

***FoodHorse SuperMarket Chain
Database Report***

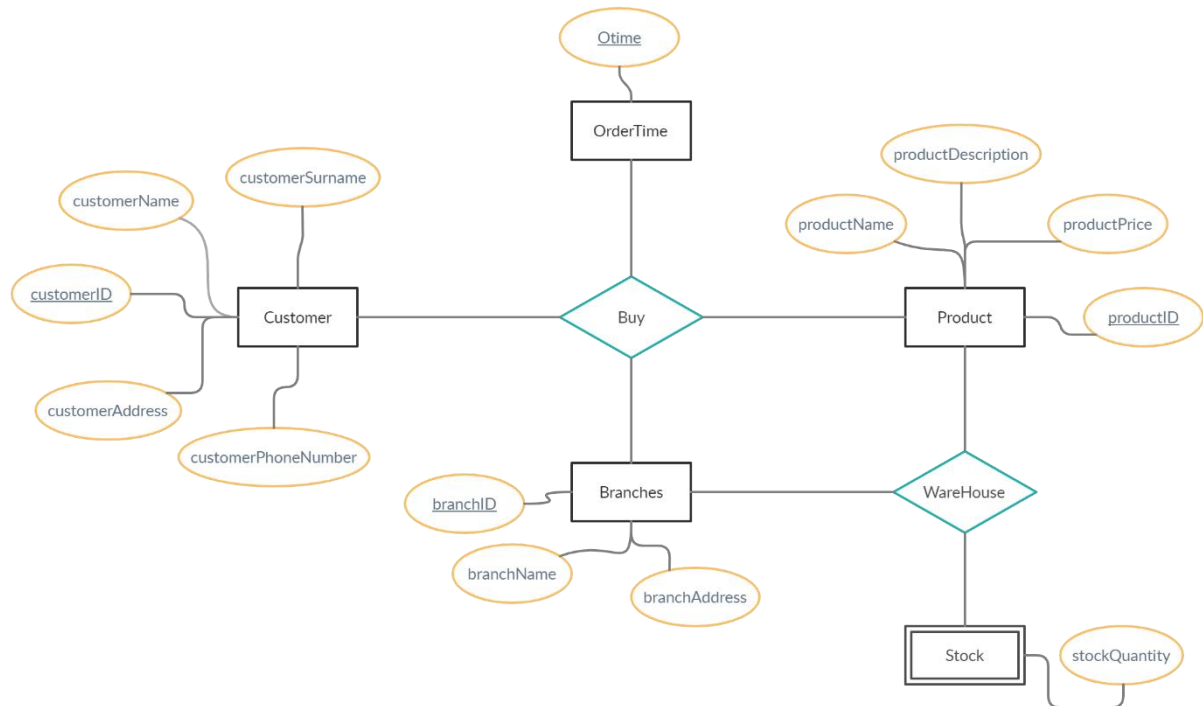
Uygar KAYA

S015570

Özyeğin University

In this Assignment, we are design and create a **FoodHorse** supermarket chain Database schema. In this report I will be explain three parts which are **ER (Entity-Relationship) Diagram**, **MySQL code (DDL (e.g., Create Table) and DML (e.g., Insert into) statements)** and **Java code** which are takes input from the user to interact with the database.

I. ER (Entity-Relationship) Diagram



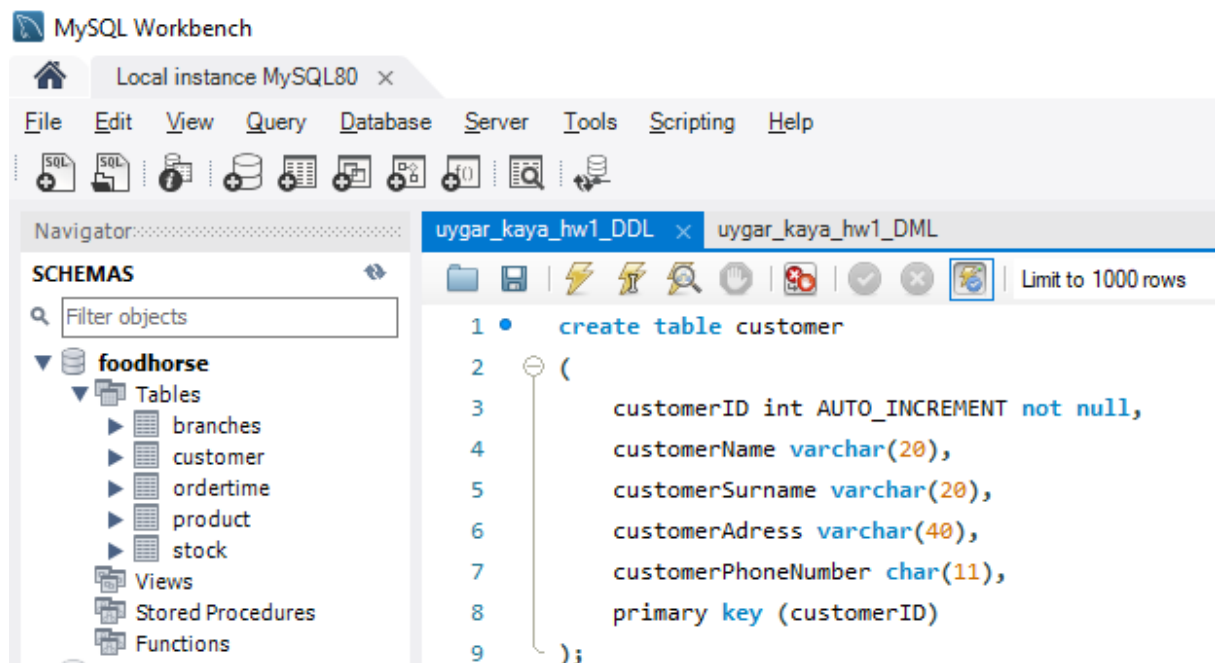
You will see five entities and two relationships in the diagram which I have drawn. The entities are all included in my FoodHorse database as tables. The customer entity has five attributes and this entity has a primary key attribute which is called **"customerID"**, in order to identify each customer uniquely. The branch entity has three attributes, 1 of which is a primary key called **"branchID"**. The product entity has four attributes and the primary key is one of them **"productID"**. Also, I created a OrderTime table in order to keep information about customers' order in the FoodHorse Database and their primary key is **"Otime"** because when the customer given an order, every order has a specific date (YYYY-MM-DD HH:MI:SS), so if we want to specific information, we can use **"Otime"**. When the **"customer"** wants to buy some products, it should go to the market **"branch"** and it buy the some **"products"** what they want, and it should buy a product within a specific certain **"period of time"**. Therefore, these four entities which are **"customer"**, **"branch"**, **"product"** and **"orderTime"** need to be together, so I crate a **"Buy"** relationship to make these entities together. In addition to these entities and relationship, I crated a **"stock"** entity because we should keep our stock in FoodHorse Database when I was designing **"Stock"** entity, I decided that the **"stockQuantity"** attribute shouldn't be primary key since if this attribute is a primary key, different stockQuantities can be given to the same product in the same branch; therefore, stockQuantitiy attribute doesn't a primary key and **"stock"** entity should be weak entity because when we buy the some product if we know the initial stock, we can find out remaining stockQuantity. Also, when the user want to store their **"products"**, they should specify which **"branch"**, and which stock's products store some stock; therefore, it should be the relationships on these entities which are **"branch"**, **"product"** and **"stock"**, so I create a **"wareHouse"** relationship to make these entities together.

MySQL Code

In this part, I created a SQL script including DDL statements (e.g., Create Table) and DML statements (e.g. insert into statements) for the FoodHorse supermarket chain database.

In **DDL (Data Definition Language)** statements, I created tables for FoodHorse supermarket chain tables using **“Create Table”** statements. In this DDL part, I created five tables which are, **customer, branches, product, orderTime and stock** table.

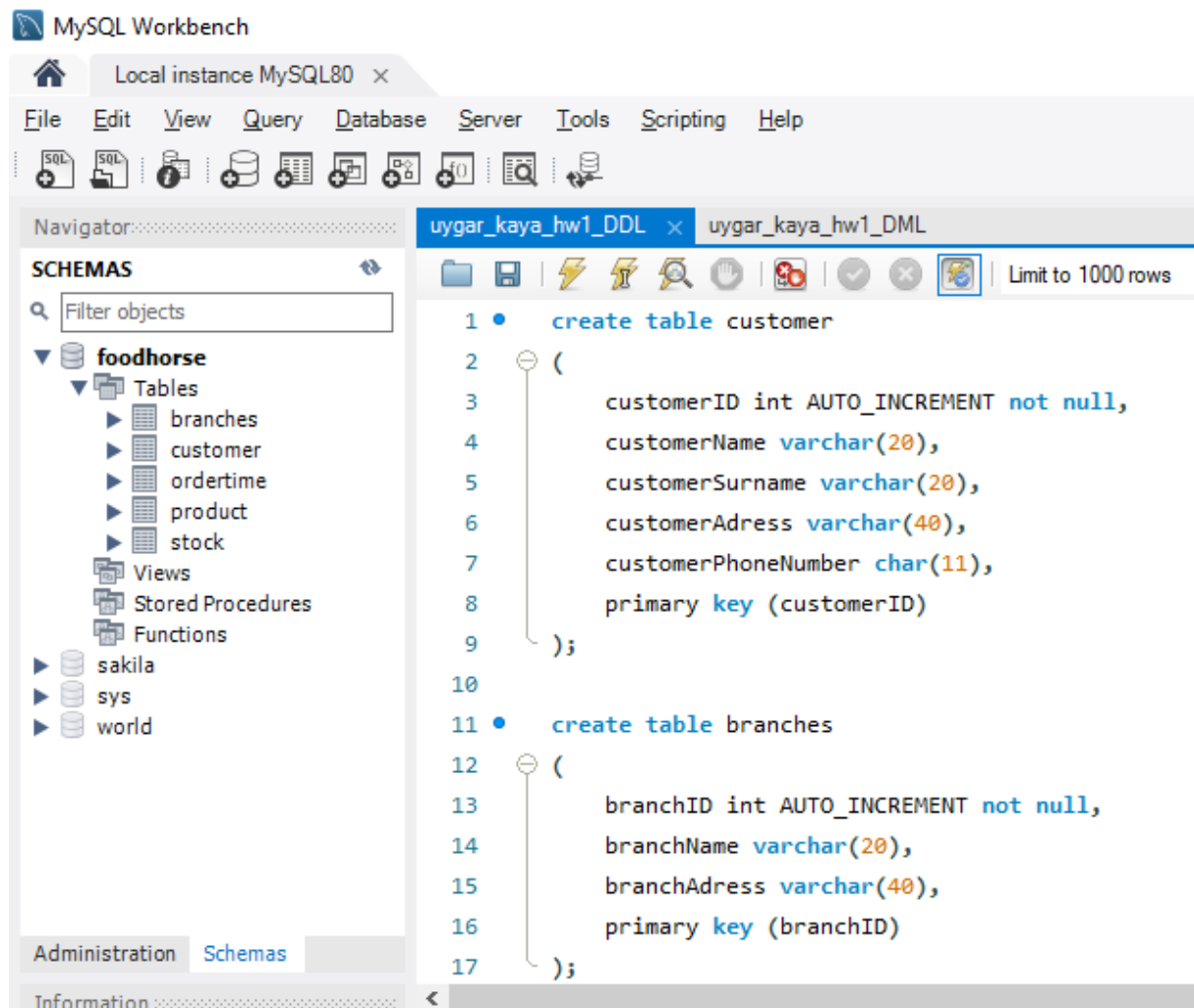
First of All, I created customer table since in given assignment we register a customer; therefore, we should keep customer's data in FoodHorse database. In order to create table, I wrote SQL Query which is **“create table customer”**.



In customer table, we have five attributes which are **customerID, customerName, customerSurname, customerAdress and customerPhoneNumber**. In given assignment one statement is **“When the user registers, a unique id is generated automatically.”** Hence, I created **customerID** attribute which is data type is **“int”** and I use **“AUTO_INCREMENT”** key word in order to create a unique customerID given automatically. Also, I set customerID as primary key because when we want to reach all the specific information of a customer, we can only with a specific ID of that customer.

Every customer has some information, in given assignment when we register a customer we need a customer name, surname, address, and phone number, so I created **customerName, customerSurname and customerAdress** attributes which is data type is **“varchar()”**. When the user enter their name and surname, varchar() provides maximum 20 character, however when the customer enter their address, varchar() provides 40 characters. I set maximum 20 character because some of customer's name and surname can be longer than other customer's name and surname and when the customer given their name, surname and address varchar() allocates as much memory as the character entered; thus, I used varchar() instead of char(). Also, customer has a phone Number, I created customerPhoneNumber attribute which is data type is **“char()”** because every person has an 11 character phone number.

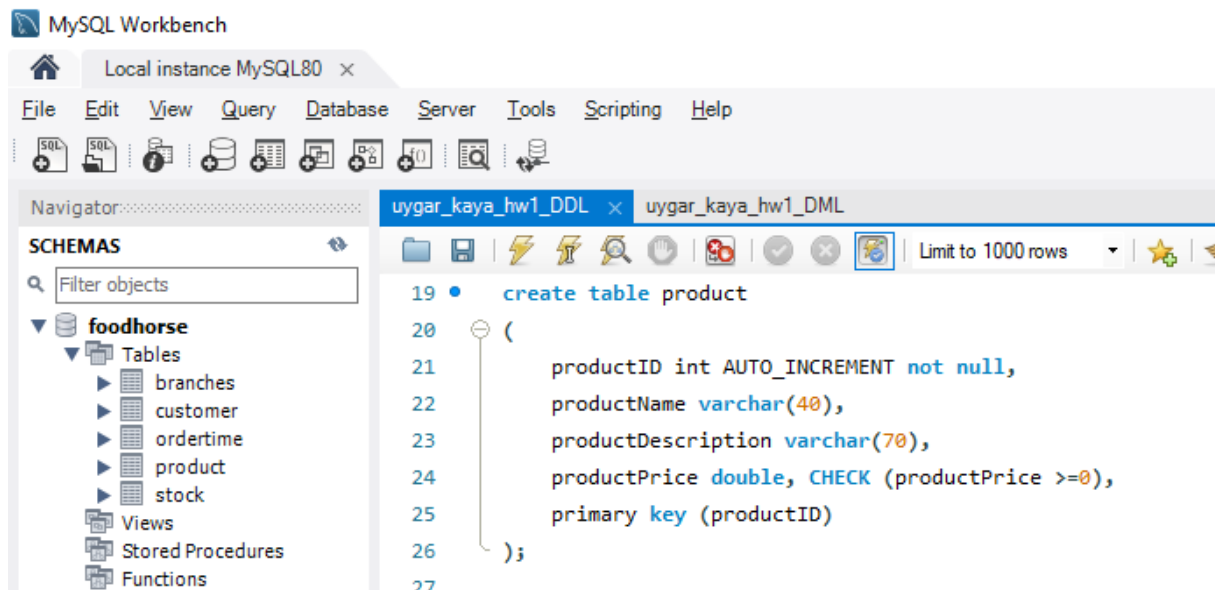
Second of All, I created branches table because in given assignment we have market branches in order to buy some product; therefore, we should keep branches' data in FoodHorse database. In order to create table, I wrote SQL Query which is **"create table branches"**.



In branches table, we have three attributes which are **branchID**, **branchName** and **branchAdress**. In given assignment one statement is **"The program should ask for the id of the branch."** Thus, I created **branchID** which is data type is **"int"** and I use **"AUTO_INCREMENT"** key word so that we can create a unique branchID given automatically. Also, I set branchID as primary key because when we want to reach all the specific information of a branch, we can only with a specific ID of that branches.

In given assignment, when we create a branch, we need a branch name and branch address, so I created **branchName** and **branchAdress** attributes which is data type is **"varchar()"**. When the user enter their branch name, varchar() provides maximum 20 character, but when the user enter their address, varchar() provides 40 characters. I used varchar() instead of char() because varchar() allocates as much memory as the character entered.

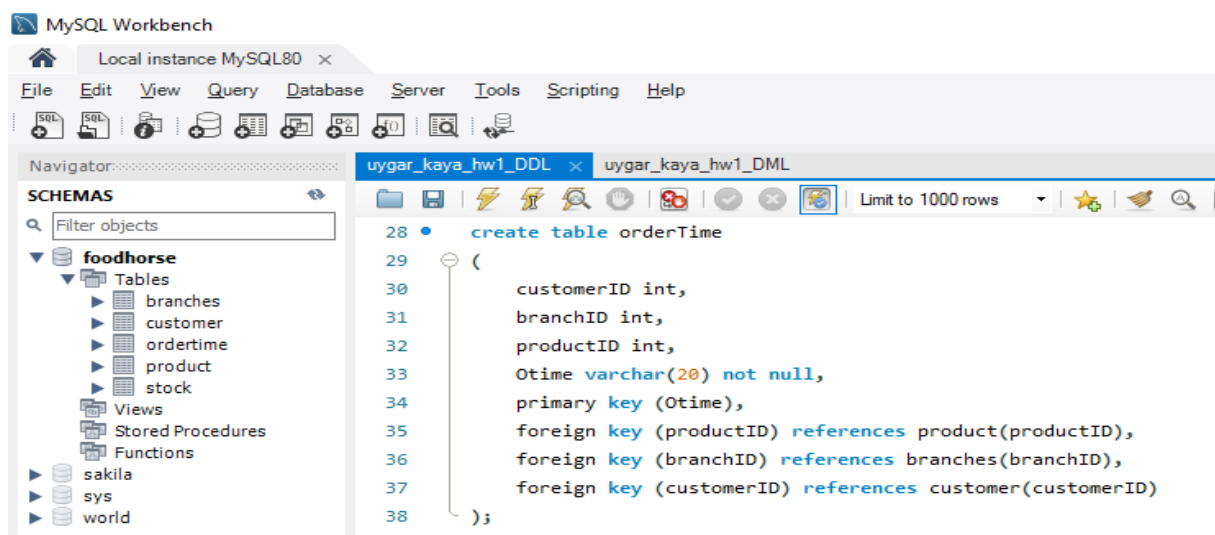
Thirdly, I created product table as when we buy some product or add some product information into the system, we should keep products' data in FoodHorse database. In order to create table, I wrote SQL Query which is **"create table product"**.



In product table, we have four attributes which are **productID**, **productName**, **productDescription** and **productPrice**. In given assignment one statement is *“In this case, the user should specify which product...”* Hence, I generate **productID** attribute and its data type is **“int”** and I use **“AUTO_INCREMENT”** key word in order to generate a unique productID given automatically. Also, I set productID as primary key because when we want to reach all the specific information of a product, we can only with a specific ID of that product.

In given assignment, when we create a product, we need a name, description and price, so I created **productName**, **productDescription** and **productPrice** attributes. **productName** and **productDescription**’s data type is **“varchar()”**. When the user enter their product name, **varchar()** provides maximum 40 character, but when the user enter their product description, **varchar()** provides 70 characters. Furthermore, I crated **productPrice** and its data type is **“double”** because some products can be decimals and it should be greater than or equal 0 in order to check this statement, I wrote SQL Query which is **“productPrice double, check (productPrice >= 0)”**

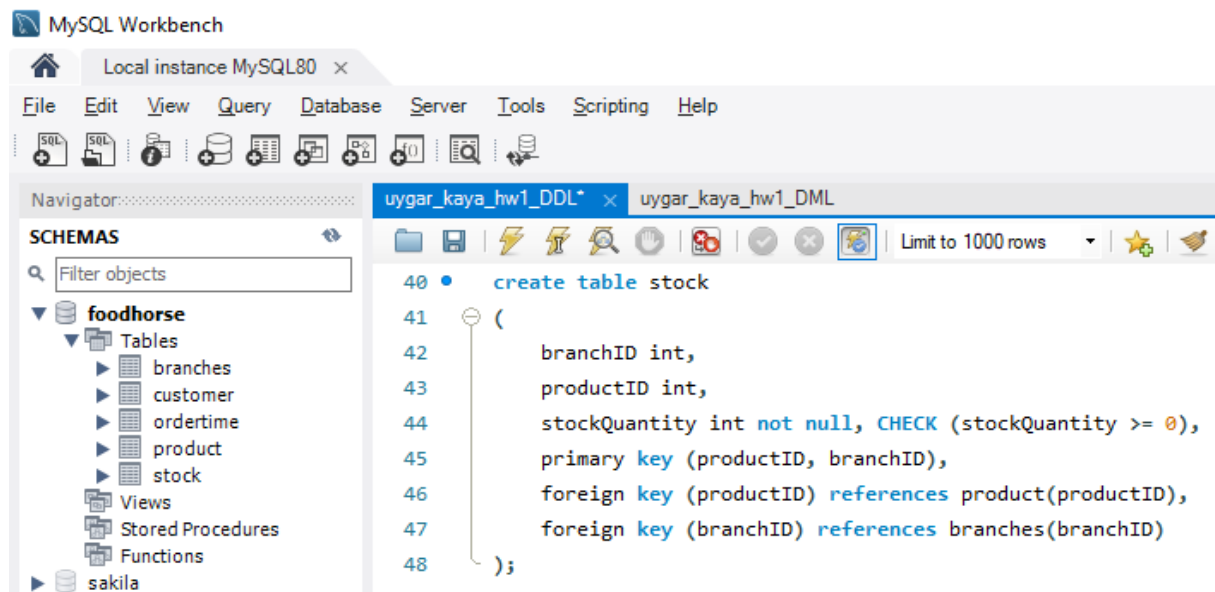
Fourthly, I generate **OrderTime** table because of the fact that in given assignment *“when we buy some product, its purchase information should be kept in your database That is, which customer bought which product from which branch and when”*, so we should keep these information in FoodHorse database. In order to create table, I wrote SQL Query which is **“create table OrderTime”**.



In orderTime table, we have one attributes which are **Otime** because in given assignment one statement is “ask to determine when we bought the product.” Hence, I generate **Otime** attribute and its data type is “**varchar()**” and Otime cannot be null, the user should specify order time because I set Otime as primary key. When each customer purchased the product within a certain time, the Date (YYYY-MM-DD HH:MI:SS) uniquely identifies the which customer bought the which products.

I created **productID, branchID and customerID** as foreign key in order to establish a relationship with other tables since sometimes I print the ID’s of other tables which are **product, branches and customer** table.

Finally, I generate stock table due to the fact that in given assignment one statement is “**Add stock information for a particular product in a given branch.**” Therefore, we should keep stock’ data in FoodHorse database. In order to create table, I wrote SQL Query which is “**create table stock**”.



```

40 • create table stock
41 (
42     branchID int,
43     productID int,
44     stockQuantity int not null, CHECK (stockQuantity >= 0),
45     primary key (productID, branchID),
46     foreign key (productID) references product(productID),
47     foreign key (branchID) references branches(branchID)
48 );
  
```

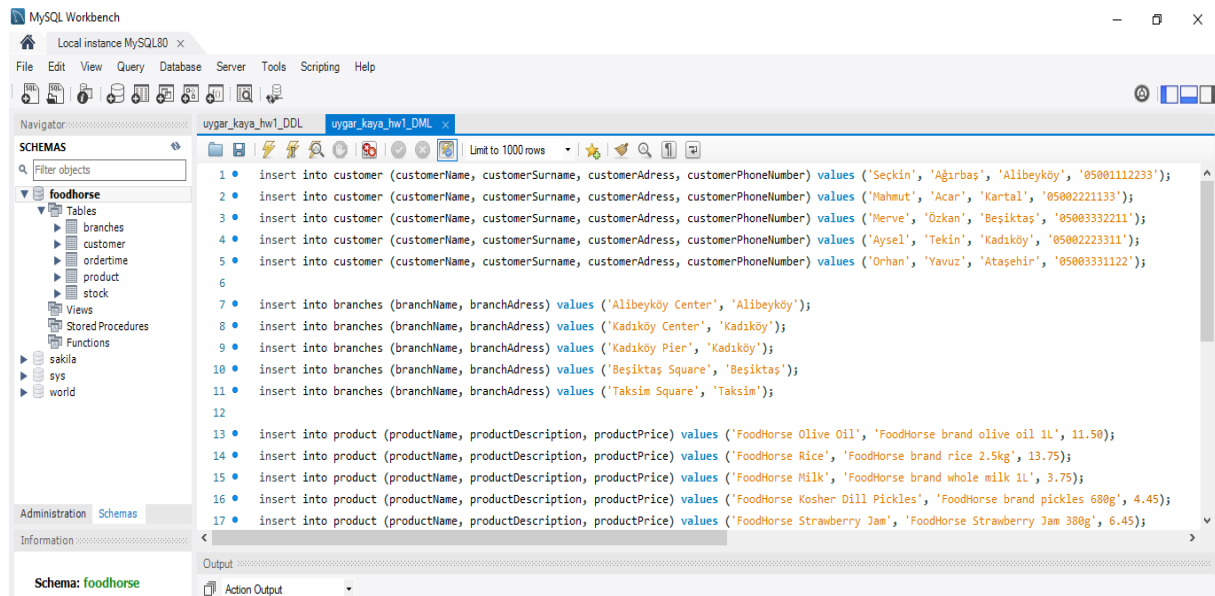
In stock table, we have one attributes which are **stockQuantity** because in given assignment one statement is “ask to determine which product and its quantity to store product.” Hence, I create **stockQuantity** and its data type is “**int**” and it shouldn’t less then 0, so we should check these status in order to check thsesse status I wrote SQL Query which is “**stockQuantity int not null, check (stockQuantity >= 0)**”

In addition to these, I set **productID and branchID** as primary key in order to user should specify which product and its quantity as well as branch number uniquely identifies the branch to store this product. Also, I set **productID and branchID** to avoid duplicates products.

Furthermore, I set **productID and branchID** as foreign key in order to establish a relationship with other tables since sometimes I print the product and branch ID’s of other tables which are **product and branches** table.

DML (Data Manipulation Language) statements is used to manage data in created table or schema objects. In this DML part, we can use some Database terminology such as **“SELECT, INSERT, UPDATE and etc.”**

In MySQL, I used some DML statements which is **“Insert”** however, in Java code I used **“Select and Update”** keywords.



In order to assign value to the specific table, I wrote SQL Query which is, **“INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);”**.

- 1) **“insert into customer (customerName, customerSurname, customerAdress, customerPhoneNumber) values ('Seçkin', 'Ağırbaş', 'Alibeyköy', '05001112233');”**
- 2) **“insert into branches (branchName, branchAdress) values ('Alibeyköy Center', 'Alibeyköy');”**
- 3) **“insert into product (productName, productDescription, productPrice) values ('FoodHorse Olive Oil', 'FoodHorse brand olive oil 1L', 11.50);”**
- 4) **“insert into orderTime(customerID, branchID, productID, Otime) values (1,2,4,'2017-02-09 03:54:12');”**
- 5) **“insert into stock (branchID ,productID, stockQuantity) values (3,1,500);”**

In order to Select a whole table, I wrote SQL Query which is, **“SELECT * FROM table_name”**

- 6) **“select * from customer”**

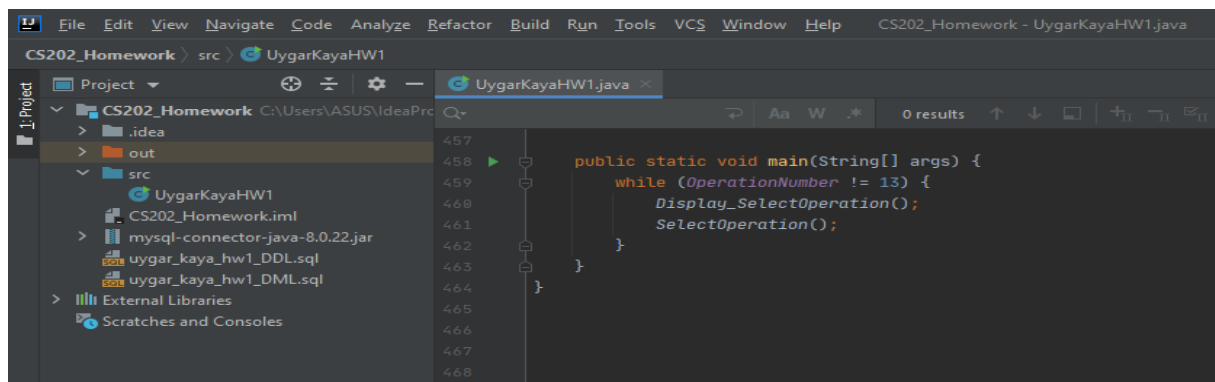
II. JAVA Code with JDBC

First of All, we should connect to the database with JDBC in order to do some work in the Database with java code. In Addition to these, we should get a operation number using a Scanner(***Scanner console = new Scanner(System.in);***) so that user can select operation what they want.

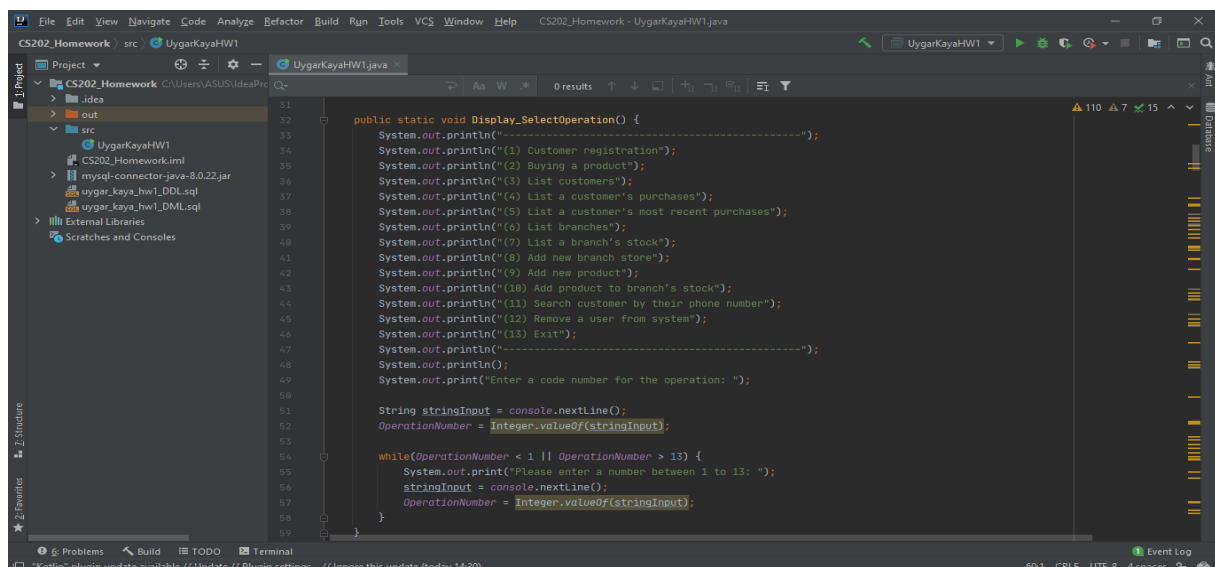
In Java class we have five major methods which are ***main()*** method, ***establishConnection()*** method, ***closeConnection()*** method, ***Display_SelectOperation()*** method and ***SelectOperation()*** method.

In ***establishConnection()*** method, we connect the Database in order to execute some written SQL Query. Also, In ***closeConnection()*** method, we close the connection with Database.

In ***main()*** method, we have the while() loop because the user should display a menu repeatedly until exiting the program and select the operations which operation run. In while() loop, if the user doesn't enter "13", the program continue while() loop, however if the user enter operation number "13", the program should stop and the program don't enter the loop again.

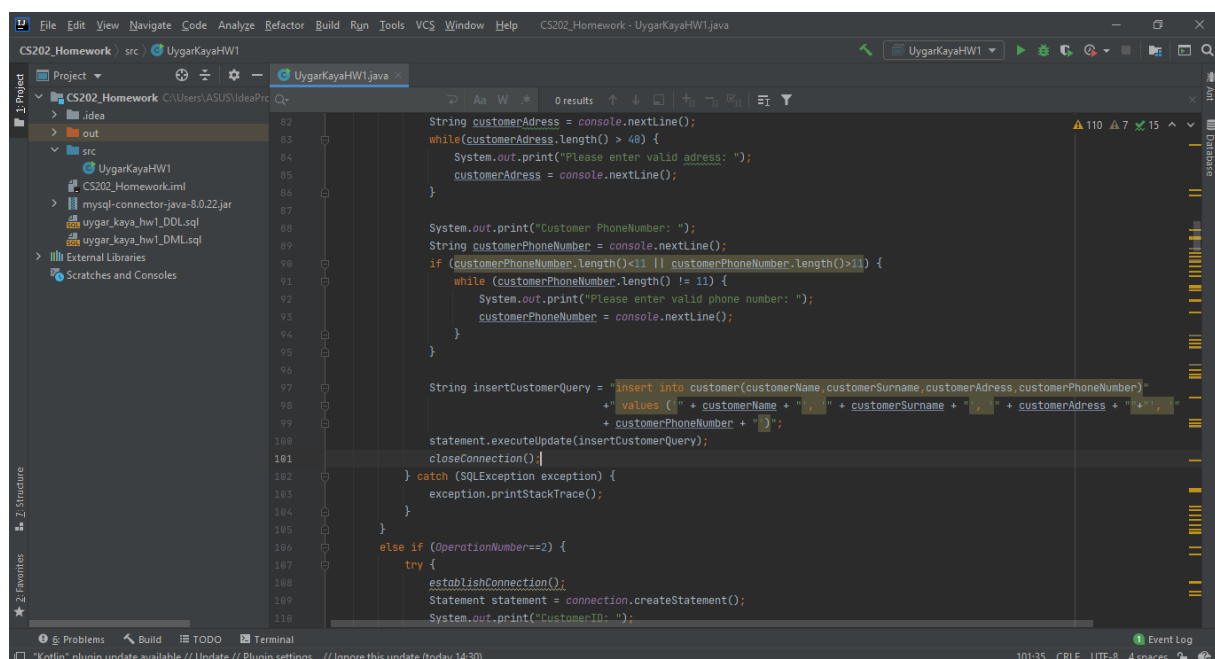


In ***Display_SelectOperation()*** method, we should print the what operations the user can run. In addition to these statement, we should control the user's enter correct "OperationNumber" because if the user enter operation number is less than "1" or grater than "13", we should get the operation number from the user Until the user enters the correct operation number. In this method, I get the operation number with "String" because in some operation user enter the name but in some operation user enter "double" or "int"; therefore, there is a confusion. In order to to be control the confusion, I take all input as strings. If I want to change the double or integer, I can use the "Double.valueOf(Stirng)" or "Integer.valueOf(String)".



In **SelectOperation()** method, we have thirteen if and else if statement in order to execute correct operation.

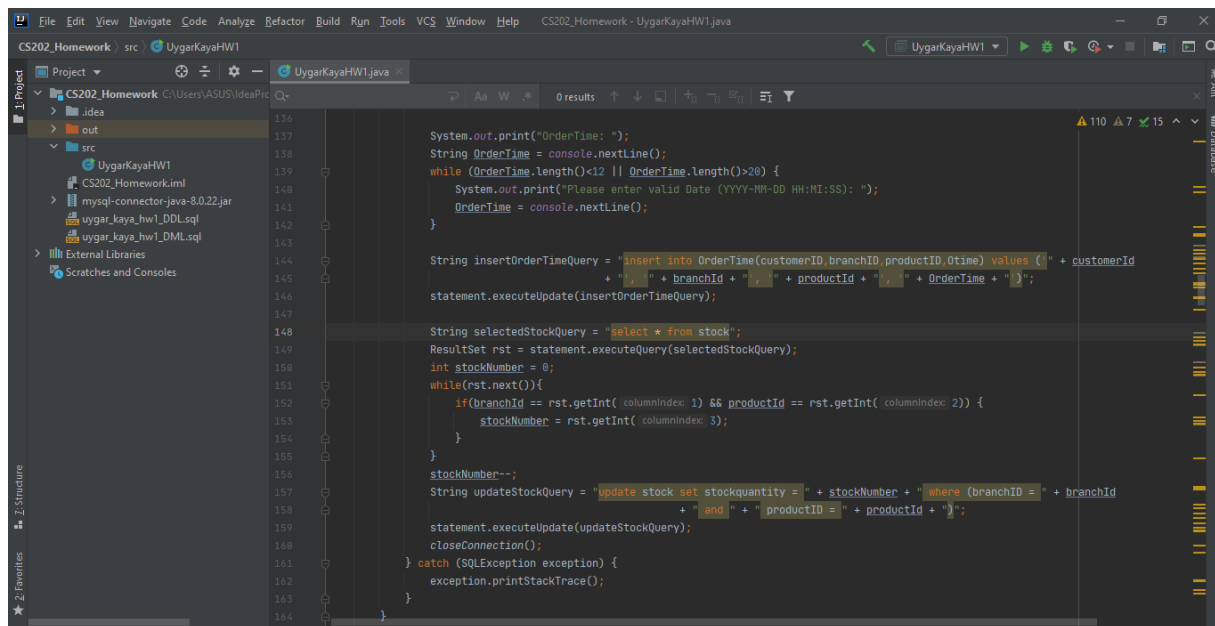
If the user operationNumber is equal to **"1"**, the the program should register a new customer. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();"** code so that we can execute the our SQL Query. Also, I get the **"customerName , customerSurname, customerAddress and customerPhoneNumber"** in order to register a customer we should get these information from the user and I control the these information if **"customerName, customerSurname"** is grater than **"20"**, we should get the customerName and customerSurname from the user Until the user enters the valid customerName and customerSurname because the name and surname can be maximum **"20"** character. Also, user should enter correct phoneNumber, phoneNumber should be **"11"** character in order to be valid. In Addition to these, we should write a SQL Query in order to register a customer in FoodHorse Database; thus, I wrote SQL Query which is,

The screenshot shows an IDE window with a Java file named 'UygurKayaHW1.java'. The code is for a registration function. It starts by getting a customer address and validating its length (must be greater than 40). Then it gets a customer phone number and validates its length (must be 11). It then constructs an SQL insert query: 'insert into customer(customerName, customerSurname, customerAddress, customerPhoneNumber) values (' + customerName + ', ' + customerSurname + ', ' + customerAddress + ', ' + customerPhoneNumber + ');'. The code uses a try-catch block to handle SQL exceptions. Finally, it calls 'closeConnection()' to close the database connection. The code is written in a dark-themed IDE with a project explorer on the left and a run/debug toolbar on the right.

```
82 String customerAddress = console.nextLine();
83 while(customerAddress.length() > 40) {
84     System.out.print("Please enter valid address: ");
85     customerAddress = console.nextLine();
86 }
87
88 System.out.print("Customer PhoneNumber: ");
89 String customerPhoneNumber = console.nextLine();
90 if (customerPhoneNumber.length() < 11 || customerPhoneNumber.length() > 11) {
91     while (customerPhoneNumber.length() != 11) {
92         System.out.print("Please enter valid phone number: ");
93         customerPhoneNumber = console.nextLine();
94     }
95 }
96
97 String insertCustomerQuery = "insert into customer(customerName, customerSurname, customerAddress, customerPhoneNumber)";
98 + "values ('" + customerName + ", " + customerSurname + ", " + customerAddress + ", " +
99 + customerPhoneNumber + ")";
100
101 statement.executeUpdate(insertCustomerQuery);
102 closeConnection();
103 } catch (SQLException exception) {
104     exception.printStackTrace();
105 }
106
107 else if (OperationNumber==2) {
108     try {
109         establishConnection();
110         Statement statement = connection.createStatement();
111         System.out.print("CustomerID: ");
```

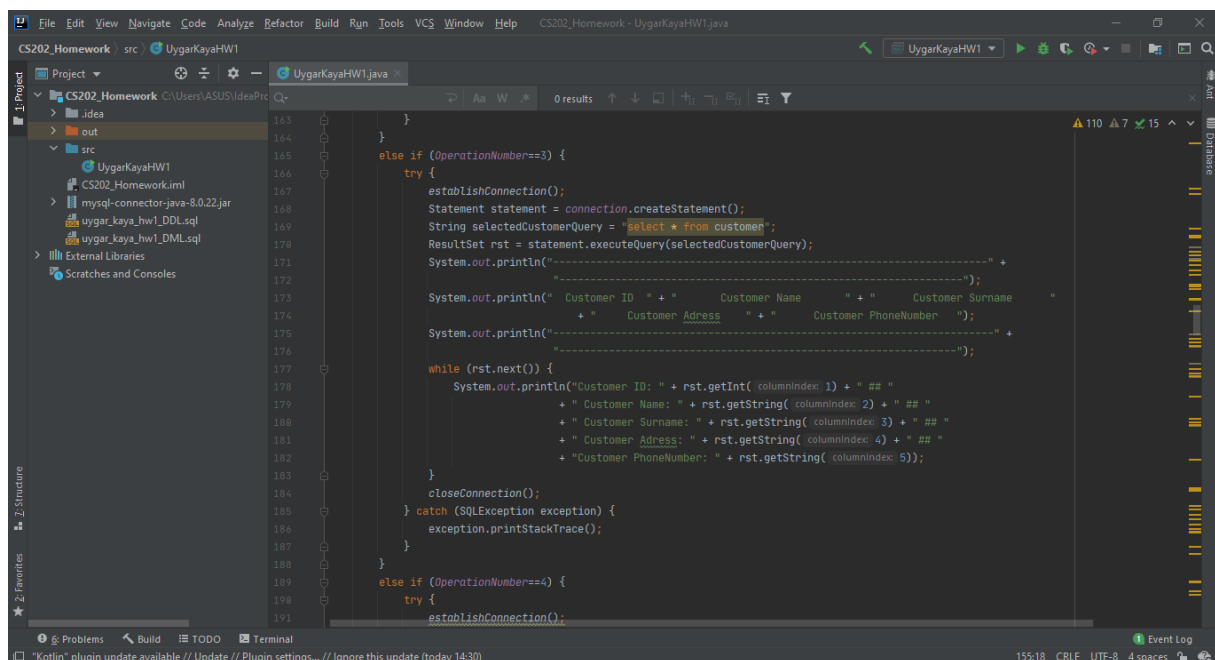
Finally, we should close the connection using **"closeConnection()"** method.

If the user operationNumber is equal to **"2"**, the the program should buy a product. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();"** code so that we can execute the our SQL Query. Also, I get the **"customerID , branchID, productID and orderTime"** in order to register a customer we should get these information from the user and I control the these information if **"customerID, branchID, productID"** is less than or equal **"0"**, we should get the customerID and branchID and productID from the user Until the user enters the valid information because ID's cannot be equal or less than 0. Also, orderTime should be between **"12-20"** character in order to be valid and orderTime format must be **"YYYY-MM-DD HH:MI:SS"**. In Addition to these, we should write two SQL Query in order to insert a order in FoodHorse Database and update the stock Database because when we buy the some product our stock decrease one by one; thus, I wrote SQL Query which is,



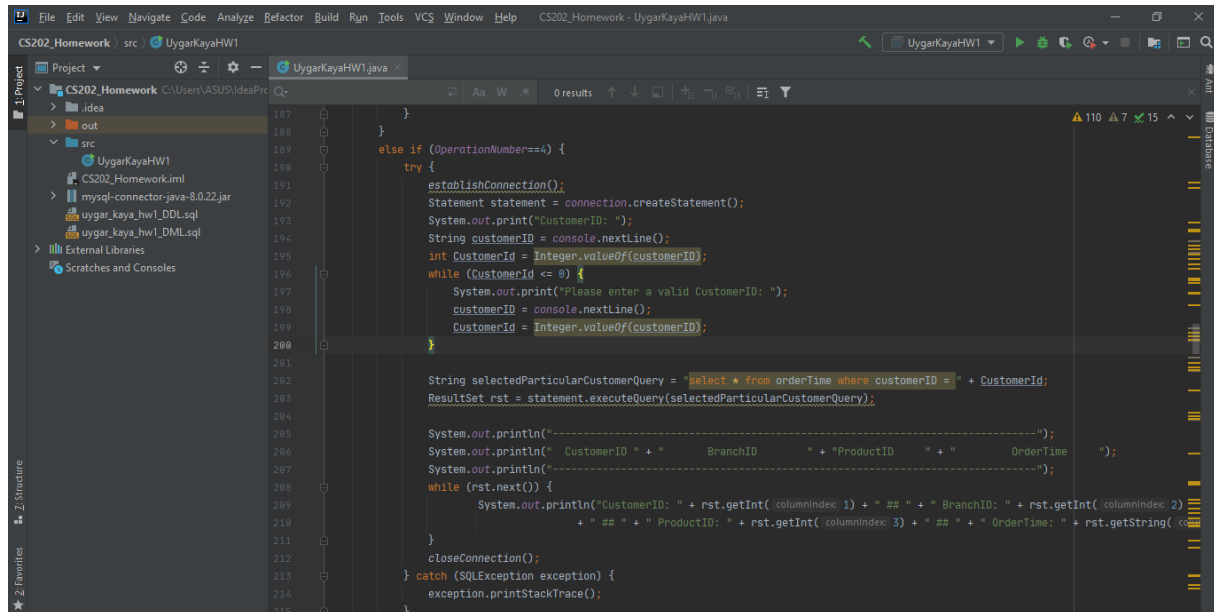
Finally, we should close the connection using ***"closeConnection()"*** method.

If the user operationNumber is equal to ***"3"***, the the program should List all information stored in the system for all the customers that are registered into the system. First of all, I call the ***establishConnection()*** method in order to connect with Database and I write a ***"Statement statement = connection.createStatement();"*** code so that we can execute the our SQL Query. Our SQL Query should be ***"select * from customer"*** because we select all of the data from the FoodHorse Database. In addition to these, I write a ***"ResultSet rst=statement.executeQuery(selectedCustomerQuery);"*** In order to hold data from the database after running the SQL query and I print the customer data when the resultset is end.



Finally, we should close the connection using ***"closeConnection()"*** method.

If the user operationNumber is equal to “4”, the the program should List all purchase information of a particular customer. First of all, I call the ***establishConnection()*** method in order to connect with Database and I write a “***Statement statement = connection.createStatement();***” code so that we can execute the our SQL Query. Our SQL Query should be “***select * from orderTime where customerID = customerId***” because when the user input “***customerId***” is equal to Database customerId, only and only we select all of the data from the FoodHorse Database and user input which is “***customerId***” should be correct; therefore, I check the user input if input less than or equal “0”, the user again enter valid “***customerId***”. In addition to these, I write a “***ResultSet rst=statement.executeQuery(selectedParticularCustomerQuery);***” In order to hold data from the database after running the SQL query and I print the orderTime data when the resultset is end.



```

187     }
188
189     else if (OperationNumber==4) {
190         try {
191             establishConnection();
192             Statement statement = connection.createStatement();
193             System.out.print("CustomerID: ");
194             String customerId = console.nextLine();
195             int CustomerId = Integer.valueOf(customerId);
196             while (CustomerId <= 0) {
197                 System.out.print("Please enter a valid CustomerID: ");
198                 customerId = console.nextLine();
199                 CustomerId = Integer.valueOf(customerId);
200             }
201
202             String selectedParticularCustomerQuery = "select * from orderTime where customerID = " + CustomerId;
203             ResultSet rst = statement.executeQuery(selectedParticularCustomerQuery);
204
205             System.out.println("-----");
206             System.out.println(" CustomerID  " + " BranchID  " + "ProductID  " + " OrderTime  ");
207             System.out.println("-----");
208             while (rst.next()) {
209                 System.out.println("CustomerID: " + rst.getInt( columnIndex 1) + " ## " + " BranchID: " + rst.getInt( columnIndex 2)
210                                     + " ## " + " ProductID: " + rst.getInt( columnIndex 3) + " ## " + " OrderTime: " + rst.getString( columnIndex 4));
211             }
212             closeConnection();
213         } catch (SQLException exception) {
214             exception.printStackTrace();
215         }
216     }

```

Finally, we should close the connection using “***closeConnection()***” method.

If the user operationNumber is equal to “5”, the the program should List the most recent 5 purchase information of a particular customer. First of all, I call the ***establishConnection()*** method in order to connect with Database and I write a “***Statement statement = connection.createStatement();***” code so that we can execute the our SQL Query. Our SQL Query should be “***select * from orderTime where customerID = customerId order by Otime DESC LIMIT 5***” because when the user input “***customerId***” is equal to Database customerId, we select the last “5” purchased items of a specific customer, sorted by order time and user input which is “***customerId***” should be correct; therefore, I check the user input if input less than or equal “0”, the user again enter valid “***customerId***”. In addition to these, I write a “***ResultSet rst=statement.executeQuery(selectedCustomerQuery);***” In order to hold data from the database after running the SQL query and I print the customer data when the resultset is end.

```

217     else if (operationNumber==5) {
218         try {
219             establishConnection();
220             Statement statement = connection.createStatement();
221
222             System.out.println("CustomerID: ");
223             String customerId = console.nextLine();
224             int CustomerId = Integer.valueOf(customerId);
225             while (CustomerId<=0) {
226                 System.out.println("Please enter a valid CustomerID: ");
227                 customerId = console.nextLine();
228                 CustomerId = Integer.valueOf(customerId);
229             }
230
231             String selectedLast5CustomerQuery = "select * from orderTime where customerID = " + CustomerId + " ORDER BY Otime DESC LIMIT 5";
232             ResultSet rst = statement.executeQuery(selectedLast5CustomerQuery);
233
234             System.out.println("-----");
235             System.out.println(" CustomerID " + " BranchID " + "ProductID " + " OrderTime ");
236             System.out.println("-----");
237             while (rst.next()) {
238                 System.out.println("CustomerID: " + rst.getInt( columnIndex: 1) + " ## " + " BranchID: " + rst.getInt( columnIndex: 2)
239                                     + " ## " + " ProductID: " + rst.getInt( columnIndex: 3) + " ## " + " OrderTime: " + rst.getString( columnIndex: 4));
240             }
241             closeConnection();
242         } catch (SQLException exception) {
243             exception.printStackTrace();
244         }
245     }

```

Finally, we should close the connection using ***"closeConnection()"*** method.

If the user operationNumber is equal to ***"6"***, the the program should List all available information of all the branches that are in the system. First of all, I call the ***establishConnection()*** method in order to connect with Database and I write a ***"Statement statement = connection.createStatement();"*** code so that we can execute the our SQL Query. Our SQL Query should be ***"select * from branches"*** because we select all of the data from the FoodHorse Database. In addition to these, I write a ***"ResultSet rst=statement.executeQuery(selectedBranchesQuery);"*** In order to hold data from the database after running the SQL query and I print the branches data when the resultset is end.

```

241         closeConnection();
242     } catch (SQLException exception) {
243         exception.printStackTrace();
244     }
245
246     else if (operationNumber==6) {
247         try {
248             establishConnection();
249             Statement statement = connection.createStatement();
250             String selectedBranchesQuery = "select * from branches";
251             ResultSet rst = statement.executeQuery(selectedBranchesQuery);
252
253             System.out.println("-----");
254             System.out.println(" Branch ID " + " Branch Name " + " Branch Address ");
255             System.out.println("-----");
256             while (rst.next()) {
257                 System.out.println("Branch ID: " + rst.getInt( columnIndex: 1) + " ## "
258                                     + " Branch Name: " + rst.getString( columnIndex: 2) + " ## "
259                                     + " Branch Address: " + rst.getString( columnIndex: 3));
260             }
261             closeConnection();
262         } catch (SQLException exception) {
263             exception.printStackTrace();
264         }
265
266     else if (operationNumber==7) {
267         try {
268             establishConnection();
269             Statement statement = connection.createStatement();
270             System.out.println("BranchID: ");

```

Finally, we should close the connection using ***"closeConnection()"*** method.

If the user operationNumber is equal to **"7"**, the the program should List all the products that are sold in a particular branch. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();"** code so that we can execute the our SQL Query. Our SQL Query should be **"select * from orderTime where branchID = branchId"** because when the user input **"branchId"** is equal to Database customerID, only and only we select all of the data from the FoodHorse Database in branches table and user input which is **"branchId"** should be valid; therefore, I check the user input. If user input less than or equal **"0"**, the user again enter valid **"branchId"**. In addition to these, I write a **"ResultSet rst=statement.executeQuery(selectedbranchIDQuery);"** In order to hold data from the database after running the SQL query and I print the orderTime data when the resultset is end.

```

265     else if (operationNumber==7) {
266     try {
267         establishConnection();
268         Statement statement = connection.createStatement();
269         System.out.print("BranchID: ");
270         String branchId = console.nextLine();
271         int BranchId = Integer.valueOf(branchId);
272         while (BranchId <= 0) {
273             System.out.print("Please enter a valid BranchID: ");
274             branchId = console.nextLine();
275             BranchId = Integer.valueOf(branchId);
276         }
277         String selectedBranchIDQuery = "select * from orderTime where branchID = " + branchId;
278         ResultSet rst = statement.executeQuery(selectedBranchIDQuery);
279
280         System.out.println("-----");
281         System.out.println(" CustomerID " + "      BranchID      " + "ProductID      " + "      OrderTime      ");
282         System.out.println("-----");
283         while (rst.next()) {
284             System.out.println("CustomerID: " + rst.getInt( columnIndex: 1) + " ## " + " BranchID: " + rst.getInt( columnIndex: 2)
285                                 + " ## " + " ProductID: " + rst.getInt( columnIndex: 3) + " ## " + " OrderTime: " + rst.getString( columnIndex: 4));
286         }
287         closeConnection();
288     } catch (SQLException exception) {
289         exception.printStackTrace();
290     }
291 }
292 else if (operationNumber==8) {
293     try {

```

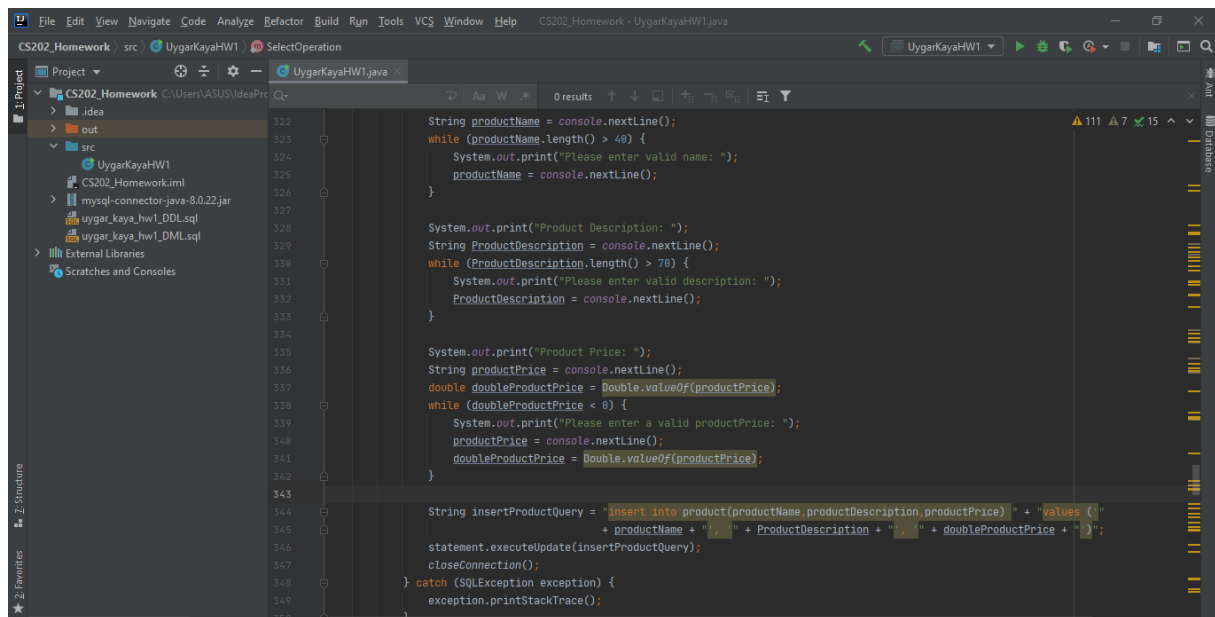
Finally, we should close the connection using **"closeConnection()"** method.

If the user operationNumber is equal to **"8"**, the the program should Add new market branch information into the system. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();"** code so that we can execute the our SQL Query. Also, I get the **"branchName , branchAdress"** in order to add a new branch we should get these information from the user and I control the these information if **"branchName"** is grater than **"20"** character and **"branchAdress"** is grater than **"40"** character, we should get the branchName and branchSurname from the user Until the user enters the valid branchName and branchSurname because the name can be maximum **"20"** character and adress can be maximum **"40"** character. In Addition to these, we should write a SQL Query in order to add a new branch in FoodHorse Database branches table; thus, I wrote SQL Query which is,

"insert into branches(branchName,branchAdress) values ('" +branchName+"','" +branchAdress+"')"

Finally, we should close the connection using **"closeConnection()"** method.

If the user operationNumber is equal to “9”, the the program should Add new product information into the system. First of all, I call the **establishConnection()** method in order to connect with Database and I write a “**Statement statement = connection.createStatement();**” code so that we can execute the our SQL Query. Also, I get the “**productName , productDescription and productPrice**” in order to add a new product we should get these information from the user and I control the these information if “**productName**” is grater than “20” character and “**productDescription**” is grater than “70” character, we should get the productName and productDescription from the user Until the user enters the valid productName and productDescription because the name can be maximum “40” character and description can be maximum “70” character. Also, “**productPrice**” should be grater than “0” because product price cannot be negative value In Addition to these, we should write a SQL Query in order to add a new branch in FoodHorse Database branches table; thus, I wrote SQL Query which is,



```

322 String productName = console.nextLine();
323 while (productName.length() > 40) {
324     System.out.print("Please enter valid name: ");
325     productName = console.nextLine();
326 }
327
328 System.out.print("Product Description: ");
329 String ProductDescription = console.nextLine();
330 while (ProductDescription.length() > 70) {
331     System.out.print("Please enter valid description: ");
332     ProductDescription = console.nextLine();
333 }
334
335 System.out.print("Product Price: ");
336 String productPrice = console.nextLine();
337 double doubleProductPrice = Double.valueOf(productPrice);
338 while (doubleProductPrice < 0) {
339     System.out.print("Please enter a valid productPrice: ");
340     productPrice = console.nextLine();
341     doubleProductPrice = Double.valueOf(productPrice);
342 }
343
344 String insertProductQuery = "insert into product(productName,productDescription,productPrice) " + "values ("
345     + productName + "," + ProductDescription + "," + doubleProductPrice + ")";
346 statement.executeUpdate(insertProductQuery);
347 closeConnection();
348 } catch (SQLException exception) {
349     exception.printStackTrace();
350 }

```

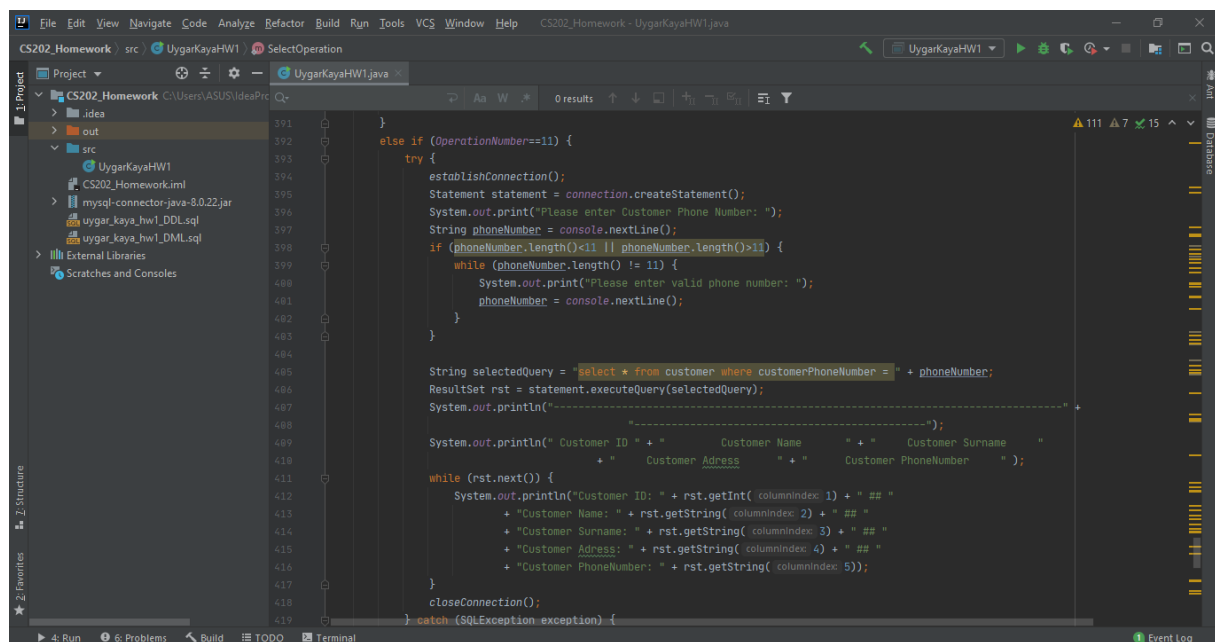
Finally, we should close the connection using “**closeConnection()**” method.

If the user operationNumber is equal to “10”, the the program should Add stock information for a particular product in a given branch. First of all, I call the **establishConnection()** method in order to connect with Database and I write a “**Statement statement = connection.createStatement();**” code so that we can execute the our SQL Query. Also, I get the “**branchID , productID and StockQuantity**” in order to add a new stock we should get these information from the user and I control the these information if “**branchID**”, “**productID**” and “**stockQuantity**” are less than “0”, we should get the branchID and productID and stockQuantity from the user Until the user enters the valid information. In Addition to these, we should write a SQL Query in order to add a new stock in FoodHorse Database stock table; thus, I wrote SQL Query which is,

“**insert into stock(branchID,productID,stockQuantity) " + "values (" + branchId + ", " + productId + ", " + StockQuantity + ")**”

Finally, we should close the connection using “**closeConnection()**” method.

If the user operationNumber is equal to **"11"**, the the program should Display all information of a specific customer given the customer's phone number. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();" code so that we can execute the our SQL Query. Our SQL Query should be "select * from customer where customerPhoneNumber = phoneNumber"** because when the user input **"phoneNumber"** is equal to Database customerPhoneNumber, only and only we select all of the data from the FoodHorse Database and user input which is **"phoneNumber"** should be correct; therefore, I check the user input if input dosen't equal **"11"** character, the user again enter valid **"phoneNumber"**. In addition to these, I write a **"ResultSet rst=statement.executeQuery(selectedQuery);"** In order to hold data from the database after running the SQL query and I print the customer data when the resultset is end.



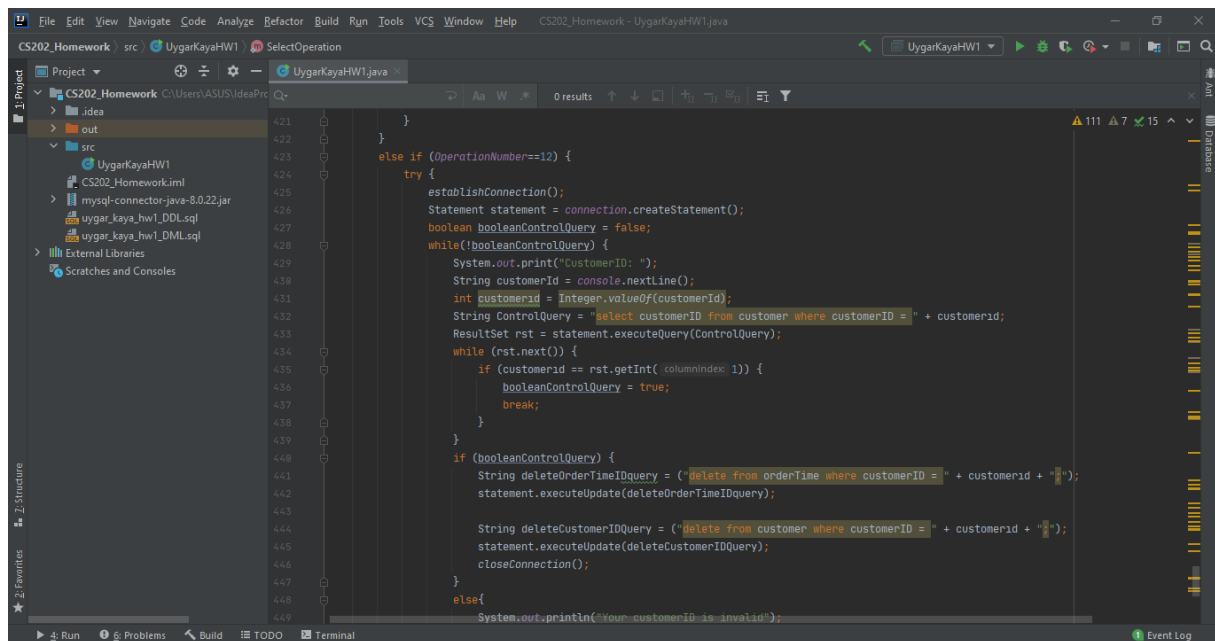
```

391     }
392     else if (OperationNumber==11) {
393         try {
394             establishConnection();
395             Statement statement = connection.createStatement();
396             System.out.print("Please enter Customer Phone Number: ");
397             String phoneNumber = console.nextLine();
398             if (phoneNumber.length()<11 || phoneNumber.length()>11) {
399                 while (phoneNumber.length() != 11) {
400                     System.out.print("Please enter valid phone number: ");
401                     phoneNumber = console.nextLine();
402                 }
403             }
404
405             String selectedQuery = "select * from customer where customerPhoneNumber = " + phoneNumber;
406             ResultSet rst = statement.executeQuery(selectedQuery);
407             System.out.println("-----");
408
409             System.out.println(" Customer ID  " + " Customer Name  " + " Customer Surname  "
410                               + " Customer Address  " + " Customer PhoneNumber  ");
411
412             while (rst.next()) {
413                 System.out.println("Customer ID: " + rst.getInt( columnIndex: 1) + " ## "
414                                   + "Customer Name: " + rst.getString( columnIndex: 2) + " ## "
415                                   + "Customer Surname: " + rst.getString( columnIndex: 3) + " ## "
416                                   + "Customer Address: " + rst.getString( columnIndex: 4) + " ## "
417                                   + "Customer PhoneNumber: " + rst.getString( columnIndex: 5));
418             }
419             closeConnection();
420         } catch (SQLException exception) {

```

Finally, we should close the connection using **"closeConnection()"** method.

If the user operationNumber is equal to **"12"**, the program should delete a customer that exists in the database if customer doesn't exists in Database we should give valid CustomerID from the user. First of all, I call the **establishConnection()** method in order to connect with Database and I write a **"Statement statement = connection.createStatement();" code so that we can execute the our SQL Query. Our SQL Query should be "select customerID from customer where customerID = customerid"** because when the user input **"customerid"** is equal to Database customerID, only and only we select customerID of the data from the FoodHorse Database because we should check customer is exists in FoodHorse Database or not if the customerID is correct and customer exists in Database, we can delete particular customer information form the customer table and their order's from the orderTime Table. In addition to these, I write a **"ResultSet rst=statement.executeQuery(ControlQuery);"** In order to hold data from the database after running the SQL query.



Finally, we should close the connection using “**closeConnection()**” method.

If the user operationNumber is equal to “**13**”, the the program should Exits the application. First of all, I call the **establishConnection()** method in order to connect with Database and I write a “**Statement statement = connection.createStatement();**” code so that we can execute the our SQL Query and I call “**System.exit(0)**” in order to finish the execution of the program finally we should close the connection using “**closeConnection()**” method.

