INFO6210 – Database Management and Database Design. **Name: Kaushal Chaudhary**
NUID: 001886763

ECOMMERCE DATABASE MANAGEMENT SYSTEM

FIRST BUY is an ecommerce online marketplace where various types of products are available for sale. Due to the rapid growth of technology, it is not possible to store data in a piece of paper or in books since the size of data is increasing day by day and the data stored in paper cannot be updated as per the requirements because of which it is important to store data in an appropriate Database Management System.

The purpose of this project is to create an effective database design that will allow the admin to maintain the records of the people who have bought products or are going to buy it from the website. The Database will also keep a track of the Order history that had taken place before and will allow the administrator to aggregate, delete or update the data with the help of MySQL.

The entities that will be present in the database are as follows:

- **Customer:** Provides the basic information of a customer such as Name, Address, Phone Number, etc.
- **Supplier_Contact:** Provides detail of a person who is associated with the supplier.
- **Payment Mode:** Provides the buying options with which a customer can buy a product.
- **Card Type:** Provides Card details.
- **Order:** Provides details about the Order that the customer has placed.
- **Manufacturer:** Provide the details of the manufacturer of a product.
- **Supply:** Provide details about the product that has been supplied by a supplier.
- **Supplier:** Provide details about the supplier.
- **Product:** Provide product details.
- **Product_Details:** Provide in-depth information of products.
- **Sub Category:** Sub category of the generalized category to which products belong.
- **Category:** Generalized category of products that belong to them.
- **Delivery Details:** Provide Delivery details of an order.
- **Delivery Person:** Provide basic information of a person who is delivering the product.
- **Shipping Details:** Provide basic information of the shipping details of an order. Sometimes the Shipping address and the billing address are different so this entity can prove to be beneficial for such cases.
- **Billing Details:** Provide details about the address on which the order has been billed.
- **Address:** This Entity contains all the addresses of the Customer, Supplier, Supplier_Contact and Delivery Boy. Each Address has a unique id which refers to a particular entity so that there won't be any problem while specifying the address of a particular entity.

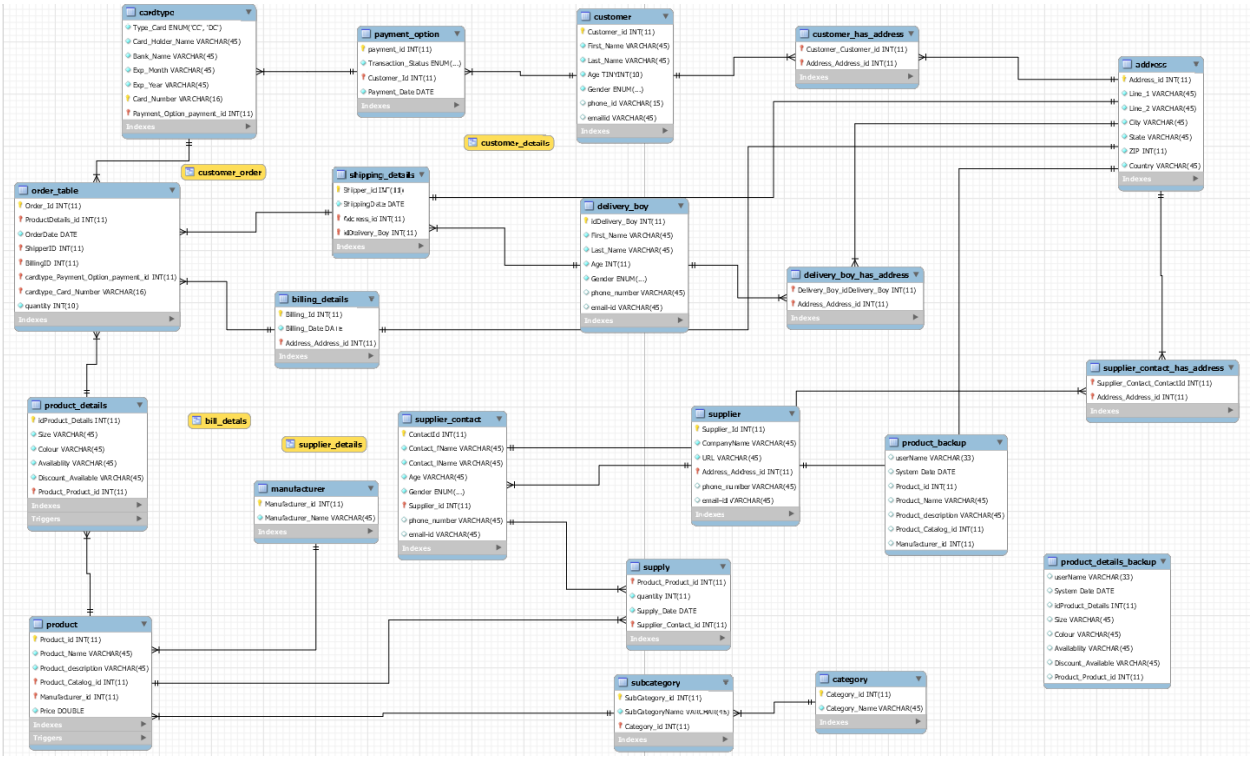
Entity Relationship:

- **One to One Relationship:** In one to one relationship, a row in the either of the entity can be related to only one row of the other entity relation.
  This kind of relationship can be found between following entities:
  - Address and Supplier
  - Address and Shipping_Details
  - Address and Billing_Details

- **One to Many Relationship:** In one to many relationship, a row in the first relation can be related to one or more rows in the second relation. But, the row in the second relation will have only single relation with the row of the first relation.
  This kind of relationship can be found between following entities:
  - Order and CardType
  - Order and Product_details
  - Supplier and Supply
  - Supply and Product
  - Product and Manufacturer
  - Category and Sub Category
  - Customer and Payment Mode

- **Many to Many Relationship:** In many to many relationships, one or more than one rows in the first relation can be related to one or more rows in the second relation. Similarly, the row in the second relation will be related to one or more rows of the first relation.
  This kind of relationship can be found between following entities:
  - Customer and Address
  - Delivery_Boy and Address
  - Supplier_Contact and Address

ER DIAGRAM:



Views: A view is a virtual table based that can be made by need of the user. We can join various table and combine them as a view.
I have created 4 views which involves joining tables to view data on a single table and they are as follows:

- **Bill_Details:** With the help of this view we can view the bill details from the order that has been placed.

```
create View bill_Detals as
select
Order_id,Manufacturer_Name,Product_Name,Product_description,Quantity,Price,Discount
_Available,((Price-(Price*Discount_Available )/100)*quantity)as Total from Order_table
inner join product_details on product_details.idProduct_Details=
Order_table.ProductDetails_id
inner join product on product_details.product_product_id= product.product_id
inner join manufacturer on Product.manufacturer_id=Manufacturer.manufacturer_id;
```

Order_id	Manufacturer_Name	Product_Name	Product_description	Quantity	Price	Discount_Available	Total
6000	Apple	I-Phone X	256 GB	2	1420	11	2527.6
6001	Apple	I-Phone X	256 GB	3	1420	18	3493.2000000000003
6002	Woodcraft	Dinning Table	For 6 people	1	230	34	151.8
6003	Google	Pixel XL	256 GB	6	1320	21	6256.799999999999
6004	Crafters	Dinning Chairs	Memory Foam	6	32	25	144

- **Supplier_Details:** This View provides the details of Supplier who has supplied the products with the help of their Supplier_contact.

```
create view supplier_details as
Select CompanyName,concat_ws('
',contact_fname,contact_lname),Supply_date,Product_Name,Product_Description,
Manufacturer_Name From manufacturer
Inner Join product on Product.manufacturer_id= Manufacturer.Manufacturer_id
inner join supply on Product.product_id = Supply.product_product_id
inner join supplier_contact on Supplier_Contact_id= contactid
inner join supplier on Supplier.supplier_id=supplier_contact.Supplier_id;
```

CompanyName	concat_ws(' ',contact_fname,contact_lname)	Supply_date	Product_Name	Product_Description	Manufacturer_Name
Hindal Co	Rock Ghag	2017-03-21	I-Phone X	64 GB	Apple
Hindal Co	Rock Ghag	2016-03-10	I-Phone X	256 GB	Apple
WoodMart	Pranav Waimbe	2017-09-11	Dinning Table	For 6 people	Woodcraft
WoodMart	Pranav Waimbe	2017-10-19	Dinning Chairs	Memory Foam	Crafters
Hindal Co	Shawn Ghatawdekar	2017-10-12	Pixel XL	256 GB	Google
Hindal Co	Shawn Ghatawdekar	2017-08-14	Pixel	128 GB	Google
Apco Inc	Aditya Pawar	2017-08-10	OLED TV	4k Pixels	Samsung

- **customer_details:** Creates a view that has details of customer who has placed the orders.

```
create view customer_details as
select
order_id,Orderdate,cardtype_card_number,First_name,Last_Name,Line_1,Line_2,City,State
,Zip,Country from order_table
inner join cardtype on order_table.cardtype_Card_Number= cardtype.Card_Number
inner join payment_option on Payment_Option_payment_id= payment_option.payment_id
inner join customer on payment_option.Customer_Id= customer.Customer_id
inner join customer_has_address on customer_has_address.Customer_Customer_id=
customer.Customer_id
inner join address on customer_has_address.Address_Address_id= address.Address_id;
```

order_id	Orderdate	cardtype_card_number	First_name	Last_Name	Line_1	Line_2	City	State	Zip	Country
6000	2017-12-11	9008768234566565	Undertaker	Deshmukh	#11 32 Cunard st	Columbus Ave	Boston	MA	2120	USA
6000	2017-12-11	9008768234566565	Undertaker	Deshmukh	#1 15 Hemenway	Fenway Park	Boston	MA	2120	USA
6002	2017-12-11	9008768298610987	Kane	Sawardekar	#11 32 Cunard st	Columbus Ave	Boston	MA	2120	USA
6004	2017-12-11	9008768298610987	Kane	Sawardekar	#11 32 Cunard st	Columbus Ave	Boston	MA	2120	USA
6001	2017-12-11	9878768296180987	Lita	Dalvi	#11 35 Bolvston st	Columbus Ave	Boston	MA	2120	USA
6003	2017-12-10	9800768234657123	Kaushal	Chaudhary	#14 Huntington Ave	Cambridge st	Boston	MA	2014	USA

- **customer_orders:** Creates a view that provides the details of the customer order that has customer details and product details.

```
create view customer_order as
select concat_ws('  ',First_Name,Last_Name) as
Customer_Name,emailid,cardtype.Card_Number,cardtype.Bank_Name,order_table.Order_I
d,product_details.Colour,product.Product_Name,product.Product_description,manufacture
r.Manufacturer_Name From manufacturer
inner join product on product.Manufacturer_id= manufacturer.Manufacturer_id
inner join product_details on product_details.Product_Product_id= product.Product_id
inner join order_table on order_table.ProductDetails_id= product_details.idProduct_Details
```

```
inner join cardtype on cardtype.Card_Number = order_table.cardtype_Card_Number
inner join payment_option on payment_option.payment_id=
cardtype.Payment_Option_payment_id
inner join customer on customer.Customer_id = payment_option.Customer_Id;
```

Customer_Name	emailid	Card_Number	Bank_Name	Order_Id	Colour	Product_Name	Product_description	Manufacturer_Name
Undertaker Deshmukh	undertaker.Deshmukh@gmail.com	9008768234566565	Bank of America	6000	Silver	I-Phone X	256 GB	Apple
Lita Dalvi	l.dalvi@yahoo.com	9878768296180987	Santander Bank	6001	Jet Black	I-Phone X	256 GB	Apple
Kane Sawardekar	kane.s@gmail.com	9008768298610987	Santander Bank	6002	Wooden	Dinning Table	For 6 people	Woodcraft
Kaushal Chaudhary	kau.c@gmail.com	9800768234657123	Bank of America	6003	Black	Pixel XL	256 GB	Google
Kane Sawardekar	kane.s@gmail.com	9008768298610987	Santander Bank	6004	Wooden	Dinning Chairs	Memory Foam	Crafters

Users:

Created a Product Manager that has given privilege on update, insert and view product_details, product, manufacturer, SubCategory and Category.

By giving this privilege, the user can only insert and update the data and cannot delete the data from the database.

```
create user 'Product_Manager' @'localhost' identified by 'pmanagerr';
grant insert on product_details to 'Product_Manager'@'localhost';
grant select on product_details to 'Product_Manager'@'localhost';
grant insert on product to 'Product_Manager'@'localhost';
grant select on product to 'Product_Manager'@'localhost';
grant update on product_details to 'Product_Manager'@'localhost';
grant update on product to 'Product_Manager'@'localhost';
grant insert on Category to 'Product_Manager'@'localhost';
grant update on Category to 'Product_Manager'@'localhost';
grant select on Category to 'Product_Manager'@'localhost';
grant insert on SubCategory to 'Product_Manager'@'localhost';
grant select on SubCategory to 'Product_Manager'@'localhost';
grant update on SubCategory to 'Product_Manager'@'localhost';
grant update on Manufacturer to 'Product_Manager'@'localhost';
grant insert on Manufacturer to 'Product_Manager'@'localhost';
grant select on Manufacturer to 'Product_Manager'@'localhost';
revoke all, grant option from 'Product_Manager'@'localhost';

select * from Product;
insert into Product values(1008,'Wooden Wardrobe','Good Quality Drawer',203,200,670);
```

✓	1	03:12:30	select * from Product LIMIT 0, 1000	7 row(s) returned
✓	2	03:14:28	insert into Product values(1008,'Wooden Wardrobe','Good Quality Drawer',203,200,670)	1 row(s) affected
✗	3	03:46:35	delete from product	Error Code: 1142. DELETE command denied to user 'Product_Manager'@'localhost' for table 'product'

We can see that delete statement cannot be used by the user since the we haven't given him the privilege to delete anything from the table.

Trigger:

I have created 2 triggers in which I will be able take a backup of the table product_details on update by any user. The old data will be stored on a separate table.
The other trigger is created to see the new values inserted by any user with the help of this, the admin will be able to know the changes made on product table and also be able to know the name of the user who made the change.

```
Delimiter %
Create trigger backup_data_product
after insert on product
for each row
begin
insert into product_backup
values(user(),sysdate(),new.Product_id,new.Product_Name,new.Product_description,new.Prod
uct_catalog_id,new.Manufacturer_id);
end;%
```

	userName	System Date	Product_id	Product_Name	Product_description	Product_Catalog_id	Manufacturer_id
	Product_Manager@localhost	2017-12-13	1008	Wooden Wardrobe	Good Quality Drawer	203	200

```
Delimiter %
Create trigger backup_data_product_details
after update on product_details
for each row
begin
insert into product_details_backup
values(user(),sysdate(),old.idProduct_Details,old.size,old.colour,old.availablity,old.Discount_Av
ailable,old.product_product_id);
end;%
```

	userName	System Date	idProduct_Details	Size	Colour	Availablity	Discount_Available	Product_Product_id
	Product_Manager@localhost	2017-12-12	2005	42*32	Blue	Sold	15	1000
	root@localhost	2017-12-13	2007	42*32	Jet Black	Yes	18	1002
	root@localhost	2017-12-13	2000	42*32	Jet Black	Sold	10	1000
	root@localhost	2017-12-13	2010	56*21	Black	Yes	21	1005
	root@localhost	2017-12-13	2012	109*130	Wooden	Yes	34	1003
	root@localhost	2017-12-13	2013	67*45	Wooden	Yes	25	1004

Stored Procedure:

- I have created a stored procedure in which if a enter an input which is the id of the table 'Product_Details', the attribute named 'availablity' will be set to sold and the old details will be reflected on the back_up table since the stored procedure will also fire the trigger.

```
Delimiter %
create procedure update_product_details(In id int)
Begin
update product_details
set availablity = 'Sold'
where product_details.idProduct_Details =id;
end;%
call update_product_details(2005);
```

when I call the stored procedure, the old value is stored in the backup table.

userName	System Date	idProduct_Details	Size	Colour	Availablity	Discount_Available	Product_Product_id
root@localhost	2017-12-13	2007	42*32	Jet Black	Yes	18	1002
root@localhost	2017-12-13	2000	42*32	Jet Black	Sold	10	1000
root@localhost	2017-12-13	2010	56*21	Black	Yes	21	1005
root@localhost	2017-12-13	2012	109*130	Wooden	Yes	34	1003
root@localhost	2017-12-13	2013	67*45	Wooden	Yes	25	1004
root@localhost	2017-12-13	2005	42*32	Blue	Yes	15	1000

- The below stored procedure can be used to find the total sales that are generated on a particular date. When we put the date into the input of the stored procedure ,the total sales we want to find gets displayed on the output.

```
Delimiter %
create procedure total_sales_per_day(in date_order date)
Begin
select Orderdate,sum((Price-(Price*Discount_Available )/100)*quantity)as Total from
Order_table
inner join product_details on product_details.idProduct_Details=
Order_table.ProductDetails_id
inner join product on product_details.product_product_id= product.product_id
inner join manufacturer on Product.manufacturer_id=Manufacturer.manufacturer_id
where order_table.orderdate = date_order
group by orderdate with rollup;
end;%

call total_sales_per_day('2017-12-10');
```

Orderdate	Total
2017-12-10	6256.799999999999