are some of the standard query optimization techniques we have used while writing our queries

1. Creating Index on all the columns where we are frequently using the where clause, group by and order by in queries
2. Avoiding select * from in query as it is time consuming and instead searching for only that information which we mainly need
3. Whenever we had any doubt in performance of our query we used explain format=json along with query to compare running time and indexing between two similar queries with different implementation
4. Similar attributes across different tables have used same datatype with similar size for faster performance if we use union and join operations so they utilize undexing properly
5. Avoiding aggregate/functional clause in where because it substantially increases the running time instead wherever required we have used views or pre calculated that data.
6. Replacing all OR in our queries in where by UNION as OR is very slow in performance
7. Keeping all the queries in same case for better cache performance as it is useful in query optimization
8. Ensuring that we don't use any OUTER JOIN as the default JOIN which means INNER JOIN is faster than OUTER JOIN
9. Wherever possible using views to make the actual query simpler and faster
10. Using NOT EXISTS instead of NOT IN as it is faster in one of our queries

EMBEDDED SQL QUERIES

Embedded SQL is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL. Embedded SQL statements are SQL statements written inline with the program source code, of the host language.
In our case we have connected our backend to our database using mysqldb. The command that we have used **from flask_mysqldb import MySQL** to import the mysql database into our backend.
Basically through embedded SQL queries we solve a very important aspect of our project through which when the user writes any information on our wepages through the frontend part, then that would get reflected in the database through the embedded sql queries written in the backend.
Also if anything is updated,inserted and deleted in the database that same will be updated,inserted or deleted in the database through the backend using embedded SQL queries.
Below mentioned are some embedded SQL Queries which deals with updation,deletion and insertion. We will mention four of the embedded SQL Queries.

In the Following query we have created the admin view embedded SQL Query. Here the cursor keyword is responsible for the changes happening in the database.Here mainly we update that how many times a admin has viewed the order. WE DEAL WITH UPDATION OF THE DATABASE.

```
@app.route('/adminOrder/<admin_id>',methods=['GET', 'POST'])
def adminViewOrder(admin_id):
    my_list =[]
    cur = my_sql.connection.cursor()
    order_list = cur.execute("SELECT * FROM orders")
    if order_list>0:
        order_all = cur.fetchall()
        for order in order_all:
            temp_dict = {}
            for index in range(11):
                if(index==0):
                    temp_dict['Order_ID']=order[0]
                elif(index==1):
                    temp_dict['Mode']=order[1]
                elif(index==2):
                    temp_dict['Amount']=order[2]
                elif(index==9):
                    temp_dict['Date']=order[9]
```

```
            my_list.append(temp_dict)
    if request.method=='POST':
        return redirect('/admin/'+str(admin_id))
    return render_template('viewOrder.html',list=my_list)
```

In the Following query we have created the admin adding seller embedded SQL Query. Here the cursor keyword is responsible for the changes happening in the database.Here mainly we take care of the part where the admin adds a seller into the database. Basically we are handling the INSERTION PART.

```
@app.route('/adminSeller/<admin_id>',methods=['GET', 'POST'])
def adminAdd_Seller(admin_id):
    if request.method=='POST':
        Seller_Details = request.form
        First_Name = Seller_Details['First_Name']
        Last_Name = Seller_Details['Last_Name']
        Email = Seller_Details['Email']
        Phone_Number = Seller_Details['Phone_Number']
        Password = Seller_Details['Password']
        Place_Of_Operation = Seller_Details['Place_Of_Operation']
        cur = my_sql.connection.cursor()
        cur.execute("INSERT INTO
seller(First_Name,Last_Name,Email,Phone_Number,Password,Place_Of_Operation
,Admin_ID) VALUES(%s, %s, %s, %s,
%s,%s,%s)",(First_Name,Last_Name,Email,Phone_Number,Password,Place_Of_Oper
ation,admin_id))
        flash('You have successfully added a seller !')
        my_sql.connection.commit()
        cur.close()
    return render_template('addSeller.html',admin_id=admin_id)
```

In the Following query we have created the placeorder embedded SQL Query. Here the cursor keyword is responsible for the changes happening in the database.Here mainly we take care of the part where the order is being placed by a customer. Basically we are handling the UPDATION PART here. We have used select * command to show the details for the same.

```
@app.route('/order/<user_id>',methods=['GET','POST'])
def placeOrder(user_id):
```

```
    global customer_cart_list
    global cart_id
    global total_val
    if request.method=='POST':
        OfferDetails = request.form
        P_code = OfferDetails['Promo_Code']
        cur = my_sql.connection.cursor()
        if(P_code=='Coupon_Code'):
            cur.execute("UPDATE cart SET Offer_ID = %s WHERE Cart_ID =
%s",(None,cart_id))
            my_sql.connection.commit()
            cur.close()
        else:
            offer_list = cur.execute("SELECT * FROM offer")
            if offer_list>0:
                offer_all = cur.fetchall()
                deduct = 0
                my_tup=()
            for tup in offer_all:
                if(tup[1]==P_code):
                    my_tup=tup
                    if(int(total_val)>int(tup[3])):
                        dval = (int(total_val)*float(tup[2]))/100
                        if(float(dval)>int(tup[4])):
                            deduct=int(tup[4])
                        else:
                            deduct=dval
                    break
            if(deduct==0):
                cur.execute("UPDATE cart SET Offer_ID = %s WHERE Cart_ID =
%s",(None,cart_id))
                my_sql.connection.commit()
                cur.close()
            else:
                cur.execute("UPDATE cart SET Offer_ID = %s WHERE Cart_ID =
%s",(my_tup[0],cart_id))
                total_val=total_val-deduct
                cur.execute("UPDATE cart SET Final_Amount = %s WHERE
Cart_ID = %s",(total_val,cart_id))
                my_sql.connection.commit()
```

```
                cur.close()
        for item in customer_cart_list:
            product_name = item['Name']
            cur = my_sql.connection.cursor()
            prod_list = cur.execute("SELECT * FROM product")
            if prod_list>0:
                prod_all = cur.fetchall()
                id = -1
            for tup in prod_all:
                if(tup[1]==product_name):
                    id = tup[0]
                    break
            cur.execute("INSERT INTO
associated_with(Customer_ID,Cart_ID,Product_ID) VALUES(%s, %s,
%s)",(user_id,cart_id,id))
            my_sql.connection.commit()
            cur.close()
        return redirect('/placeOrder'+'/'+str(user_id))
    return render_template('order.html',list=customer_cart_list)
```

In the Following query we have created the embedded SQL Query for the part where we show the seller what product and what quantity needs to be added and then acordingly inserting and updating it.. Here the cursor keyword is responsible for the changes happening in the database.Here mainly we take care of the part where if the seller adds the correct product and the quantity then we insert otherwise we show him the message that what he is adding is incorrect. Basically we are handling the UPDATION PART and INSERTION PART .

```
@app.route('/sell/<seller_id>',methods=['GET', 'POST'])
def sell(seller_id):
    if request.method=='POST':
        ProdDetail = request.form
        Name = ProdDetail['Name']
        Brand = ProdDetail['Brand']
        Quantity = ProdDetail['Quantity']
        cur = my_sql.connection.cursor()
        prod_list = cur.execute("SELECT * FROM product")
        if prod_list>0:
            prod_all = cur.fetchall()
```

```
            c_tup = ()
            for tup in prod_all:
                if(tup[1]==Name and tup[3]==Brand):
                    c_tup = tup
                    break
            if c_tup==() or int(Quantity)<0:
                flash('Invalid Product details or Quantity')
            else:
                cur.execute("INSERT INTO
sells(Seller_ID,Product_ID,No_of_Product_Sold) VALUES(%s, %s,
%s)",(seller_id,tup[0],Quantity))
                my_sql.connection.commit()
                cur.close()
                flash('Product added successfully ')
    return render_template('addProduct.html')
```