DBMS Project Report

Members:-
Saksham Bhupal - 2020573
Sarthak Maini - 2020576
Aryan Vohra - 2020557
Harjeet Singh Yadav - 2020561

Project: Database Management System for an Online Retail Store

1) Scope of the Project:

The project follows the following description:

X is a multi branch online retail store.Each location has maximum 1 branch.All branches have all product categories.

It receives supplies from wholesale vendors whose data (Name transactions etc) is stored.After receiving the supplies they are placed as a stock in order of their date of purchase.
The list of vendors can be changed based on the business requirement.

Each customer first needs to create an account mentioning his credentials like address ph no and name etc.The customer can view the menu via the website and filter them by category .The customer purchases items by placing them in a cart .After he has decided on what they want the customer goes to the payment option in which he has option to chose his payment mode.Then the customer receives a confirmation that his order is confirmed along with the tracking information of the delivery.After receiving the delivery the customer has an option to give a feedback review regarding his purchase.
Customers can buy membership (Gold , Premium) which gives them several benefits when the decide to shop.Customers are categorized based on their purchases.There are 3 categories of customers Normal,Gold and Premium.Based on the category of the customer they get different amounts of discounts on various items.

Normal - 5%
Gold - 15%
Premium - 20%

There are multiple people available for delivery.

Each branch of the store has multiple employees.But each employee can work in a particular branch only.New employees are hired periodically and old employees are fired if they do not meet the job requirements.Each employee has a different role.The roles of any employee can't be changed ,he can only be replaced by an employee having a different role.Each employee is hired at a particular salary which can be incremented or decremented by the manager depending on the performance of the employee.

Each branch Manager will update the inventory of their respective branch.
Each branch Manager has the responsibility for placing the order from the supplier and the order history is maintained .
The manager can also hire/fire employees and change their salaries ,except his own salary .

1.junior -	*Salary:- x*
2.senior-	*Salary x+20%*

The roles are:-

1) *Product Handler - manages the products that are in stock. Takes care of restocking and updating inventory.*
2) *Customer Care:- handles all the customer reviews and complaints*
3) *Courier:- delivers products to the location specified by the customer*

Among the employees there is one manager who manages the work of all other employees.The manager can change after a specific time.

Products:- id, name , DoM , weight , Country , quantity ,price,manufacturing company
There are multiple categories of products ,each category of product is termed as a product group in the retail store.Ranging from clothing items ,groceries , cosmetics,, electronics.

The details of edible products include net weight , ingredients , date of expiry, date of manufacture , veg/non-veg , nutritional content.

Cosmetics - date of expiry , composition, direction of use
Clothing- category(Men/female/Unisex) , size , type
Electronics :- name, warranty, voltage

2) Stakeholders of the Project:

The stakeholders of our project include:-

- The owner of the retail store
- Customers shopping from the retail store
- The suppliers supplying to the retail store
- The managers of each branch
- People involved in courier delivery
- Product handlers at each branch

3) Entities involved *(Primary keys)*

1) Supplier (*Supplier ID*)
2) Product (*Product ID*):- Electronics,Eatables,Clothing,Cosmetics
3) Employee (*Employee ID*):- Manager,Courier
4) Branch (*Branch ID*)
5) Review (*Weak Entity*) - Discriminant - *Order ID*
6) Customer (*Customer ID*)
7) Cart (*Cart ID*)

4) Relationship Sets

- Supplies (ternary relationship between supplier,branch and product)
- Works_for (between employee and branch)
- Submits (between review and aggregation of cart and customer)
- Updates (between cart and customer)
- Delivers (ternary relationship between branch,courier and aggregation of cart and customer)
- Orders (between cart and customer)
- Contains(between product and cart)

- Is a (Specialization between Product and Electronics,Eatables,Clothing,Cosmetics)
  (Also a specialization between Employee and Manager,Courier)

5) Weak Entities

There is only one weak entity “Review” .Review is a weak entity because the existence of review completely depends on the order (The aggregation of cart and customer).It is the aggregation that defines the weak entity Review and thereby becomes its identifying entity with the discriminant Order ID.

Some other entities like cart which could have potentially been a weak entity ,have not been made so in order to highlight the importance of them.For example in case of cart,we wanted to highlight that the customer can update the cart again and again,or it can even turn out that he/she ends up deleting all the items in the cart.But if it were so then cart necessarily needs to be a strong entity .It is so because every weak entity necessarily needs to be in total participation with its identifying entity.Leaving the cart empty and maintaining the identity of the cart simultaneously, could only have been made possible if cart was treated as a strong entity.

6) Entities Participation Type

Reason for Total Participation

- We have used total participation in the relationship between Employee and branch in “Works_for '' relation since we felt that Each branch needs to have at least 1 employee and each employee needs to be present in 1 branch . Hence all branches and employees participate in the “Works_for'' relationship.

- The supplier - supplies - products relationship with product . Total participation from the supplier side since all suppliers need to participate in this relationship since they sell at least one product.

- The weak entity “Review” is in total participation in the relationship with the aggregation of Customer and Cart since each weak entity is identified by its corresponding strong entity and thereby all weak entities must be in total participation in the relationship with their corresponding strong entities.

7) Mapping Constraints

Many to many relationship between :-
Supplier,Product and Branch
Cart and Product
Branch and Aggregation of Customer and Cart

Many to one relationship between:-
Employee and Branch
Cart and Customer
Review and Aggregation of Customer and Cart
Courier and Aggregation of Customer and Cart

One to One relationship between:-
Customer and Cart

8)Ternary Relationship

- Delivers (ternary relationship between branch,courier and aggregation of cart and customer)
  All of the branch ,courier and order(aggregation of cart and customer) are intricately involved in the process of delivery that a delivery process can't happen if any one of these is absent,thereby it is made as a ternary relationship.

- Supplies (ternary relationship between supplier,branch and product)
  A supply can not happen without the combined involvement of supplier,branch and product,thereby supply is made as a ternary relationship.

9) Relational Schema

Supplier(Supplier_ID, Supplier_name, Supplier_city)

Sup_contact(Supplier_ID,Supplier_contact)

Product(Product_ID, Date_of_mfg, Product_name, Price, Product_type, Quantity, Net_weight,Manufacturing_company)

Branch(Branch_ID, Branch_street,Branch_city,Branch_state, Branch_pincode,Branch_name)

Increment(Branch_ID,Employee_ID,amount,date_of_increment)

Br_contact(Branch_ID,Branch_contact)

Supplies(Supply_ID,Supplier_ID,Branch_ID, Product_ID, Date_of_supply,Quantity_Supplied,Total_amount)

Electronics(Product_ID, Voltage, Warranty)

Eatables(Product_ID,Date_of_expiry, Ingredients)

Cosmetics(Product_ID,Date_of_expiry, Composition, Directions_of_use)

Clothing(Product_ID,Category, Size, Type)

Customer(Customer_ID, Customer_type, Customer_name, Customer_street, Customer_city, Customer_state, Customer_pincode, Gender, Age)

Cus_contact(Customer_ID,Customer_contact)

Cart(Cart_ID, Amount)

Contains(Cart_ID, Product_ID, Quantity)

Updates(Update_ID,Customer ID, Cart ID, Type_of_updation)

_Order(Order_ID, Customer ID, Cart_ID, Discount, Type_of_payment, Date_of_order)

Employee(Employee_ID, Employee_name, Employee_role, Employee_type, Base_salary, Employee_Street, Employee_state, Employee_city, Employee_pincode)

Emp_contact(Employee_ID,Employee_contact)

Works_for(Employee_ID, Branch_ID, Date_of_appointment)

Manager(Employee_ID, Date_of_promotion)

Delivers(Order_ID,Employee_ID,Branch_ID,Tracking_status)

Review (Order_ID,Type_of_review,Review_text,Date_and_time_of_review**)**

DDL Commands

create table supplies(Supply_ID int **Primary key**, Supplier_ID int, Branch_ID int, Product_ID int, Date_of_supply date, Quantity_supplied bigint, Total_amount double, foreign key(Branch_ID) references Branch(Branch_ID), foreign key(Product_ID) references Product(Product_ID), foreign key(Supplier_ID) references Supplier(Supplier_ID));

```
CREATE TABLE `dbms_project`.`supplies` (`Supply_ID` int Primary key, `Supplier_ID` int,
`Branch_ID` int, `Product_ID` int, `Date_of_supply` text, `Quantity_supplied` int, `Total_amount`
text, Foreign key(Proudct_ID,Brach_ID) references Branch(Branch_ID),
Product(Product_ID))
```

```
create table clothing(Product_ID int Primary key, Category text, Size int, Type text, foreign
key(Product_ID) references(Product))
```

```
create table electronics (Product_ID int Primary key, Voltage int, Warranty int, foreign
key(Product_ID) references Product(Product_ID));
```

```
create table eatables (Product_ID int, Date_of_expiry date, ingredients text, foreign
key(Product_ID) references Product(Product_ID))
```

```
create table cosmetics (Product_ID int Primary key, Date_of_expiry text, Composition text,
Directions_of_use text,  foreign key(Product_ID) references Product(Product_ID))
```

```
create table br_contact (Branch_ID int , Branch_Contact varchar(11), primary key(Branch_ID,
Branch_Contact), foreign key(Branch_ID) references Branch(Branch_ID));
```

```
CREATE TABLE branch (Branch_ID int Primary key, Branch_Street text, Branch_city text,
Branch_state text, Branch_pincode text, Branch_Name text);
```

```
CREATE TABLE product (Product_ID in Primary key, Date_of_mfg text, Product_name bigint,
Price text, Quantity double,Net_weight binary, Manufacturing_company text, Product_type text);
```

```
CREATE TABLE cart (Cart_ID int primary key, currency int );
```

```
CREATE TABLE contains (Quantity int, Cart_ID int, Product_ID int, Primary key(Cart_ID,
Product_ID), foreign key(Cart_ID) references Cart(Cart_ID), foreign key(Product_ID)
references Product(Product_ID));
```

```
CREATE TABLE cus_contact (Customer_ID int, Customer_contact varchar(10), foreign
key(Customer_ID) references Customer(Customer_ID), primary key(Customer_ID,
Customer_contact));
```

```
create table increment(Branch_ID integer,Employee_ID integer, date_of_increment date,
amount integer, foreign key(Branch_ID) references Branch(Branch_ID),foreign
```

key(Employee_ID) references Employee(Employee_ID),primary key(Branch_ID,Employee_ID,date_of_increment));

CREATE TABLE delivers (Employee_ID int, Order_ID int, Delivery status text, Branch_ID int, primary key(Employee_ID, Order_ID), **Foreign key(Order_ID,Brach_ID) references Branch(Branch_ID), Order(Order_ID)**)

create table emp_contact(Employee_contact varchar(10), Employee_ID, primary key(Employee_ID, Employee_contact), foreign key(Employee_ID) references Employee(Employee_ID));

CREATE TABLE manager (Date_of_promotion text, Employee_ID int **Primary key, foreign key(Employee_ID) references Employee(Employee_ID)**)

CREATE TABLE order (Order_ID int **Primary key**,Discount int,Date_of_order text, Customer_ID int, Cart_ID int, Type_of_payment text **Foreign key(Customer_ID,Cart_ID) references Customer(Customer_ID),Cart(Cart_ID)**)

create table review(Order_ID int, Review_text text, Date_and_time_of_review datetime, Type_of_review text, primary key(Order_ID, Date_and_time_of_time), foreign key(Order_ID) references Order(Order_ID))

CREATE TABLE updates (Update_ID int **Primary key**, Cart_ID int, Customer_ID int,Type of updation text **Foreign key(Cart_ID) references Cart(Cart_ID), Foreign key(Customer_ID) references Customer(Customer_ID)**)

CREATE TABLE works_for (Date_of_appointment date, Employee_ID int, Branch_ID int, **Primary key(Employee_ID, Branch_ID, Date_of_appointment), foreign key(Employee_ID) references Employee(Employee_ID), foreign key(Branch_ID) references Branch(Branch_ID)**)

CREATE TABLE sup_contact (Supplier_contact varchar(10), Supplier_ID int, primary key(Supplier_ID, Supplier_contact), foreign key(Supplier_ID) references Supplier(Supplier_ID));

CREATE TABLE customer (Customer_ID int **Primary key**, Customer_name text,Customer_DOB date, Customer_street text, Customer_city text,Customer_state text, Customer_pincode text,Age int, Gender text,Customer_type text)

CREATE TABLE `final`.`supplier` (`Supplier_name` text, `Supplier_ID` int **Primary key**, `Supplier_city` text)

CREATE TABLE employee (Employee_ID int **Primary key**, Employee_name text,Base_salary int, Employee_street text,Employee_state text,Employee_city text,Employee_pincode text, Employee_role text, Employee_type text,Increment int)

10) SQL Queries

Q1)
Select P.Products_name from Products as P where P.Product_type = “Clothing”

Q2)
Select E.Employee_ID from Employee as E,works_for as W where E.Employee_ID =W.Employee_ID GROUP BY W.Branch_ID HAVING E.Salary >= 20000

Q3)
select distinct E.Employee_name from Employee order by name desc where E.Base_Salary BETWEEN 1000 AND 3000

Q4)
Select distinct C.Customer_ID from Customer as C, Order as O, Contains as T where C.Customer_ID = O.Customer_ID and O.Cart_ID in (select Cart_ID from Contains, Product where Contains.Product_ID = Product.Product_ID and Product.Product_type = “Clothing”) and Customer.Customer_type = “Normal”;

Q5)
SELECT supplier_name, supplier_City form supplier Join Supplies on Supplier.Supplier_ID == Supplies.Supplier_ID where Supplies.Branch_ID in {29,23,45,23} and Product_ID = 20

Q6).
Checking the tracking status of all Electronics products
Select D.Tracking_status from Delivers, Orders where Orders.Product_ID in (Select Product.Product_ID from Product where Product.Product_type = “Electronics”)

Q7
Select Delivers.Employee_ID from Delivers Join Orders on Delivers.Order_ID == Orders.Order_ID where Customer_ID == 20.

Q8).
select count(*) from Review as R,Order1 as O where R.Type_of_Review = "feedback" and O.Order_ID = R.Order_ID and O.Customer_ID = 1;

Q9)
Select S.Supplier_ID from Supplier as S, Supplies as T where S.Supplier_ID = T.Supplier_ID and not exists (Select Supplies.Supplier_ID from Product, Supplies where Product.Product_ID = Supplies.Product_ID and Product.Product_type != “Cosmetics” and Supplies.Supplier_ID = S.Supplier_ID)

Q10)
Select count(*) from Supplier as S group by S.Supplier_City   HAVING S.Supplier_name like “%D%”

<u>10 SQL QUERIES NEW:-</u>

<u>SQL Queries new</u>

1) (Select Count(*) from customer group by Customer_Type order by count(*) desc;(Mgr)

   This Query generates how profitable the business is by indicating the total count of each type of member in descending order.

2) Select Count(*) from Contains where Quantity > 2000;

   This query generates the no of carts which have a large no of items in them

3) Select C.Customer_ID from Customer C,_order O,cart Ca,contains Co,product P where C.Customer_ID = O.Customer_ID and O.Cart_ID = Ca.Cart_ID and Ca.Cart_ID = Co.Cart_ID and Co.Product_ID = P.Product_ID and P.product_name = "felis" And C.Customer_ID in (Select Customer_ID from Customer where Customer_Type = 'gold');

   All customers who are gold and have ordered gold tea

4) Select Ca.Cart_ID from Cart as Ca,Contains as Co where Ca.Cart_ID = Co.Cart_ID and exists(select * from Product P where P.product_ID = Co.Product_ID and P.product_Type = 'Clothing') and exists (select * from Product P where P.product_ID = Co.Product_ID and P.product_Type = 'Eatables');

   All carts that have both clothing and eatables

5) Select Min(Date_Of_Appointment) from Works_for Natural Join Employee group by Employee_Role;
   Displaying the oldest employee of the company in each role of the company

6) Select E.Employee_ID from Employee E ,Increment I where E.Employee_ID = I.Employee_ID and exists (Select Employee_ID from Increment I1 where I.Employee_ID = E.Employee_ID and I.amount != I1.amount);

   Show all employees that have received a raise atleast 2 times in their history of wroking

7) Select Avg(Amount) from Cart Natural Join _Order where _Order.discount<= 20;
   The average amount paid by any user for all transactions where discount <=14% ie user is a normal type of customer

8) Select Avg(Amount) from Cart Natural Join _Order group by _Order.Discount having Discount >=20;
   The average cost of items for all users who are gold members (who have a discount >= 20%)
9) Select Employee_ID from Delivers where Delivery_status = 'not delivered';
   Display all those couriers who are currently on their way to delivering the orders

10)//Increase the salary of all employees working for more than 6 years by 10000
   update Increment
   set amount = 10000, date_of_increment = getdate();
   where Employee_ID in (select W.Employee_ID from works_for as W where 2022 - year(W.Date_of_joining) > 6);

Grants and Revokes

1) Grant UPDATE ON dbms TO MANAGER;
2) Grant SELECT ON Products To USER;
3) Grant SELECT ON Products,Orders,Cart,Deliver,review To Employee;
4) Revoke UPDATE ON MANAGER TO MANAGER
5) Show grants for Manager
6) Show grants for USER
7) Grant UPDATE ON CART TO USER;

Views

1) Create view Electronic AS SELECT E.Product_ID, Voltage, Warranty, Date_of_mfg, Product_name, Price, Product_type, Quantity, Net_weight,Manufacturing_company from Electronics E,Product P where E.Product_ID = P.Product_ID;
2) Create view Eatable AS SELECT P.Product_ID, Date_of_mfg, Product_name, Price, Product_type, Quantity, Net_weight,Manufacturing_company,Date_of_expiry, best_before, Ingredients from Eatables E,Product P where E.Product_ID = P.Product_ID;
3) Create view Cosmetic AS SELECT P.Product_ID, Date_of_mfg, Product_name, Price, Product_type, Quantity, Net_weight,Manufacturing_company,Date_of_expiry, best_before, Composition, Directions_of_use from Cosmetics E,Product P where E.Product_ID = P.Product_ID;
4) Create view Clothings AS SELECT P.Product_ID, Date_of_mfg, Product_name, Price, Product_type, Quantity, Net_weight,Manufacturing_company,Category, Size, Type from Clothing E,Product P where E.Product_ID = P.Product_ID;

<u>Indexes</u>

Creating Indexes on tables :-

1) Create UNIQUE INDEX idx_customer ON Customer (Customer_ID);
2) Create UNIQUE INDEX idx_contains ON Contains (Cart_ID,Product_ID);
3) Create INDEX idx_contain_quant ON Contains (Quantity);
4) Create UNIQUE INDEX idx_product ON Product (Product_ID);
5) Create INDEX idx_product_name ON Product (Product_name);
6) Create INDEX idx_product_type ON Product (Product_type);
7) Create UNIQUE INDEX idx_cart ON Cart (Cart_ID);
8) Create INDEX idx_cart_amt ON Cart (Amount);
9) Create UNIQUE INDEX idx_Order ON _Order (Order_ID);
10) Create INDEX idx_increment ON Increment (Amount);
11) Create UNIQUE INDEX idx_Works_for ON Works_for (Employee_ID, Branch_ID, Date_of_appointment);
12) Create UNIQUE INDEX idx_delivers ON Delivers (Order_ID);
13) Create UNIQUE INDEX idx_employee ON Employee (Employee_ID);
14) Create INDEX idx_emp_role ON Employee (Employee_role);
15) Create INDEX idx_discount ON _Order (Discount);
16) Create INDEX idx_delivers_track ON Delivers (tracking_status);
17) Create Unique INDEX idx_updates ON Updates (Update_ID);
18) Create Index idx_temp ON Delivers (Type_Of_Updation);
19) Create UNIQUE INDEX idx_incre ON Increment (Branch_ID,Employee_ID,date_of_increment);
20) Create UNIQUE INDEX idx_Manager ON Manager (Employee_ID);
21) Create UNIQUE INDEX idx_Emp_cont ON Emp_contact (Employee_ID,Employee_contact);

Triggers:

```
1)create trigger Customer_age_trigger before insert on Customer for each row set
new.Age = 2022 - year(new.Customer_DOB);
```

```
2)create trigger Employee_age_update before insert on Employee for each row set
new.Employee_age = (2022 - new.Employee_DOB);
```

```
3)Delimiter ; //
CREATE TRIGGER Discount
Before
Insert
on _order
for each row
BEGIN
IF (select Customer_type from Customer where Customer_ID = new.Customer_ID) =
'gold' THEN
SET new.Discount := 15;
ELSEIF (select Customer_type from Customer where Customer_ID =
new.Customer_ID) = 'premium' THEN
SET new.Discount = 20;
ELSE
SET new.Discount = 5;
END IF;
END; //
```