WEEK-1 PROGRESS

Team name: Divide&Conquer

Members:
Arundhati Bhattacharya (2018225)
Abhinesh Prajapati (2018005)
Pabitra Bansal (2018060)
Himanshu Garg (2018337)
Shashikant Kumar (2018096)

Application and domain of the database: **E-commerce website which deals with rentals, buying, and selling of individual, original and/or antique art pieces**.

The goal is to ease the process of trading between antique sellers and collectors through rich data that helps design better algorithms for more optimized and user-specific search results and recommendations

Stakeholders:

1) Buyers: A registered user using the platform to buy products.
2) Sellers: A registered user using the platform to sell products.
3) Rentees: A registered user using the platform to rent products from the owners.
4) Owner: A registered user from whom renters can rent the products.
5) Employees: People who handle the working of the Company such as data analysts, website maintainers, developers, creatives, etc.
6) Administrator: Director, managers of the company.
7) Logistics: Companies handling shipping, packaging, and door to door delivery.

Contribution of each member:
Each member worked equally in this project and most of the work was done with all the members sitting together while collaborating and discussing.

WEEK-2 PROGRESS

Questions each stakeholder will ask:

Registered user

a. Buyers
b. Rentees
c. Sellers
d. Owner

Buyers

1) Sorting the products on the basis of the stars of the product (ratings given)
   Have to select all the products from Products_to_sell table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then has to sort X by the stars rating before displaying it to the user.
2) Can see all his/her past sessions.
   Can search for the UserID in the Sessions table to get the session history of the user.
3) Sorting the products on the basis of the stars of the sellers
   Have to select all the products from Products_to_sell table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then, using their User_ID, we can get the stars of the sellers from the Sellers table and sort according to them.
4) Sorting the products on the basis of the price of the product.
   Have to select all the products from Products_to_sell table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then has to sort X by the Price from the products_to_Sell table before displaying it to the user.
5) Can view the products related only to a particular category/s.
   Using the Category_ID, we can traverse through all the products in the Products table having that ID and also available in Products_to_sell table.

6) Can get the details of all the past orders done from his/her ID.
   Using the buyers UserID, we can get the details of all the orders associated with his/her ID from the orders table.
7) Can get the details of the delivery executive that is delivering the product.
   From Order_ID, we can get DeliveryID, and using that, we can get DPID from delivery_details table, thus can get the contact details of the delivery executive.
8) Can read the reviews related to any product which is available on the website.
   We have to search for the PID of the product in the Reviews table to get all the reviews associated with the product.
9) Can view the recommendations on the basis of his/her past surfing through the website.
   Since B_browses_P contains all the products that a particular buyer has viewed, so we can easily create an algorithm that would recommend him/her the new products.
10) Can see the trending products at a time.
    Since B_browses_P contains all the products that have been viewed and the no. of times it has been viewed by different users; using this and the information related to all the products sold, we can decide the products trending at a time.

Rentees (the one who rents the products from the owners)

1) Sorting the products on the basis of the stars of the product (ratings given)
   Have to select all the products from Products_to_rent table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then has to sort X by the stars rating before displaying it to the user.
2) Can see all his/her past sessions.
   Can search for the UserID in the Sessions table to get the session history of the user.
3) Sorting the products on the basis of the stars of the owners
   Have to select all the products from Products_to_rent table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then, using their User_ID, we can get the stars of the owners from the Owners table.

4) Can view the products related only to a particular category/s which is available for rent.
   Using the Category_ID, we can traverse through all the products in the Products table having that ID and which is also available in Products_to_rent table.
5) Can get the details of all the past orders done from his/her ID.
   Using the rentees UserID, we can get the details of all the orders associated with his/her ID from the orders table.
6) Sorting the products on the basis of the rent price per day of the product.
   Have to select all the products from Products_to_rent table, then using the PID, has to search for that product in the Products table and save it in a table, let say it 'X'. Then has to sort X by the RentPricePerDay from the Products_to_rent table before displaying it to the user.
7) Can get the details of the delivery executive that is delivering the product.
   From Order_ID, we can get DeliveryID, and using that, we can get DPID from delivery_details table, thus can get the contact details of the delivery executive.
8) Can read the reviews related to all the products available for rent.
   We have to search for the PID of the product in the Reviews table to get all the reviews associated with the product.
9) Can view the products which are available to rent in a given time span.
   Form Products_to_rent table can see the products available to rent in a given timespan using AvailableFrom and AvailableTill attribute.
10) Can sort the products according to Minimum/Maximum rent days for which the product is available.
11) Can see the trending products at a time.
    Since B_browses_P contains all the products that have been viewed and the no. of times it has been viewed by different users; using this and the information related to all the products sold, we can decide the products trending at a time.
12) Can view the recommendations on the basis of his/her past surfing through the website.
    SInce B_browses_P contains all the products that a particular buyer has viewed, so we can easily create an algorithm that would recommend him/her the new products.

Sellers/Owners

1) Can see the details of all the products that he/she had sold/rented in the past.
2) Can see the details of all the products that he/she is selling at present on the website.
3) Can see the total number of users who have viewed his product in a given timespan.
   Using B_browses_P, we can easily see the total no. of times a particular product has been viewed in a given time span.
4) Can see all his/her past sessions.
   Can search for the UserID in the Sessions table to get the session history of the user.
5) Can search for all the reviews associated with all his/her products and sort them according to the stars.
   This would allow the seller to analyse the problems that buyers/renters are having related to that product and would help them in improving it.
   Using the UserID of the seller/owner of the product, we can get all his/her products from the Products table, thus using the PID of these products, we can easily get the reviews associated with these products from the Review table.
6) Can see the trending products at a time.
   This would allow the seller/owner to provide the products similar to the one trending online on the website.
   Since B_browses_P contains all the products that have been viewed and the no. of times it has been viewed by different users; using this and the information related to all the products sold, we can decide the products trending at a time.

Logistics:

1) Assign a Delivery Executive depending on the country and city of Delivery
   Can be assigned by checking the Address of the user obtained via the UserID from the Orders table
2) View the most well-rated Delivery Executive

Can be viewed by selecting the tuple with max number of Stars and belonging to a particular Company from DeliveryExec table

3) List of products that were delivered on a particular day.
   Find all the products information from OrderDetails, DeliveryDetails and DeliveryCompletionDetails tables, where the DeliveryDate is today.
4) List of products that were delivered today in the "XYZ" city.
   We have two tables, Order and User. Find all the UserID who live in "XYZ" city then check whether UserID is present in Orders table or not if it is present, then check the product id is present in DeliveryCompletionDetails which only stores delivery details for completed deliveries; if the product is delivered, then select it.
5) List of non-delivered products.
   Iterate through DeliveryDetails table and check the Completed attribute. If the status is "False," then select the details corresponding to OrderID from the Order table

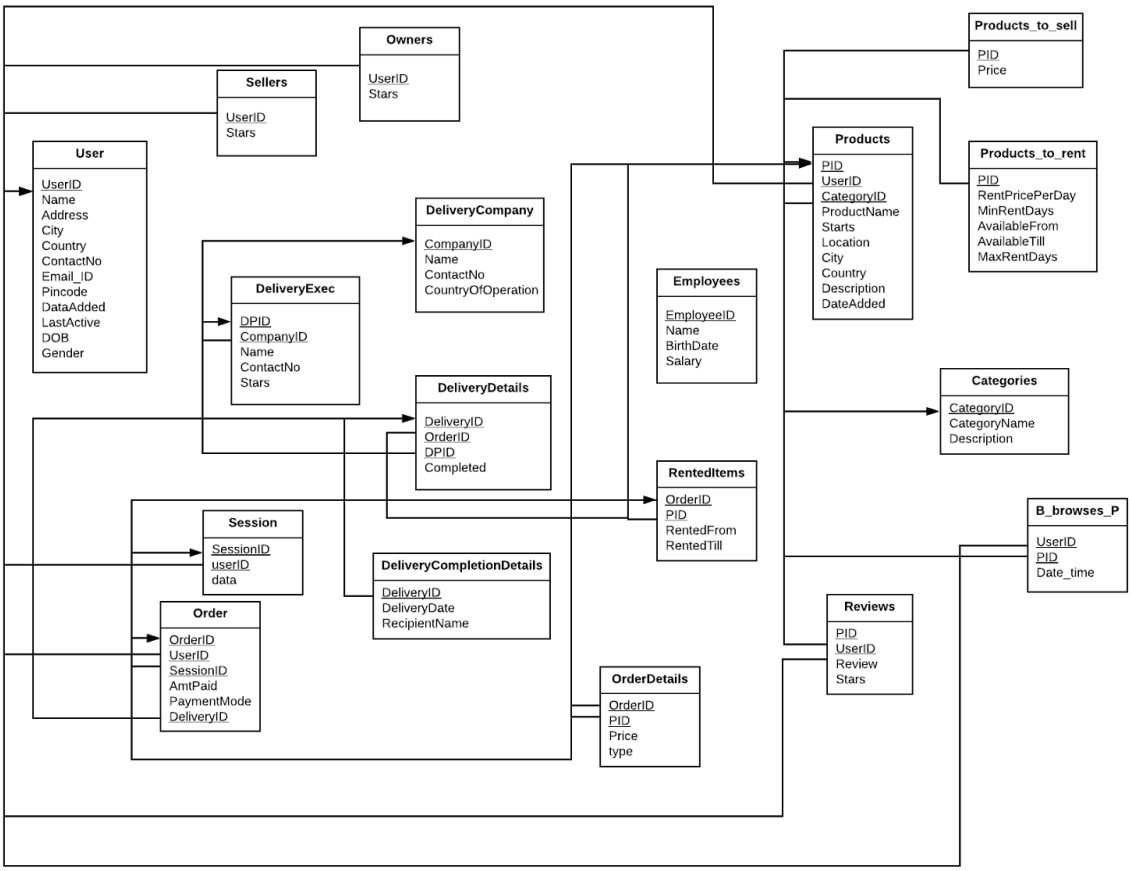
Employees and Administrators:

1) Show count of all new users who registered in 2020
   We have to use the table “Users” and use the “count” functions to count all the users who have DateAdded equal to 2020.
2) Collect information on the most viewed product by a user and information about the user like age, country, to design algorithms for better recommendations
   Fetch required info from B_Browses_P, User and Product tables
3) View all Registered Users who do more than one kind of trading (Buying, Selling, Renting, Giving out for rent)
   Select all UserID that is common in more than one of the tables -- Sellers, Owners, Renters
4) To analyse the trend in the market.
   Can be done using the orders history and the B_browses_P table.
5) Accessing all the details of a particular Order no.
   We have a table named Order which contains all the details of a particular transaction as the employee has access to this table he can apply query using the OrderID attribute.
6) List all the products sold or rented on a particular date
   The combination of three tables is needed naming “Products” which gives the PID which will be compared with the PID of the table “BillDetails” which will

give us OrderDetailID which we will use it to compare the OrderDetailID of table “Order” which provides us with the date on which the product was sold which can be used to compare with the required date.

7) Get all the owners who have at least a 4-star rating.
   We use the “User” table to get the details of the user and in the “where” clause we use the “IN” keyword with the “Owners” table to ensure that the user is an Owner and also considering Stars >= 4.
8) The number of products in all categories.
   We use the “order by” clause while with CategoryID in the products table so that we can group all the products with the same CategoryID and after using “count” we can count the number of products each belonging to a particular category.

WEEK-3 PROGRESS



Tables:

1. User(UserID char(6) **primary key**,
   Name varchar(20) not null,
   Address varchar(50) not null,
   City varchar(20) not null,
   Country varchar(10) not null,
   ContactNo varchar(15) not null,
   Email_ID varchar(20) not null **with constraint (**Email_ID like '%_@__%.__%'**)**,
   Pincode char(6) not null,
   DateAdded datetime not null,

LastActive datetime not null,
DOB datetime not null **with constraint**(DOB < 01-01-2007 (Age>13 yrs)),
Gender varchar(25) not null **with constraint**(Gender = 'Male' or Gender = 'Female' or Gender = 'Don't want to disclose'))

2. Owners(UserID char(6) **foreign key** references User(UserID),
   Stars real not null **with constraint**(0<= Stars <= 5))
3. DeliveryCompany(CompanyID char(6) **primary key**,
   Name varchar(20) not null,
   ContactNo varchar(15) not null,
   CountryOfOperation varchar(20) not null)
4. DeliveryExec (DPID char(6) primary key,
   CompanyID char(6) **foreign key** references DeliveryCompany(CompanyID),
   Name varchar(20) not null,
   ContactNo varchar(15) not null,
   Stars real not null **with constraint**(0<= Stars <= 5))
5. DeliveryDetails(DeliveryID char(6) **primary key**,
   OrderID char(6) **foreign key** references Order(OrderID),
   DPID char(6) **foreign key** references DeliveryExec(DPID),
   Completed tinyint not null)
6. DeliveryCompletionDetails(DeliveryID char(6) **foreign key** references DeliveryDetails(DeliveryID) and **with constraint**(DeliveryDetails.Completed = True),
   DeliveryDate datetime not null,
   RecipientName varchar(20) not null);
7. Categories(CategoryID char(6) **primary key**,
   CategoryName varchar(20) not null,
   Description text not null)
8. Products(PID char(6) **primary key**,
   UserID char(6) **foreign key** references User(UserID),
   CategoryID char(6) **foreign key** references Categories(CategoryID),
   ProductName varchar(20) not null,
   Stars real not null **with constraint**(0<= Stars <= 5),
   Location varchar(20) not null,
   City varchar(20) not null,
   Country varchar(20) not null,
   Description text not null,
   DateAdded datetime not null)

9. Products_to_sell(PID char(6) **foreign key** references Products(PID),
   Price real not null)
10. Products_to_rent(PID char(6) **foreign key** references Products(PID),
   RentPricePerDay real not null,
   MinRentDays int not null **with constraint**
   (MinRentDays<=(AvailableTill - AvailableFrom)),
   AvailableFrom datetime not null,
   AvailableTill datetime not null,
   MaxRentDays int not null **with**
   constraint(MinRentDays<=(AvailableTill - AvailableFrom))

11. Order (OrderID char(6) **primary key,**
   UserID char(6) **foreign key** references User(UserID),
   SessionID **foreign key** references Session(SessionID),
   AmtPaid real not null,
   PaymentMode varchar(20) not null,
   DeliveryID char(6) **foreign key** references DeliveryDetails(DeliveryID))
12. OrderDetails(OrderID char(6) **foreign key** references Order(OrderID),
   PID char(6) **foreign key** references Products(PID),
   Price real not null,
   type varchar(6) not null **with constraint** (type = 'rented' or type
   = 'sold'),
   Primary key(OrderID,PID))
13. Employees (EmployeeID int **primary key**,
   Name varchar(20) not null,
   BirthDate date not null,
   Salary real not null **with constraint**(Salary > 0))
14. RentedItems (OrderID char(6) **foreign key** references Order(OrderID),
   PID char(6) **foreign key** references Products(PID),
   RentedFrom date not null,
   RentedTill date not null **with constraint**(RentedTill >=
   RentedFrom)
   Primary key(OrderID,PID))
15. Session (SessionID timestamp **primary key**,
   userID varchar(6) **foreign key** references User(UserID),
   data text not null)
16. B_browses_P (SessionID timestamp **foreign key** references Session,
   UserID char(6) **foreign key** references User(UserID),
   PID char(6) **foreign key** references Products(PID),

Date_time datetime not null),
Primary key(Session, PID, UserID))

WEEK-4 PROGRESS

The tables were populated with ~100 rows of data and the database is hosted on Azure.

Create table queries:

```
create table User
( UserID varchar(6),
        Name varchar(20) not null,
        Address varchar(50) not null,
        City varchar(10) not null,
        Country varchar(10) not null,
        ContactNo varchar(15) not null,
        Email_ID varchar(20) not null check(Email_ID like '%_@__%._%'),
        Pincode varchar(12) not null,
        DateAdded datetime not null,
        LastActive datetime not null,
        DOB date not null check(DOB < '2007-01-01'),
        Gender varchar(25) not null check(Gender in ('Male','Female','Do not wish to
disclose')),
        primary key (UserID)
);

create table Sessions
(       SessionID datetime,
        UserID varchar(6),
        primary key(SessionID,UserID),
        Data text,
        foreign key(UserID) references User(UserID) on delete cascade on update
cascade
);

create table Categories
(       CategoryID varchar(6),
        CategoryName varchar(6) not null,
```

```
        Description text,
        primary key(CategoryID)
        );

create table Employees (EmployeeID varchar(6) not null,
Name varchar(20) not null,
BirthDate date not null check (BirthDate < '2007-01-01'),
Salary real not null check (Salary > 0),
primary key(EmployeeID));

create table Products(PID varchar(6),
UserID varchar(6) not null,
CategoryID varchar(6) not null,
ProductName varchar(20) not null,
Stars real not null
check(Stars>=1 and Stars <= 5),
Location varchar(20) not null,
City varchar(20) not null,
Country varchar(20) not null check(Country in (select Country from
DeliveryCompany)),
Description text,
DateAdded datetime not null,
primary key (PID),
foreign key (UserID) references Owners(UserID) on delete cascade on update
cascade,
foreign key (CategoryID)references Categories(CategoryID) on delete cascade on
update cascade);

create table Products_to_sell(PID varchar(6) unique,
Price real not null,
foreign key (PID)references Products(PID) on delete cascade on update cascade);

create table Products_to_rent(PID varchar(6),
RentPricePerDay real not null,
MinRentDays int not null
check(MinRentDays<=datediff(AvailableTill,AvailableFrom)),
MaxRentDays int not null
check(MaxRentDays<=datediff(AvailableTill,AvailableFrom)),
AvailableFrom date not null,
```

```
AvailableTill date not null,
primary key(PID),
foreign key (PID) references Products(PID) on delete cascade on update cascade);

create table DeliveryCompany(CompanyID varchar(6),
Name varchar(20) not null,
ContactNo varchar(15) not null,
CountryOfOperation varchar(20) not null,
primary key(CompanyID));

create table DeliveryExec (DPID varchar(6),
CompanyID varchar(6),
Name varchar(20) not null,
ContactNo varchar(15) not null,
Stars real not null
check(Stars>=1 and Stars <= 5),
primary key (DPID),
foreign key (CompanyID) references DeliveryCompany(CompanyID) on delete
cascade on update cascade);

create table DeliveryDetails( DeliveryID varchar(6),
DPID varchar(6),
Completed tinyint not null,
primary key (DeliveryID),
foreign key (DPID) references DeliveryExec(DPID) on delete cascade on update
cascade);

create table DeliveryCompletionDetails
(       DeliveryID varchar(6) check(DeliveryID in (select DeliveryID from
DeliveryDetails where Completed = 1)),
        DeliveryDate datetime not null,
        RecipientName varchar(20) not null,
        foreign key(DeliveryID) references DeliveryDetails(DeliveryID) on delete
cascade on update cascade
        );

create table Orders(OrderID varchar(6) not null,
UserID varchar(6) not null,
SessionID datetime ,
```

```
AmtPaid real not null,
PaymentMode varchar(20) not null,
DeliveryID varchar(6) not null unique,
primary key(OrderID),
foreign key (UserID) references User(UserID) on delete cascade on update cascade,
foreign key (SessionID) references Sessions(SessionID) on delete cascade on update
cascade,
foreign key (DeliveryID) references DeliveryDetails(DeliveryID) on delete cascade on
update cascade
);

create table B_browses_P
(       SessionID datetime,
        UserID varchar(6),
        PID varchar(6),
        Date_time datetime not null,
        foreign key(SessionID) references Sessions(SessionID) on delete cascade on
update cascade,
        foreign key(UserID) references User(UserID) on delete cascade on update
cascade,
        foreign key(PID) references Products(PID) on delete cascade on update
cascade
        );

create table OrderDetails(OrderID varchar(6) ,
PID varchar(6) ,
Price real not null,
type varchar(6) not null check (type = 'rented' or type = 'sold'),
primary key(OrderID,PID),
foreign key (OrderID) references Orders(OrderID) on delete cascade on update
cascade,
foreign key (PID) references Products(PID) on delete cascade on update cascade);

create table Owners(UserID varchar(6),
Stars real not null
check(Stars>=1 and Stars<= 5),
foreign key (UserID) references User(UserID) on delete cascade on update cascade);
```

Indexes:

We know that the primary key of a table is also an index, so we have only mentioned the indexes explicitly created by us.

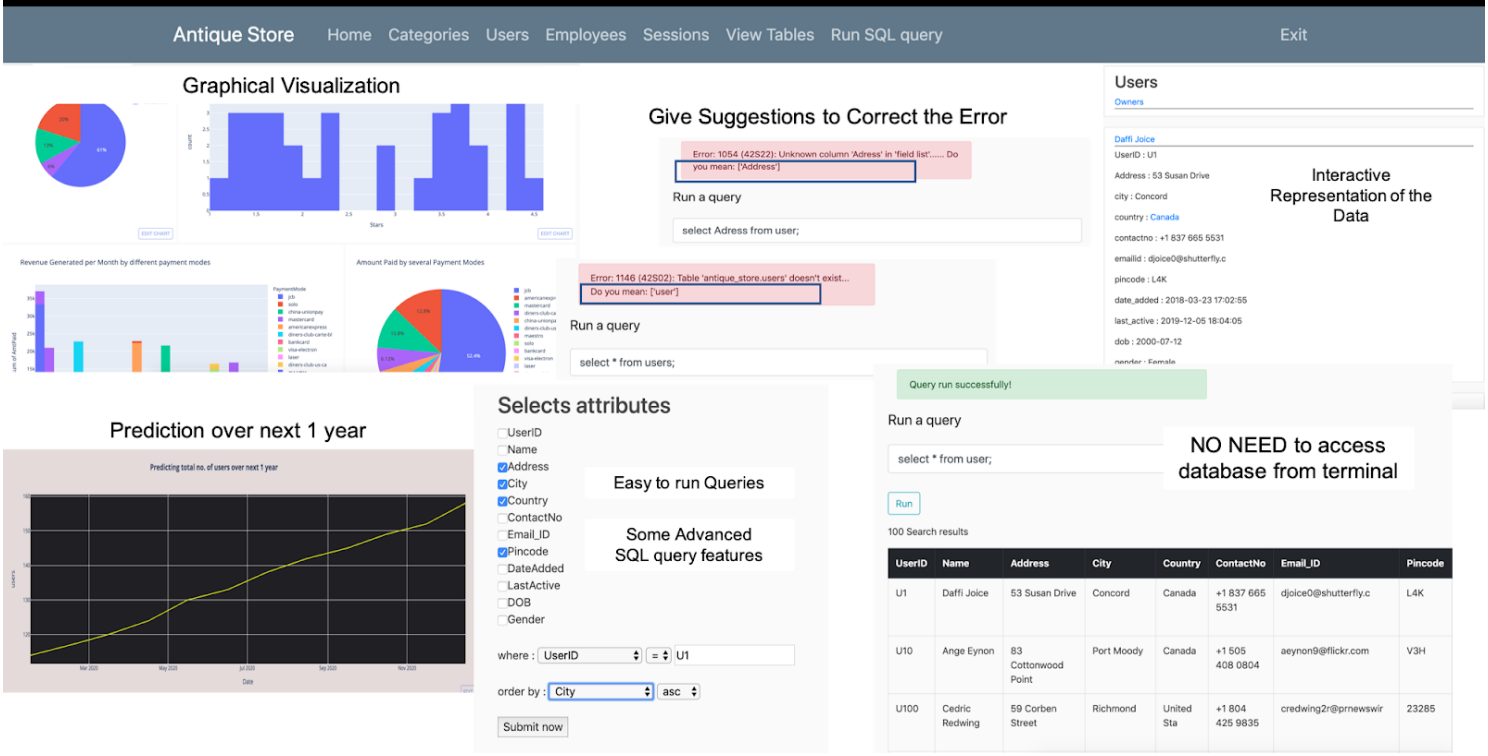
- Products: UserID, CategoryID
- DeliveryExec: CompanyID
- DeliveryDetails: DPID
- DeliveryCompletionDetails: DeliveryID
- Orders: DeliveryID, UserID, SessionID

Bonus Features :

Dashboard present at: https://antique-store-dashboard.herokuapp.com/

1. Dashboard to run the queries easily and efficiently.
2. Saved previously run queries in cache memory of the browser and recommend them while typing new queries.
3. 'View Table' feature allows the user to see all the table names and their attributes present in the database; allow them to select by just clicking on them.
4. Also it allows users to run advanced sql features 'sorting' wrt a particular column using GUI.
5. Other advanced sql features like "=, >, <" are also present to view the desired data only.
6. Graphical data visualization allows the user to analyse the data quickly and easily, and allows the user to take business oriented decisions easily.
7. Predicting Total number of Users over the next one year using Regression model : allows users to set achievable goals for the company growth.
8. Database in 3NF.

Bonus Features



Relational queries:

1. **see the description of products having stars>=3**
   $\pi_{Description}((\sigma_{Categories.CategoryID=Products.CategoryID} X \pi_{CategoryID}(\sigma_{Stars>=3}(Products)))$

2. **see the past sessions of all the users living in delhi.**
   $\pi_{SessionID}(\sigma Session.UserID=User.UserID(Session X \pi(UserID(\sigma_{Address\ like\ \%Delhi\%}(User))))$

3. **List of all non-delivered products.**
   $\pi_{OrderID}\ (\sigma_{Completed\ =\ 0}(DeliveryDetails))$

4. **List all the product names whose category is Silver.**
   $\pi_{ProductName}(\sigma_{CategoryName='Silver'}(Products \bowtie_{Products.categoryid\ =\ Categories.categoryid} (Categories)))$

5. **List all the product names whose stars are greater than four.**
   $\pi_{ProductName}(\sigma_{Product.Stars>4}(Products))$

6. **List all the product names whose price is greater than thousand.**
   $\pi_{ProductName}(\sigma_{Products.PID=Products\_to\_sell.PID \wedge Products\_to\_sell.Price>1000} (Products \times Products\_to\_sell))$

7. **To see the sessions history of a user using its UserID (let UserID = U54)**
   $\pi_{SessionID,data}(\sigma_{userID='U54'}(Session))$

8. **List all the past orders ordered by a buyer (let UserID of the buyer = U94)**
   $\pi_{O.OrderID}(\sigma_{O.UserID='U94' \wedge O.OrderID=DD.OrderID}(\rho_{O}(Order) \times \rho_{DD}(DeliveryDetails)))$

9. **List all the non-delivered orders of a particular buyer (let UserID = U52)**
   $\pi_{O.OrderID}(\sigma_{O.UserID='U52' \wedge O.OrderID=DD.OrderID \wedge Completed=0}(\rho_{O}(Order) \times \rho_{DD}(DeliveryDetails)))$

10. **List all the products registered by a seller on the website to sell (let UserID of the seller = U21)**
    $\pi_{Products.PID}(\sigma_{UserID='U21' \wedge Products.PID=Products\_to\_sell.PID}(Products \times Products\_to\_sell))$

Embedded SQL Queries:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQL
{
   public static void main(String[] args)
   {
       Connection conn = null;
       try
       {
           String url
="jdbc:mysql://dbms-proj-server.mysql.database.azure.com:3306/antique_store?useSSL=tru
e&requireSSL=false";
```

```
            conn = DriverManager.getConnection(url, "dbmsproj@dbms-proj-server",
"Dbmspr0j3ct75");
            System.out.println("Got it!");
//           Query 1
            Statement stmt = null;
            String query = "select paymentmode,sum(amtpaid) as totalamt from orders
group by paymentmode";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("Sum of amount paid by different modes");
                System.out.println(query);
                while (rs.next())
                {
                    String mode = rs.getString(1);
                    Float totalamt = rs.getFloat(2);
                    System.out.printf("%-25s %.2f\n",mode,totalamt);
                }
                System.out.println("\n\n");
            }
            catch (SQLException e )
            {
                throw new Error("Problem", e);
            }
            finally
            {
                if (stmt != null) { stmt.close(); }
            }
//           Query 2
            query = "select format(std(productcount),2)  std_deviation from (select
categoryid,count(*) productcount from products group by categoryid) t";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("Find the standard deviation of the number of
products in each category.");
                System.out.println(query);
```

```
                while (rs.next())
                {
                    String dpid = rs.getString(1);
                    System.out.printf("%s\n",dpid);
                }
                System.out.println("\n\n");
            }
            catch (SQLException e )
            {
                throw new Error("Problem", e);
            }
            finally
            {
                if (stmt != null) { stmt.close(); }
            }

//           Query 3
            query = "select format(variance(price),2) from products_to_sell;";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("Find the variance of the price of products from
products_to_sell.");
                System.out.println(query);
                while (rs.next())
                {
                    String productName= rs.getString(1);
                    System.out.printf("%s\n",productName);
                }
                System.out.println("\n\n");
            }
            catch (SQLException e )
            {
                throw new Error("Problem", e);
            }
            finally
            {
                if (stmt != null) { stmt.close(); }
```

```
            }

//          Query 4
            query = "select p.productname from products p where p.stars>4";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("List all the product names whose stars are greater
than four.");
                System.out.println(query);
                while (rs.next())
                {
                    String productName= rs.getString(1);
                    System.out.printf("%s\n",productName);
                }
                System.out.println("\n\n");
            }
            catch (SQLException e )
            {
                throw new Error("Problem", e);
            }
            finally
            {
                if (stmt != null) { stmt.close(); }
            }

//          Query 5
            query = "select productname from products p , products_to_sell pts where
p.pid=pts.pid and pts.price>9000";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("List all the product names whose price is greater
than nine thousand.");
                System.out.println(query);
                while (rs.next())
                {
```

```
                String productName= rs.getString(1);
                System.out.printf("%s\n",productName);
            }
            System.out.println("\n\n");
        }
        catch (SQLException e )
        {
            throw new Error("Problem", e);
        }
        finally
        {
            if (stmt != null) { stmt.close(); }
        }

//          Query 6
        query = "select avg(amtpaid) as avgamt from orders as o where o.userid =
'U94'";
        try
        {
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("Average amount spent per transaction by a
customer(let UserID of the buyer = 'U94')");
            System.out.println(query);
            while (rs.next())
            {
                Float avgAmt = rs.getFloat(1);
                System.out.printf("%.2f\n",avgAmt);
            }
            System.out.println("\n\n");
        }
        catch (SQLException e )
        {
            throw new Error("Problem", e);
        }
        finally
        {
            if (stmt != null) { stmt.close(); }
        }
```

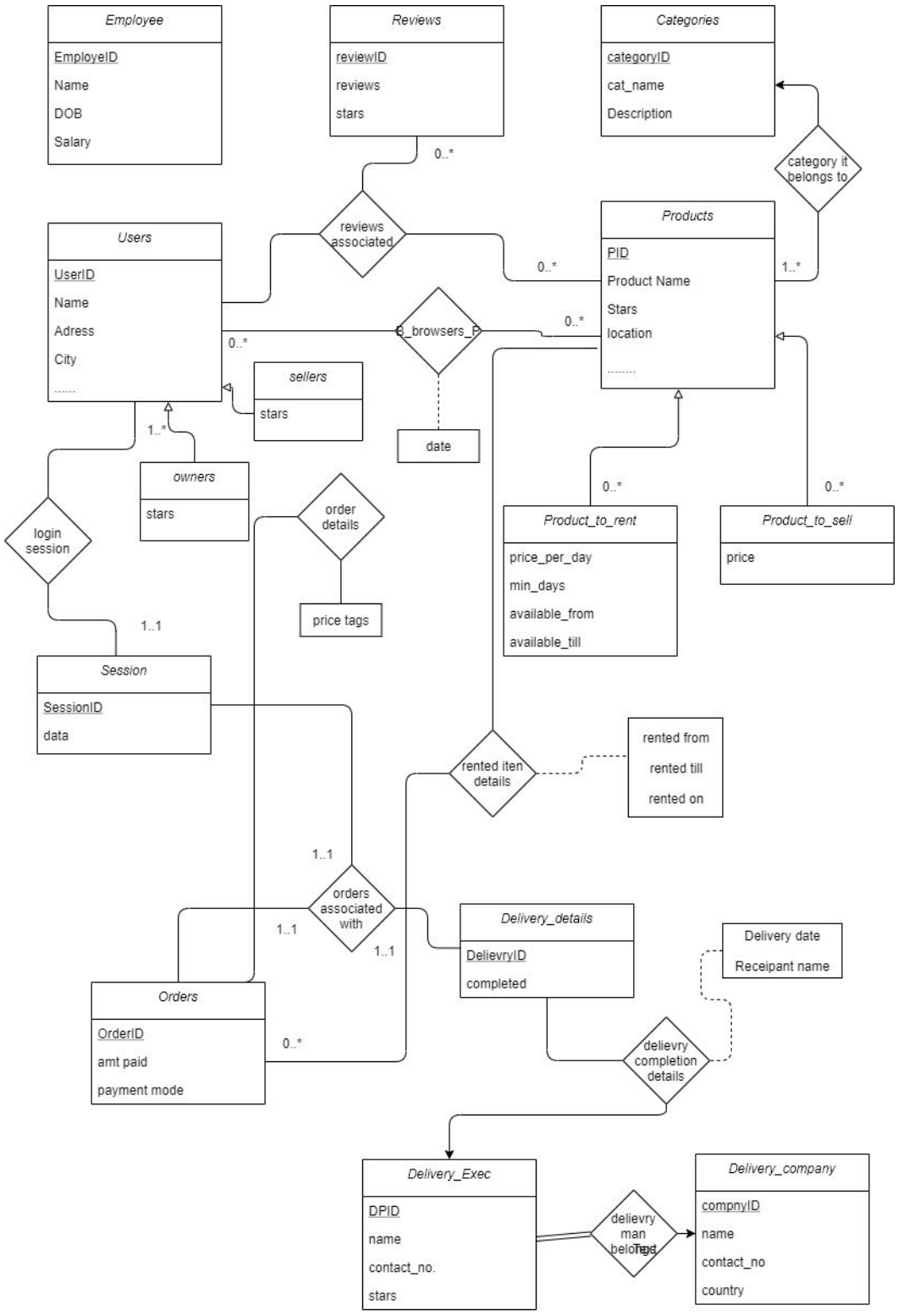
```
//          Query 7
          query = "select orderid from orders o, deliverydetails dd where o.userid =
'U52' and o.deliveryid = dd.deliveryid and dd.completed = 0";
          try
          {
              stmt = conn.createStatement();
              ResultSet rs = stmt.executeQuery(query);
              System.out.println("List all the non-delivered orders of a particular
buyer (let UserID = 'U52')");
              System.out.println(query);
              while (rs.next())
              {
                  String orderID = rs.getString(1);
                  System.out.printf("%s\n",orderID);
              }
              System.out.println("\n\n");
          }
          catch (SQLException e )
          {
              throw new Error("Problem", e);
          }
          finally
          {
              if (stmt != null) { stmt.close(); }
          }

//          Query 8
          query = "select pts.pid,pts.price,p.categoryid from products p ,
products_to_sell pts where p.userid = 'U21' and p.pid = pts.pid";
          try
          {
              stmt = conn.createStatement();
              ResultSet rs = stmt.executeQuery(query);
              System.out.println("List all the products registered by a seller on the
website to sell (let UserID of the seller = U21)");
              System.out.println(query);
              while (rs.next())
              {
```

```
                String pid = rs.getString(1);
                Float price = rs.getFloat(2);
                String categoryID = rs.getString(3);
                System.out.printf("%-4s %.2f %-4s\n",pid,price,categoryID);
            }
            System.out.println("\n\n");
        }
        catch (SQLException e )
        {
            throw new Error("Problem", e);
        }
        finally
        {
            if (stmt != null) { stmt.close(); }
        }

        //Query 9
        query = "select t1.dpid from deliveryexec t1 where t1.stars = (select
max(t2.stars) from deliveryexec t2)";
        try
        {
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("View the most well-rated Delivery Executive");
            System.out.println(query);
            while (rs.next())
            {
                String dpid = rs.getString(1);
                System.out.printf("%s\n",dpid);
            }
            System.out.println("\n\n");
        }
        catch (SQLException e )
        {
            throw new Error("Problem", e);
        }
        finally
        {
            if (stmt != null) { stmt.close(); }
```

```
            }

//          Query 10
            query = "select p.productname from products p,categories c where
p.categoryid = c.categoryid and c.categoryname = 'Silver'";
            try
            {
                stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);
                System.out.println("List all the product names whose category is
Silver.");
                System.out.println(query);
                while (rs.next())
                {
                    String productName= rs.getString(1);
                    System.out.printf("%s\n",productName);
                }
                System.out.println("\n\n");
            }
            catch (SQLException e )
            {
                throw new Error("Problem", e);
            }
            finally
            {
                if (stmt != null) { stmt.close(); }
            }



        }
        catch (SQLException e)
        {
            throw new Error("Problem", e);
        }
        finally
        {
            try
            {
                if (conn != null)
```

```
                    conn.close();
            }
            catch (SQLException ex)
            {
                System.out.println(ex.getMessage());
            }
        }
    }
}
```

E-R Diagram


Employee
EmployeID
Name
DOB
Salary
Reviews
reviewID
reviews
stars
Categories
categoryID
cat_name
Description
category it belongs to
0..*
reviews associated
Users
UserID
Name
Adress
City
.....
0..*
Products
PID
Product Name
Stars
location
.....
1..*
0..*
_browsers_P
0..*
sellers
stars
date
1..*
owners
stars
login session
order details
price tags
1..1
Session
SessionID
data
0..*
Product_to_rent
price_per_day
min_days
available_from
available_till
0..*
Product_to_sell
price
rented iten details
rented from
rented till
rented on
1..1
orders associated with
1..1
1..1
Delivery_details
DelievryID
completed
Delivery date
Receipant name
Orders
OrderID
amt paid
payment mode
0..*
delievry completion details
Delivery_Exec
DPID
name
contact_no.
stars
delievry man belongs
Delivery_company
compnyID
name
contact_no
country