ONLINE RETAIL STORE : ShopStop

Problem Statement :
In today's time , every other person is busy in his/her errands and want to save as much time as possible and hence , we ***ShopStop*** *, an online retail store comes into existence by helping a user order items like clothes, food_items ,daily usage items etc. at a competitive price in comparison to offline retail stores .We have also introduced subscription system for frequent users to save more .*

Week 1:

StakeHolders *:*

1. *Users*
   a. *Can order multiple items under a single order .*
   b. *Can place at most 1 order any day.*
   c. *Can check the delivery status of each of the orders placed.*
   d. *Can have at most 1 active subscription to avail benefit.*
   e. *Each user will be allotted a unique UserID (U_ID)*
   f. *Should provide one phone number as a point of contact.*
2. *Employee*
   a. *Uniquely identified by a E_ID(employee ID).*
   b. *Some employees are managers who manage a category.*
   c. *Each Category has only 1 manager .*
   d. *Should have one phone number as a point of contact .*
   e. *Category Manager for a particular category can be changed at most once any day .*

3. *Vendor*
   a. *Supplies products to our retail store and are identified using V_ID(Vendor ID) .*
   b. *Supplies products of one of the categories and each Category has only 1 vendor .*
   c. *Has availability status which if true means that the vendor can provide all products of the related category as per our requirement while a false status means it doesn't have any products to supply.*
   d. *Should provide one phone number .*
4. *Delivery_Agent*
   a. *Delivers the pending order.*
   b. *Updates the status of the order once delivered.*
   c. *A single delivery agent can deliver more than one order.*
   d. *Given a pending order, any delivery agent can deliver it .*
   e. *Should provide only one phone number as a point of contact.*

<u>Week 2:</u>

<u>Entities :</u>

1. *Users*
2. *Employee*
3. *Vendor*
4. *Delivery_Agent*
5. *Product*
    a. *One product belongs to only one category and is uniquely identifiable via a product ID (P_ID).*
6. *Category*
    a. *One category can have multiple products but a product can belong to only one category .*
    b. *Each Category is managed by a single manager .*
    c. *Each category has at least one product .*
7. *Subscription*
    a. *There are 3 types of subscriptions uniquely identified by a S_ID(1,2 or 3).*
    b. *Each subscription costs 100Rs/month but has a higher benefit for longer duration subscriptions . ex: 3 months has a benefit of 3% while 12 months have a benefit of 8% on Net_Amount.*
    c. *Any user can have at most 1 active subscription.*
8. *Orders*
    a. *Can be placed by a user (1 user can place at most 1 order everyday).*
    b. *One order belongs to only one of the Users but a user can belong to multiple orders.*
    c. *Each order is uniquely identified by an Order ID (O_ID).*
    d. *Once successfully an order has been placed, it can not be modified/ canceled .*

<u>Relationships</u>

1. ***Delivery_Agent(1)-- Delivers--(m) Orders***
2. ***Users(m,partial)--has -- (1,partial)Subscription***
3. ***Users(1,partial)-- Places -- (m)Orders***
4. ***Orders(m) -- Consist_of -- (n)Product***
5. ***Category(1)--Availability -- (1)Vendor***
6. ***Category(1) –Contains – (m)Product***
7. ***Employee(1,partial) -- Manages - - (1) Category***

Week 3:

Attributes(Entities) :

1. *Users*
    - *U_ID*
    - *Name (FirstName,LastName)*
    - *Address (AptNumber,Street, State,PinCode)*
    - *ContactNo*
    - *Email*
    - *Age*
2. *Employee*
    - *E_ID*
    - *Name(First Name, Last Name)*
    - *DateofJoining*
    - *Address(AptNumber,Street,State,PinCode)*
    - *ContactNo*
    - *Email*
    - *Age*
3. *Vendor*
    - *V_ID*
    - *StoreName*
    - *ContactNo*

4. *Delivery Agent*
    - *D_ID*
    - *Name(First Name, Last Name)*
    - *DateofJoining*
    - *Address(AptNumber,Street,State,PinCode)*
    - *ContactNo*
    - *Email*
    - *Age*
5. *Product*
    - *P_ID*
    - *Name*
    - *Quantity*
    - *Price*

6. *Orders*
    - *O_ID*
    - *PurchaseDate*

        - *Net_Amount*
    7. *Category*
        - *C_Name*

    8. *Subscription*
        - *S_ID*
        - *Duration*
        - *Benefit*
        - *Price*

Note : Age has been mentioned as per the input by the user on the date of joining (in case of employee and delivery agent) / date of authorisation (in case of user) .

Attributes(Relationship) :

1. *Manages*
    - *DateofAppointment*
2. *Availability*
    - *AStatus*
3. *Has*
    - *PurchaseDate*

4. *Consists of*
    - *Quantity*

5. *Delivers*
    - *DStatus*

6. *Contains*
7. *Places*

Week 4:

Relational Schema :

*Users (**U_ID**, Name(FirstName, LastName), Address(AptNumber,Street, State,PinCode), Email,Age, ContactNo)*

*Employee (**E_ID**, Name(FirstName, LastName), Address(AptNumber,Street, State,PinCode), Email, Age, ContactNo, DateofJoining)*

*Vendor(**V_ID**, StoreName, ContactNo, C_Name)*

*Delivery Agent(**D_ID**, Name, Address(AptNumber,Street, State,PinCode), Email,Age, ContactNo, DateofJoining)*

*Order(**O_ID,** PurchaseDate, Net_Amount, U_ID)*

*Product (**P_ID**, Name, Quantity, Price, C_Name)*

*Category (**C_Name**)*

*Subscription (**S_ID**, Duration, Benefit,Price)*

(Relationship)

*Manages (**E_ID, C_Name, DateOfAppointment**)*

*Delivers (**D_ID, O_ID**, DStatus)*

*Has (**U_ID, PurchaseDate,** S_ID)*

*Consist_of (**P_ID, O_ID**, Quantity)*

*Availability (**V_ID**, AStatus)*

Constraints(including integrity Constraints)

Users
- ***U_ID INT PRIMARY KEY,***
- ***FirstName VARCHAR(20) NOT NULL,***
- ***LastName VARCHAR(20) NOT NULL,***
- ***AptNumber INT NOT NULL,***
- ***Street VARCHAR(15) ,***
- ***State VARCHAR(15) NOT NULL,***
- ***Pincode INT NOT NULL,***
- ***Email VARCHAR(50) NOT NULL,***
- ***Age INT NOT NULL check (Age > 0),***
- ***ContactNo VARCHAR(15) UNIQUE NOT NULL***

Employee
- ***E_ID INT PRIMARY KEY ,***
- ***FirstName VARCHAR(20) NOT NULL,***

- ***LastName VARCHAR(20) NOT NULL,***
- ***AptNumber INT NOT NULL,***
- ***Street VARCHAR(15) NOT NULL,***
- ***State VARCHAR(15) NOT NULL,***
- ***Pincode INT NOT NULL,***
- ***Email VARCHAR(50) NOT NULL,***
- ***Age INT NOT NULL,***
- ***ContactNo VARCHAR(15) NOT NULL UNIQUE,***
- ***DateofJoining DATE NOT NULL***

Delivery_Agent

- ***D_ID INT PRIMARY KEY,***
- ***FirstName VARCHAR(20) NOT NULL,***
- ***LastName VARCHAR(20) NOT NULL,***
- ***AptNumber INT NOT NULL,***
- ***Street VARCHAR(15) ,***
- ***State VARCHAR(15) NOT NULL,***
- ***Pincode INT NOT NULL,***
- ***Email VARCHAR(50) NOT NULL,***
- ***Age INT NOT NULL check(Age > 18),***
- ***ContactNo VARCHAR(15) UNIQUE NOT NULL,***
- ***DateofJoining DATE NOT NULL***

Vendor

- ***V_ID INT PRIMARY KEY ,***
- ***StoreName VARCHAR(9) NOT NULL,***
- ***Phone VARCHAR(50) NOT NULL,***
- ***C_Name VARCHAR(19) NOT NULL,***
- ***FOREIGN KEY (C_Name)***

Product

- ***P_ID INT PRIMARY KEY ,***
- ***Name VARCHAR(17) NOT NULL,***
- ***Quantity VARCHAR(2) NOT NULL(Quantity>=0),***
- ***Price VARCHAR(5) NOT NULL (Price>0),***
- ***C_Name VARCHAR(19) NOT NULL ,***
- ***FOREIGN KEY (C_Name)***

Orders

- ***O_ID INT PRIMARY KEY ,***
- ***PurchaseDate date Default(CURRENT_DATE()) NOT NULL,***
- ***Net_Amount int default(0) NOT NULL check(Net_Amount>=0),***
- ***U_ID INT NOT NULL ,***

- ***FOREIGN KEY (U_ID)***

Category

- ***C_Name VARCHAR(19) PRIMARY KEY***

Subscription

- ***S_ID INT PRIMARY KEY ,***
- ***Duration VARCHAR(9) NOT NULL,***
- ***Benefit INT NOT NULL (Benefit>0 and Benefit<10),***
- ***Price INT NOT NULL (Price>0)***

Manages

- ***C_Name varchar (19) NOT NULL,***
- ***DateofAppointment date,***
- ***E_ID int DEFAULT(NULL),***
- ***FOREIGN KEY (E_ID, C_NAME)***
- ***Primary Key (C_Name, E_ID,DateOfAppointment)***

Consist_of

- ***O_ID int NOT NULL,***
- ***P_ID int NOT NULL,***
- ***Quantity int check(Quantity > 0),***
- ***FOREIGN KEY (O_ID,P_ID)***
- ***Primary Key(O_ID,P_ID)***

Delivers

- ***D_ID int NOT NULL,***
- ***O_ID int NOT NULL,***
- ***DStatus varchar(3) NOT NULL Default("NO"),***
- ***Foreign Key (D_ID,O_ID)***
- ***Primary Key (D_ID, O_ID)***

Has

- ***SNO INT NOT NULL ,***
- ***PurchaseDate DATE NOT NULL,***
- ***U_ID INT NOT NULL ,***
- ***S_ID INT NOT NULL,***
- ***FOREIGN KEY (U_ID,S_ID)***
- ***Primary Key(U_ID,PurchaseDate)***

Availability

- ***V_ID int Primary Key,***
- ***AStatus varchar(3) NOT NULL DEFAULT('YES'),***
- ***Foreign Key (V_ID)***

Week 5 & 6:

Why is there no Weak Entity?

1. Order : Order can be uniquely identified by O_ID → not a weak entity .
2. Subscription : because there will not be a total participation from the subscription side because there might be a particular subscription(let say S_ID=1) that doesn't belong to any of the users but still it does exist according to our subscription model.
3. Product : Product can be uniquely identified by a P_ID .
4. Category : Already there is total participation and according to the model , each category is distinct and hence it can't be a weak entity because a weak entity does not have a stand alone primary key of its own . Also, according to the model , if a category has no products, we can add more products to it any time and hence it doesn't have dependency on Product .

Week 10

Indexes-

1. *create index Purchase_Idx on orders(PurchaseDate);*
2. *create index Location on users(State);*
3. *create index DeliveryStatus on delivers(DStatus);*
4. *create index Age_idx on users(age);*
5. *create index on_price on product(price);*

Note: A lot more indexes were used by they were already made being a part of primary key or being unique.

Triggers-

1) Trigger to update the product quantity in inventory after a user purchases some amount of the product.

```
Create Trigger update_product After Insert ON consist_of
        FOR EACH ROW
        Update Product
        Set Product.Quantity=Product.Quantity-New.Quantity
        Where Product.P_ID=New.P_ID;
```

2)Trigger to update the net amount of an order each time user adds an item to the cart

```
Create Trigger update_amount After Insert ON consist_of
        FOR EACH ROW
        Update Orders
        Set Orders.Net_Amount=Orders.Net_Amount+ ((New.Quantity)*(Select price from
Product where Product.P_ID=new.P_ID));
```

3)Trigger to update a product quantity if it falls below 10 in the inventory, if the Vendor of the category has the product available.

```
CREATE TRIGGER place_order AFTER Insert ON consist_of
        FOR EACH ROW
        Update Product
         Set Product .Quantity=Product .Quantity+25
        Where Product.Quantity<=10 and Product.P_ID=New.P_ID and product.C_Name in
                (
                select V.C_Name from vendor as V,availability as A where
                V.V_ID=A.V_ID and AStatus="YES"
                );
```

4) Trigger to check whether age is greater than (or equal) to 18 or not before insertion of a new employee and prevent insertion in case of age less than 18.

```
DELIMITER //
CREATE TRIGGER InsertPreventTrigger BEFORE INSERT ON Employee
 FOR EACH ROW
 BEGIN
 IF(new.Age <18) THEN
 SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'You can not insert record';
END IF;
END //
```

Week 11

Views-

V1.

```
CREATE VIEW User_details AS
SELECT U_ID,FirstName,LastName,Email,ContactNo
FROM users;
```

V2.

```
Create View Order_History AS
Select U_ID,Orders.O_ID,PurchaseDate,Net_Amount,DStatus,Delivery_Agent.FirstName as
DeliveryAgent,Delivery_Agent.ContactNo as ContactNo
From Orders natural join Delivers natural join Delivery_Agent
Order by U_ID;
```

V3.

```
Create View Subscription_History AS
Select * from has
Order by U_ID;
```

V4.

```
Create view still_active as
Select U_ID,S_ID
from  Has natural join Subscription
where DATE_ADD(PurchaseDate, INTERVAL SUBSTRING(Duration, 1, 2) month)>curdate();
```

V5.

```
Create view DeliveryAgentDetails as
Select D_ID, FirstName, LastName, Email, ContactNo from Delivery_Agent;
```

V6.

```
Create view PreviousDelivery as
Select D_ID, O_ID from Delivers
where DStatus = 'YES';
```

(Optimization done by using Index on DStatus field in Delivers relation)

V7.

```
 Create view PendingDelivery as
Select * from Delivers
where DStatus = 'NO';
```

(Optimization done by using Index on DStatus field in Delivers relation)

V8.

```
Create view view_manager AS
select E_ID, C_Name
from manages
where DateofAppointment in (select max(DateofAppointment) from manages group by
C_Name);
```

V9.

```
CREATE VIEW Employee_details AS
SELECT E_ID,FirstName,LastName,Email,ContactNo,DateofJoining
FROM employee;
```

Week 12

Grants-

1. Grant select,insert on project65.users to 'User';
2. Grant select on project65.User_details to 'User';
3. Grant select on project65.*Order_History* to 'User';
4. Grant select on project65.*Subscription_History* to 'User';
5. Grant select,insert on project65.delivers to 'User';
6. Grant select,insert on project65.consist_of to 'User';
7. Grant select,update on project65.product to 'User';
8. Grant select on project65.still_active to 'User';
9. Grant select, insert, update on project65.orders to 'User';
10. Grant select on project65.category to 'User';
11. Grant select on project65.DeliveryAgentDetails to 'DeliveryAgent';
12. Grant select on project65.PreviousDelivery to 'DeliveryAgent';
13. Grant select, update on project65.PendingDelivery to 'DeliveryAgent';
14. Grant select on project65.Employee_details to "Employee";
15. Grant select on project65.view_manager to "Employee";

Week 13 & 14

Queries -

NOTE : To make the queries readable , we have mentioned the creation (DDL) of views (if used) under a topic VIEWS in the current document .

1. *Mention the total number of Orders in each category.*

   select C_Name, count(*) as cnt from (select O_ID, P.P_ID, C.Quantity, P.C_Name from consist_of as C, product as P where C.P_ID = P.P_ID) as R group by C_Name;

2. *Select users of age X belonging to a Given States(S1,S2,S3…,Sn);*
   //Query is optimized by using an index on the Location field
   and Age of Users relation.

   *select * from users where State in (S1, S2, S3,...,Sn) and age = X;*

3. *Mention the most valuable customer of a particular month,year (by orders).*

   *select U_ID, max(cnt) from (select U_ID, count(*) as cnt from (select * from orders where MONTH(Orders.PurchaseDate) = 8 AND YEAR(Orders.PurchaseDate) = 2021) as R group by U_ID order by count(*) desc) as Q;*

4. *Find the revenue a given (month,year) say (M,Y) of the company.*

   select SUM(Net_Amount) as Revenue from Orders where
   MONTH(PurchaseDate) = M AND YEAR(PurchaseDate) = Y ;

5. *Show all products of a given Category(say C) within the price limit say(R)*
   //Query is optimized by using an index on the on_price field of Product relation.

   *Select **
   From Product as P
   Where P.price<=R and P.C_Name=C
   Order by C_Name,price;

6. *Display all the orders of a given Purchase date.*
   //Query is optimized by using an index on the PurchaseDate field of Orders relation.

   *select * from Orders where PurchaseDate = '20200321';*

   —---

Embedded Queries:

7. *(G)Total sales of the company in a given month (throughout all the years) category wise in descending order.*

   *Select P.C_Name as Category,sum(P.Price*C.Quantity) as TotalSales*
   from Orders as O, Consist_of as C, Product P where C.P_ID = P.P_ID and
   O.O_ID = C.O_ID and MONTH(PurchaseDate) = 3
   group by P.C_Name order by TotalSales desc;

(Note : we are taking revenue of all years into account)

8. *(G)To show which subscription plan has been bought how many times until now.*

   select S_ID, count(*) from has group by S_ID;

   (Note : we are considering all users till current date)

9. *Show all the orders placed by a given user.*

   Select O_ID,PurchaseDate,NetAmount,DStatus,DeliveryAgent,ContactNo from Order_History where U_ID= input()

(Note : we will segregate data for a single U_ID when we take the input from a User in the frontend)

10. *List of all Employees who are currently working as an employee (non-managerial position)*

    Select E_ID from Employee where E_ID Not in (Select E_ID from view_Manager);

11. *Given a user_id , display the subscription_ID of the user if it is currently active .*

    Select S_ID from still_active where U_ID = input()

NOTE :
Following functions are implemented in python using various numbers of nested and sequential SQL Embedded Commands , please check the code for the same.

- *User :*
    1. *Create Account*
    2. *Personal info*
    3. *Order History*
    4. *Order details*
    5. *Subscription History*
    6. *Place Order*

- *Employee :*
    1. *Current position*
    2. *Personal Info*

- *Vendor :*

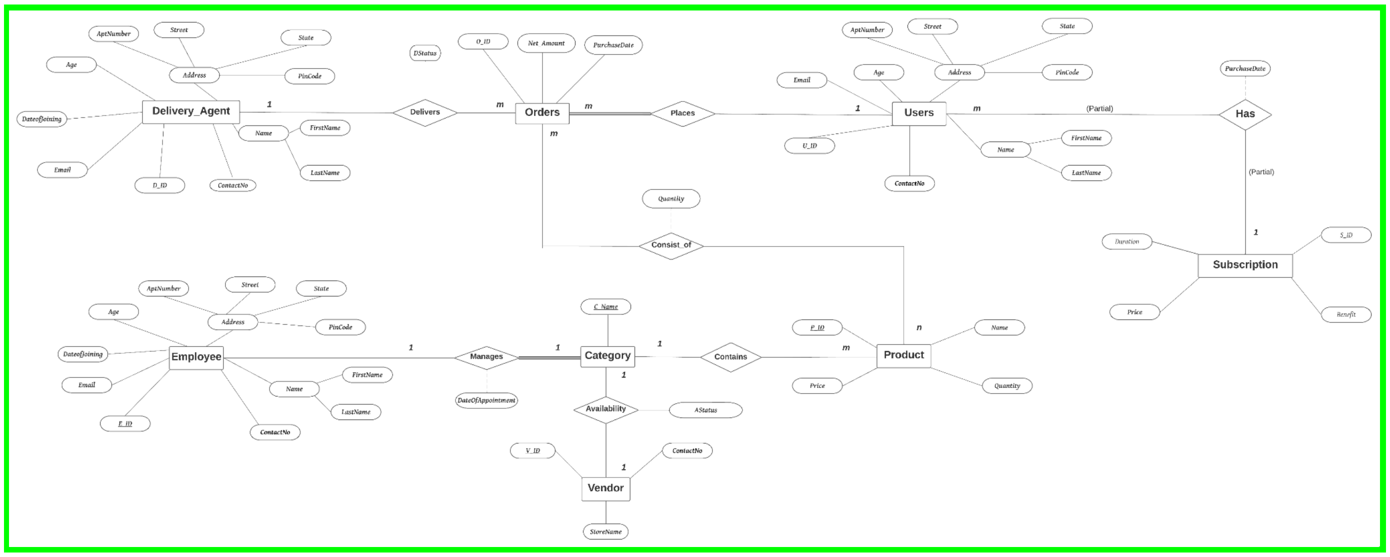
    a. *Personal Information*
    b. *Current Status of supply*
    c. *Modify Supply Status*

- *DB_Admin :*
    a. *Appoint A Category Manager*
    b. *Analyze Revenue against Categories for a given month (considering all years)*
    c. *Analyze Subscription vs Users_Enrolled*

- *Delivery_Agent :*
    a. *Personal Information*
    b. *Delivered Orders History*
    c. *Deliver a Pending Order*

CONTRIBUTION OF EACH MEMBER :
Every member of the group is involved in each of the tasks .

ER DIAGRAM:



Link(View Only) :
https://lucid.app/lucidchart/d73ad70f-99f3-41d7-9bef-a12cc06b90d8/edit?invitationId=inv_3d91672e-e58e-410e-ae2d-d41286228597

NOTE: ***Root is our DB admin other than that we have 4 other stakeholders i.e User, Employee , DeliveryAgent, Vendor***