Modern Database System

First Task

Warehouse Application

Prepared By: Fadi Atallah (OU3IYA)

Supervised By: Dr. Kovács László

Table of Contents

Table Of Figures

1- Database Implementation:

I worked on oracle DBMS to create the database.

1.1 - ER Model:

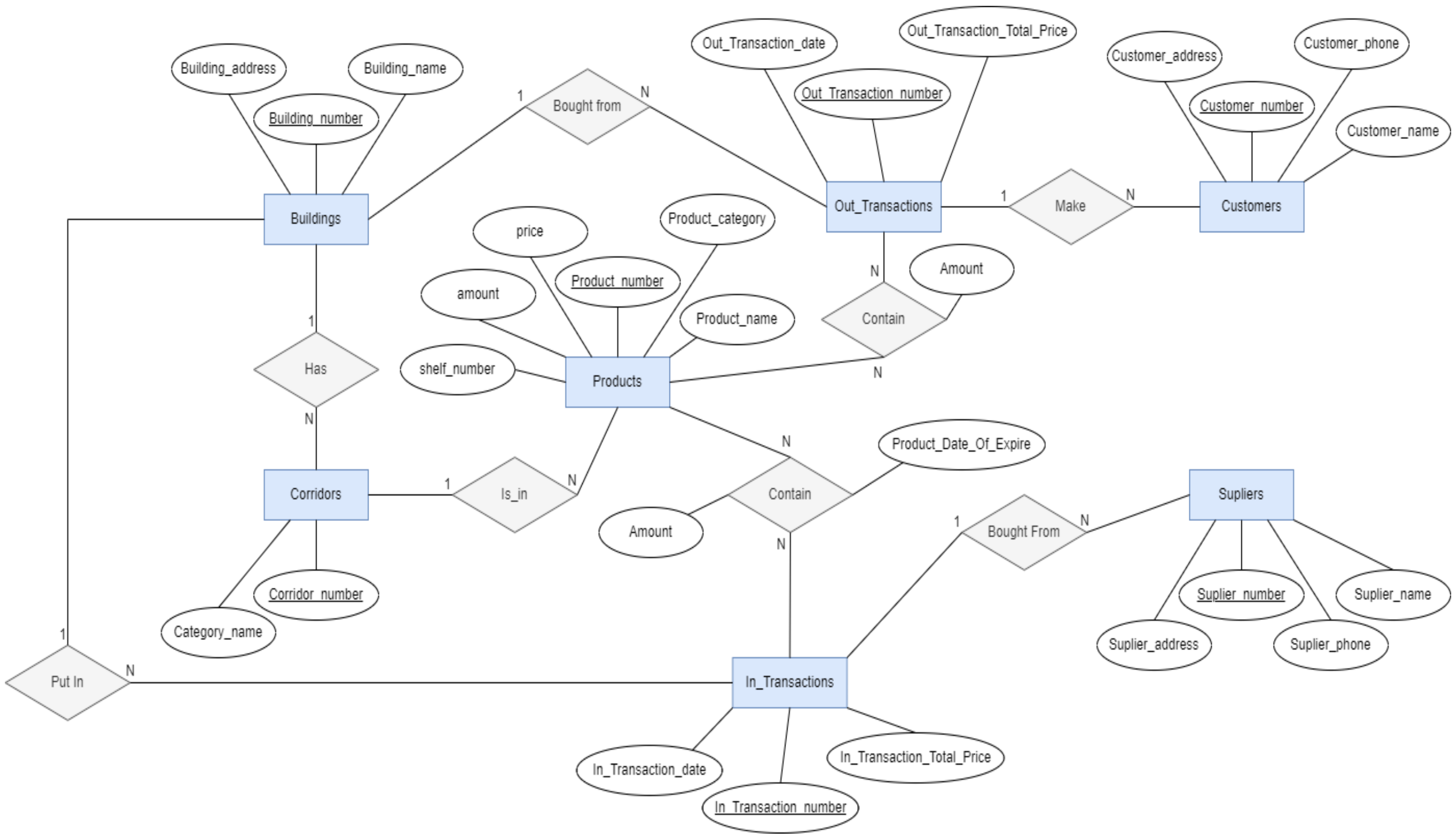


Figure 1-1 - ER Model

1.2 - Relational Model:

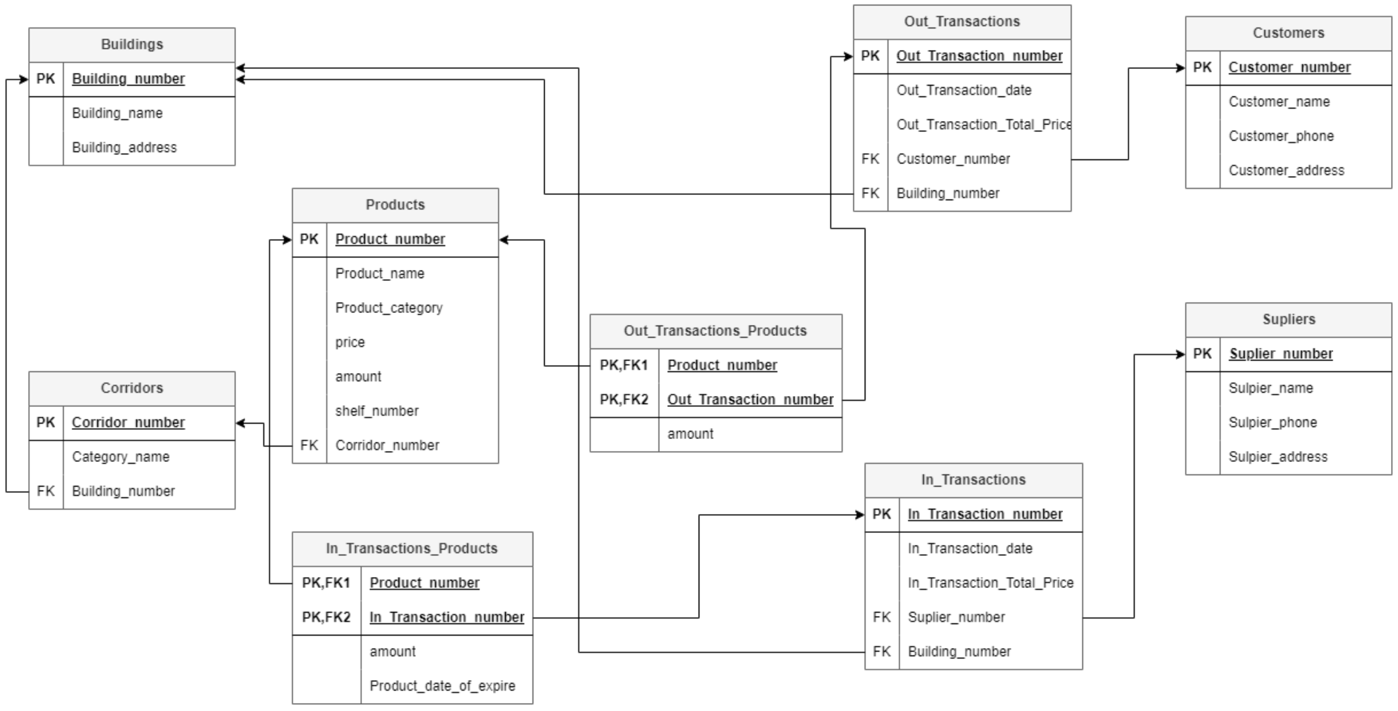


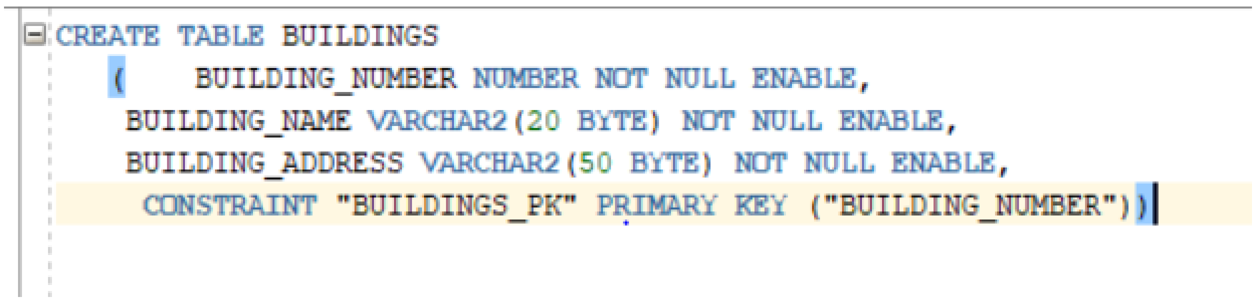
Figure 1-2 - Relational Model

1.3 – SQL Implementation:

1.3.1- Create Tables:

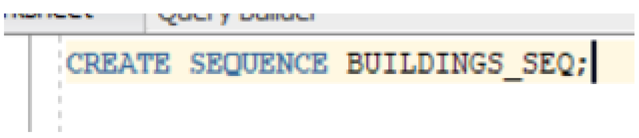
- First of all, I started with creating the tables shown in the Relational Model.

The first table was Buildings table. I used this SQL command to create it:



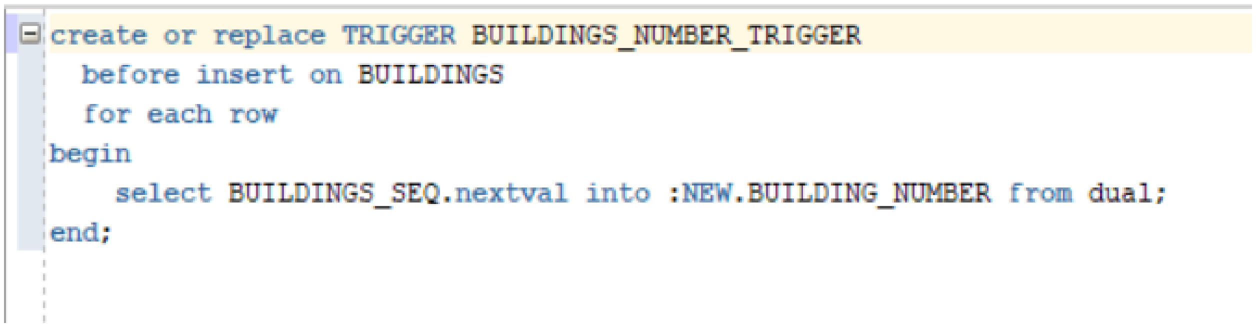
```
CREATE TABLE BUILDINGS
   (    BUILDING_NUMBER NUMBER NOT NULL ENABLE,
    BUILDING_NAME VARCHAR2(20 BYTE) NOT NULL ENABLE,
    BUILDING_ADDRESS VARCHAR2(50 BYTE) NOT NULL ENABLE,
     CONSTRAINT "BUILDINGS_PK" PRIMARY KEY ("BUILDING_NUMBER"))
```

Then I create a sequence called BUILDINGS_SEQ to use its value as unique value for each row while inserting new row to the table:



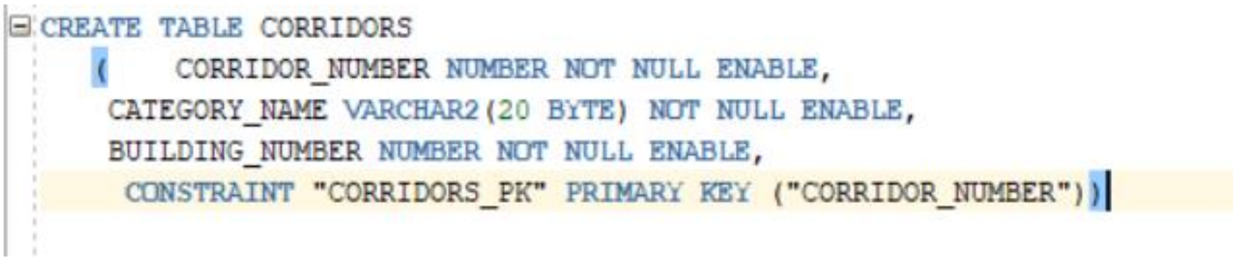
```
CREATE SEQUENCE BUILDINGS_SEQ;
```

After That I created a trigger which will execute before inserting into building table so I can assign the sequence value to the primary key of the Buildings table:



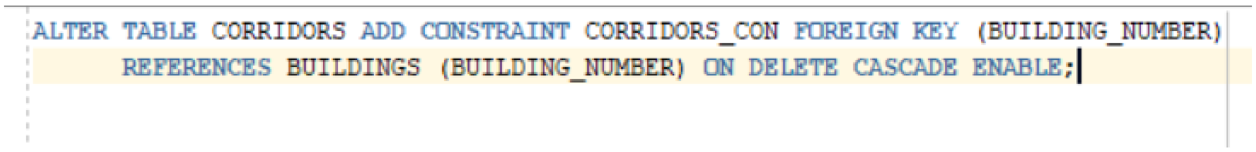
```
create or replace TRIGGER BUILDINGS_NUMBER_TRIGGER
  before insert on BUILDINGS
  for each row
begin
    select BUILDINGS_SEQ.nextval into :NEW.BUILDING_NUMBER from dual;
end;
```

- After I finished with Buildings table, I created Corridors table that have the Building_Number as a foreign_key in it as shown below:



```
CREATE TABLE CORRIDORS
    (    CORRIDOR_NUMBER NUMBER NOT NULL ENABLE,
     CATEGORY_NAME VARCHAR2(20 BYTE) NOT NULL ENABLE,
     BUILDING_NUMBER NUMBER NOT NULL ENABLE,
      CONSTRAINT "CORRIDORS_PK" PRIMARY KEY ("CORRIDOR_NUMBER"))
```

This SQL command is to set Building_Number as foreign key and put the Building_Number column in Buildings table as a reference:

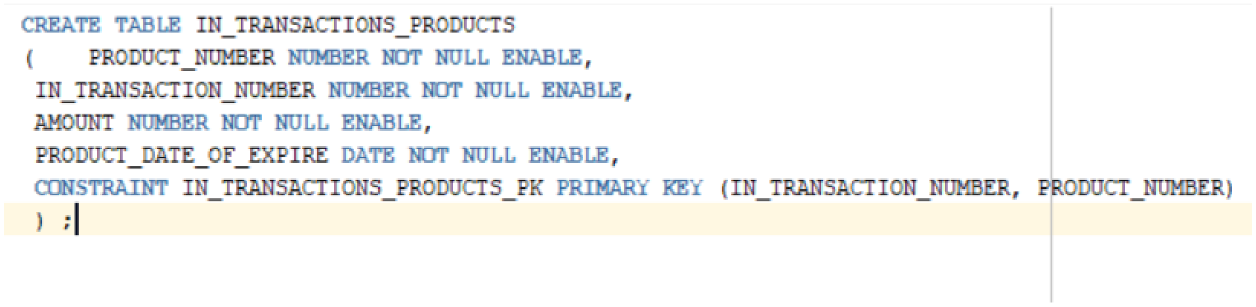


```
ALTER TABLE CORRIDORS ADD CONSTRAINT CORRIDORS_CON FOREIGN KEY (BUILDING_NUMBER)
      REFERENCES BUILDINGS (BUILDING_NUMBER) ON DELETE CASCADE ENABLE;
```

As with the previous table, I created a sequence and a trigger to make the primary key as auto generated.

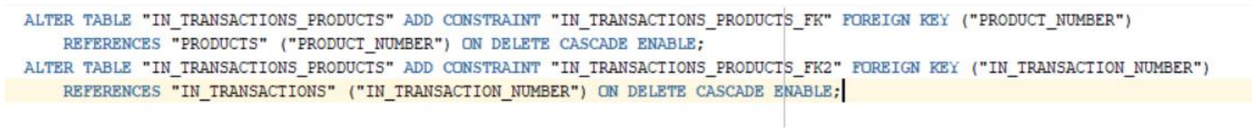
And so on for the Products, Customers, Supliers, In_Transactions, and Out_Transactions tables.

- For In_Transactions_Products table I used both Product_Number , And In_Transaction_Number as primary key:



```
CREATE TABLE IN_TRANSACTIONS_PRODUCTS
 (    PRODUCT_NUMBER NUMBER NOT NULL ENABLE,
  IN_TRANSACTION_NUMBER NUMBER NOT NULL ENABLE,
  AMOUNT NUMBER NOT NULL ENABLE,
  PRODUCT_DATE_OF_EXPIRE DATE NOT NULL ENABLE,
  CONSTRAINT IN_TRANSACTIONS_PRODUCTS_PK PRIMARY KEY (IN_TRANSACTION_NUMBER, PRODUCT_NUMBER)
  ) ;
```

And I also connected them with the Products, and In_Transaction tables as foreign keys:



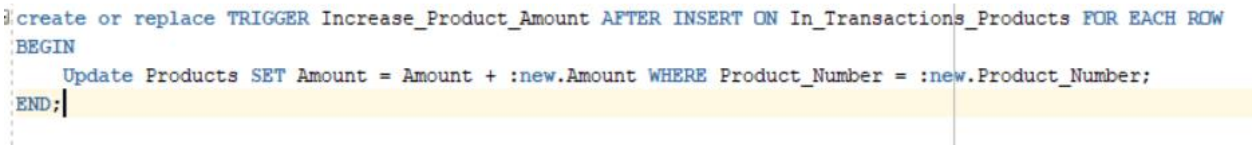
```
ALTER TABLE "IN_TRANSACTIONS_PRODUCTS" ADD CONSTRAINT "IN_TRANSACTIONS_PRODUCTS_FK" FOREIGN KEY ("PRODUCT_NUMBER")
    REFERENCES "PRODUCTS" ("PRODUCT_NUMBER") ON DELETE CASCADE ENABLE;
ALTER TABLE "IN_TRANSACTIONS_PRODUCTS" ADD CONSTRAINT "IN_TRANSACTIONS_PRODUCTS_FK2" FOREIGN KEY ("IN_TRANSACTION_NUMBER")
    REFERENCES "IN_TRANSACTIONS" ("IN_TRANSACTION_NUMBER") ON DELETE CASCADE ENABLE;
```

And also, I did the same for the Out_Transactions_Products.

1.3.2 – Create Triggers:

I created four triggers to update some values after inserting into In_Transactions_Products, and Out_Transactions_Products.

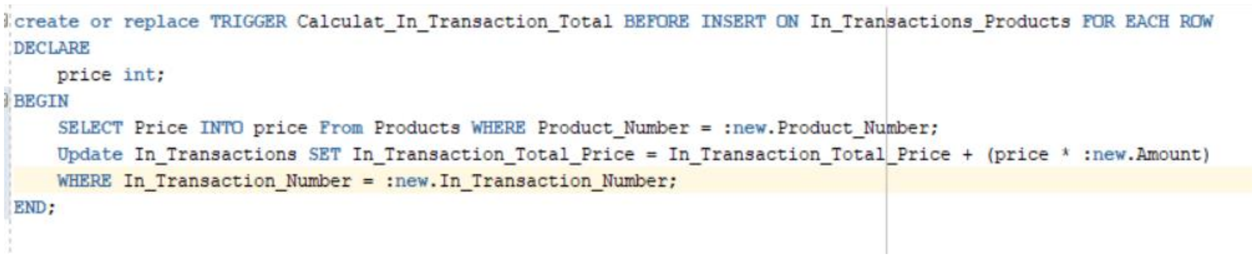
- The first trigger is (Increase_Product_Amount) that should execute after inserting into In_Transactions_Products to increase the product amount in Products table:



```
create or replace TRIGGER Increase_Product_Amount AFTER INSERT ON In_Transactions_Products FOR EACH ROW
BEGIN
    Update Products SET Amount = Amount + :new.Amount WHERE Product_Number = :new.Product_Number;
END;
```

While new.Amount is the value of amount column in In_Transactions_Products table.

- Another trigger is also after inserting into In_Transactions_Products. This trigger is to calculate total price for the In_Transaction by multiply the product price with the amount that inserted to the table. It is called (Calculat_In_Transaction_Total)



```
create or replace TRIGGER Calculat_In_Transaction_Total BEFORE INSERT ON In_Transactions_Products FOR EACH ROW
DECLARE
    price int;
BEGIN
    SELECT Price INTO price From Products WHERE Product_Number = :new.Product_Number;
    Update In_Transactions SET In_Transaction_Total_Price = In_Transaction_Total_Price + (price * :new.Amount)
    WHERE In_Transaction_Number = :new.In_Transaction_Number;
END;
```

- The other two triggers are implemented in the same way but they will execute after inserting into Out_Transactions_Products. One of them is to decrease the product amount, and the other one is to calculate total price for the Out_Transaction.

2- Application Development:

2.1 – Class Diagram:

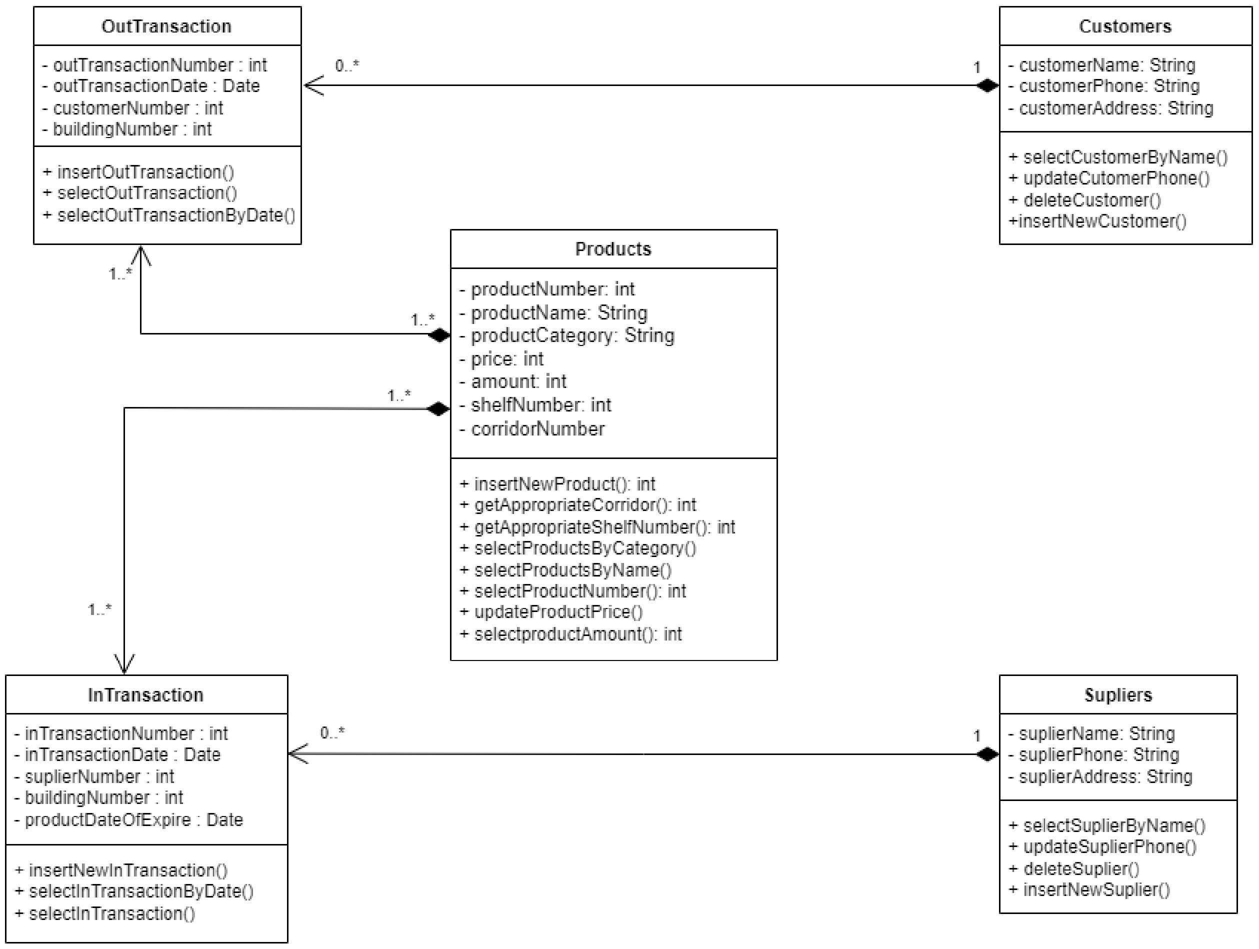


Figure 2-1 - Class Diagram

2.2 – Use-Case Diagram:

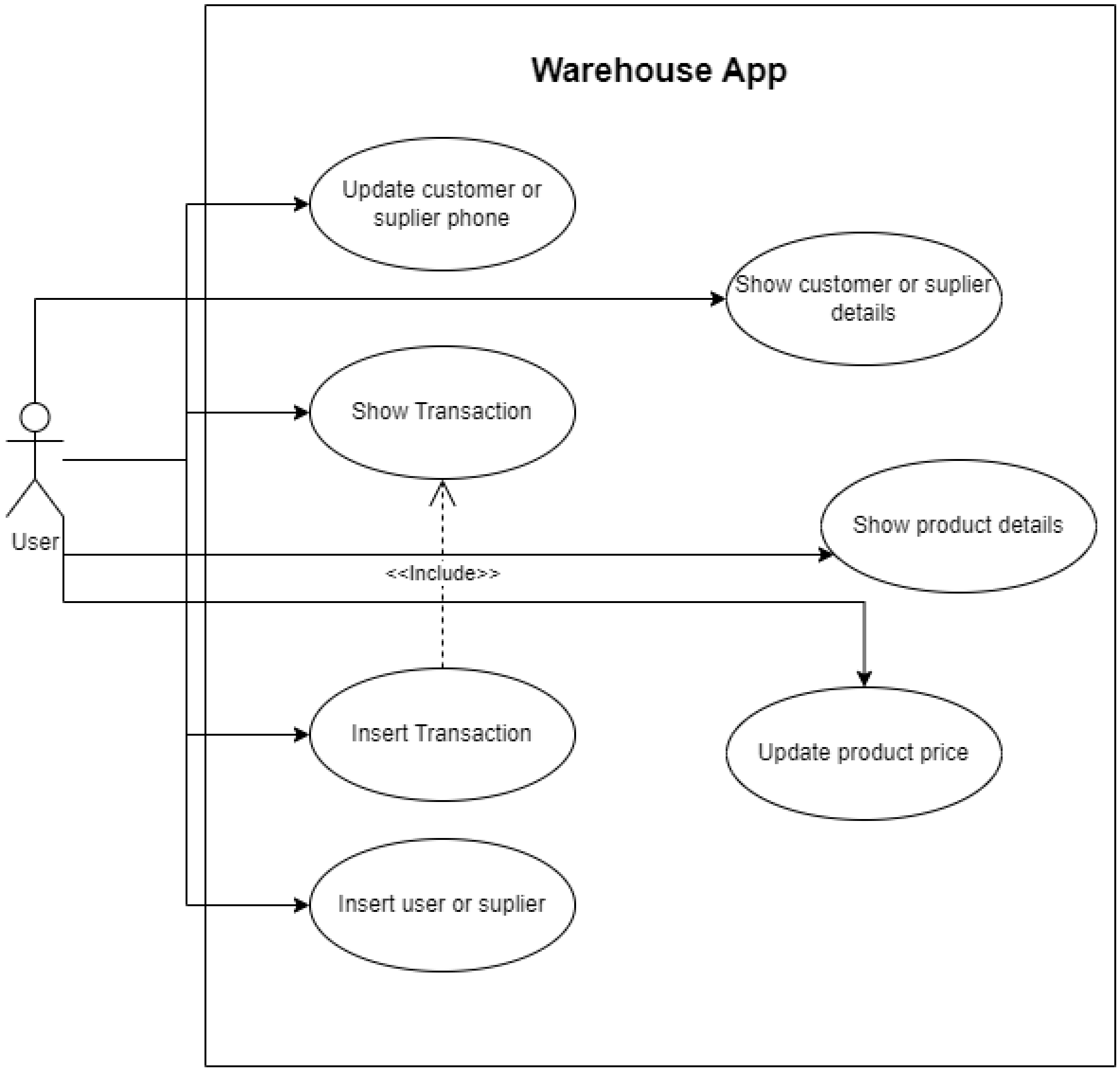


Figure 2-2 - Use-Case Diagram

2.3 – Code Implementation:

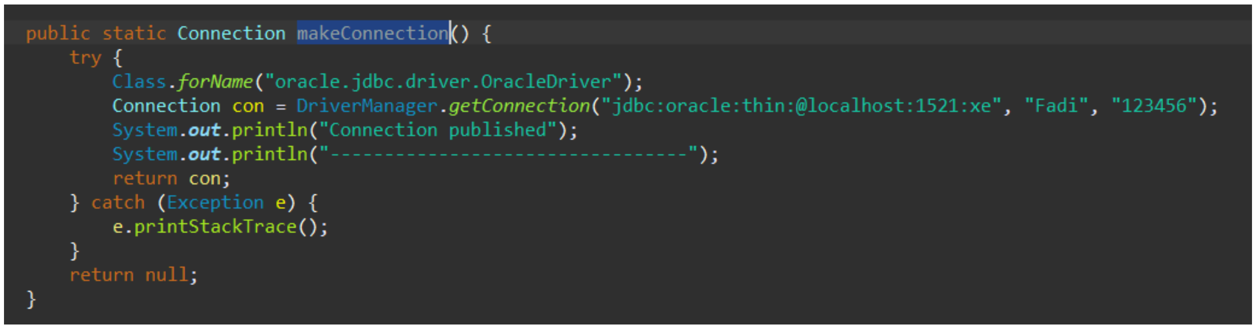
I used Java programing language with JDBC to implement this application.

2.3.1 – Database Connection Class:

I created a class to create connection with the database and close it after finishing.

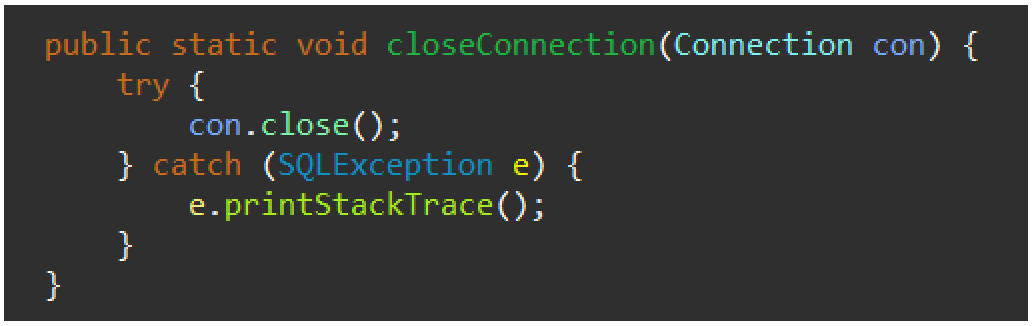
The functions in this class are static so I can call it without making an instance of the class.

- First function is makeConnection(). In this function I choose the driver manager, then I get the connection by sending the database url, username, and password. After that I return the connection.



```
public static Connection makeConnection() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "Fadi", "123456");
        System.out.println("Connection published");
        System.out.println("---------------------------------");
        return con;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

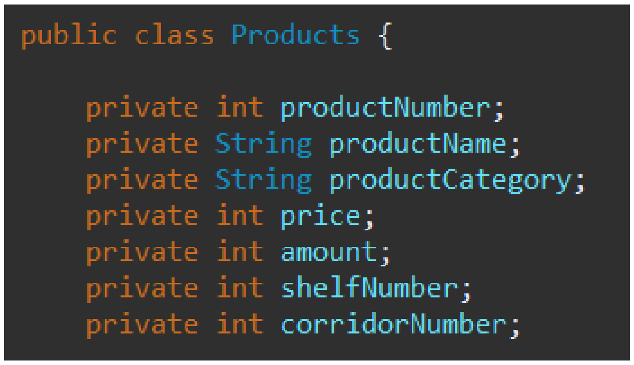
- There is another function called closeConnection(), that I called in the end of the program to close the connection with the database.



```
public static void closeConnection(Connection con) {
    try {
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

2.3.2 – Product Class:

In this class I declare these attributes:



```
public class Products {

    private int productNumber;
    private String productName;
    private String productCategory;
    private int price;
    private int amount;
    private int shelfNumber;
    private int corridorNumber;
```

Then I started with coding the functions I need:

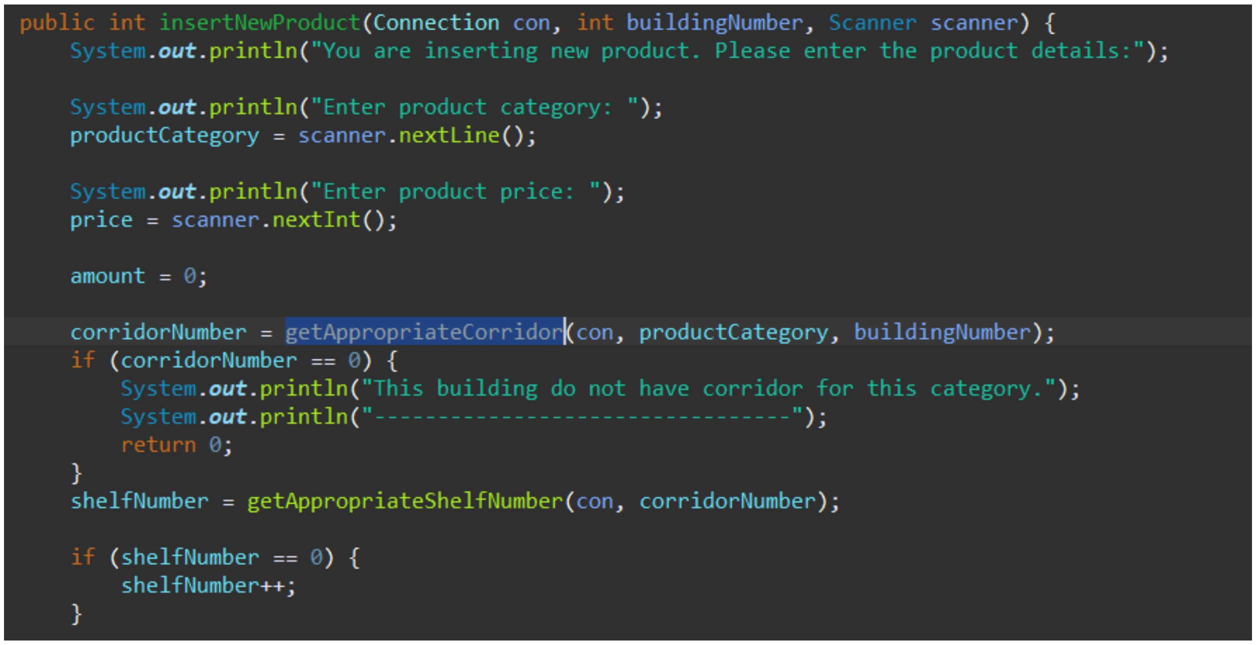
- insertNewProduct(): This function is to insert new product to the Products table in the database.

I asked the user to enter the new product details like the product category, and price.

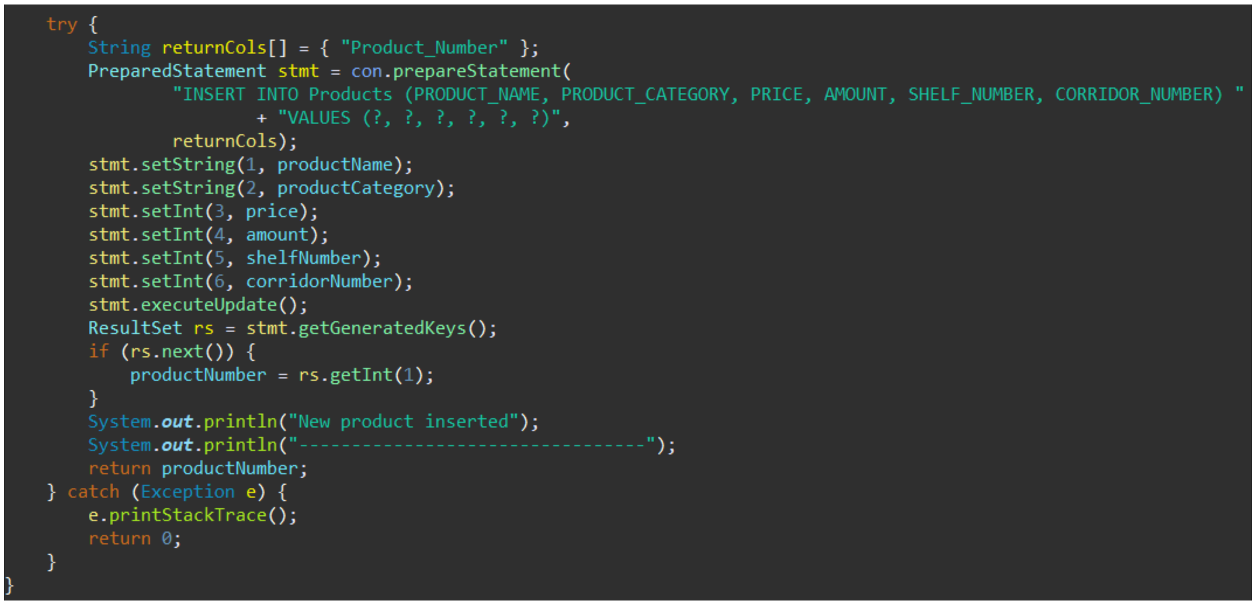
I assigned the amount of the product to 0 because this function will be called only when the user wants to make new In Transaction and add a product name that is not found in the chosen building products, and after that the amount will change depending on the amount that the user will add in the In Transaction using the trigger I mentioned before.

The program will determine in which shelf and corridor should we put this new product by calling the getAppropriateCorridorNumber(), and getAppropriateShelfNumber() functions which I will explain them later.

The code is shown below:

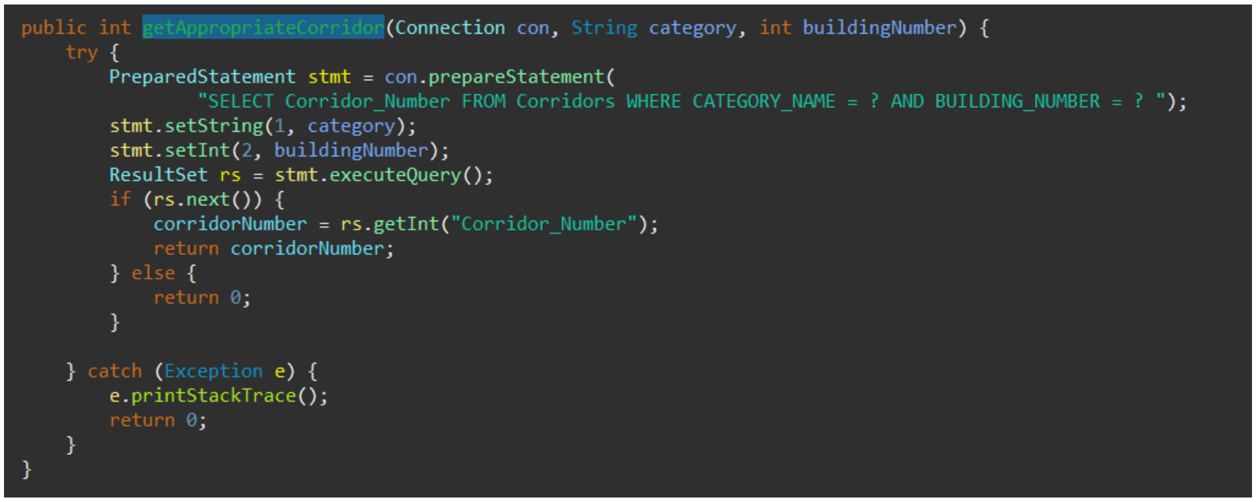


```java
public int insertNewProduct(Connection con, int buildingNumber, Scanner scanner) {
    System.out.println("You are inserting new product. Please enter the product details:");

    System.out.println("Enter product category: ");
    productCategory = scanner.nextLine();

    System.out.println("Enter product price: ");
    price = scanner.nextInt();

    amount = 0;

    corridorNumber = getAppropriateCorridor(con, productCategory, buildingNumber);
    if (corridorNumber == 0) {
        System.out.println("This building do not have corridor for this category.");
        System.out.println("--------------------------------");
        return 0;
    }
    shelfNumber = getAppropriateShelfNumber(con, corridorNumber);

    if (shelfNumber == 0) {
        shelfNumber++;
    }
```



```java
    try {
        String returnCols[] = { "Product_Number" };
        PreparedStatement stmt = con.prepareStatement(
                "INSERT INTO Products (PRODUCT_NAME, PRODUCT_CATEGORY, PRICE, AMOUNT, SHELF_NUMBER, CORRIDOR_NUMBER) "
                        + "VALUES (?, ?, ?, ?, ?, ?)",
                returnCols);
        stmt.setString(1, productName);
        stmt.setString(2, productCategory);
        stmt.setInt(3, price);
        stmt.setInt(4, amount);
        stmt.setInt(5, shelfNumber);
        stmt.setInt(6, corridorNumber);
        stmt.executeUpdate();
        ResultSet rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            productNumber = rs.getInt(1);
        }
        System.out.println("New product inserted");
        System.out.println("--------------------------------");
        return productNumber;
    } catch (Exception e) {
        e.printStackTrace();
        return 0;
    }
}
```

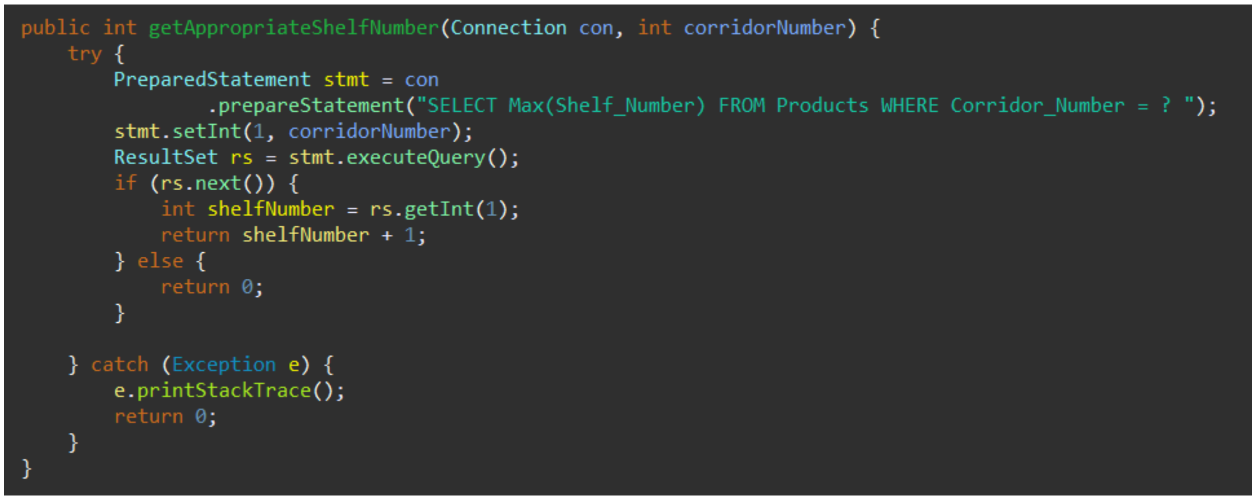
- getAppropriateCorridorNumber(): This function is to know which is the appropriate corridor number in the chosen building should we put the new product depending on the product category, because I divided the building to many corridors, each corridor has products from the same category. So, I just sent the product category, and the building number to the function, and the function will return the corridor number by selecting it from the database.



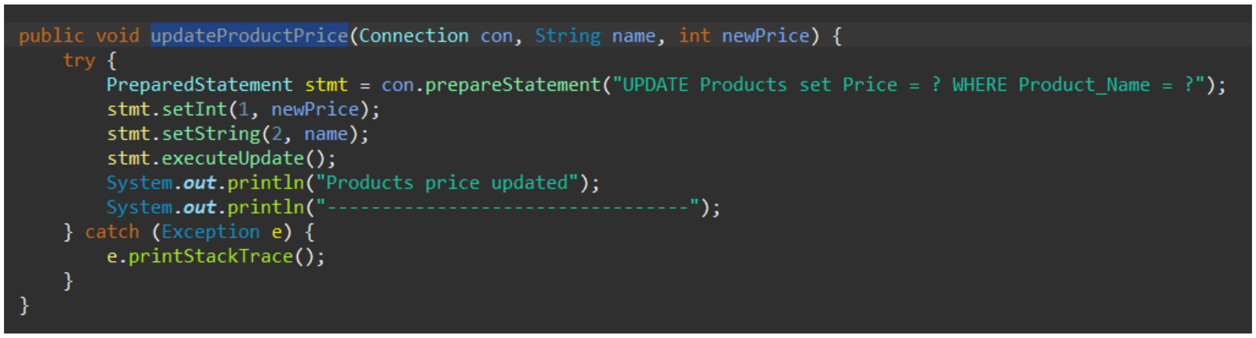
```java
public int getAppropriateCorridor(Connection con, String category, int buildingNumber) {
    try {
        PreparedStatement stmt = con.prepareStatement(
                "SELECT Corridor_Number FROM Corridors WHERE CATEGORY_NAME = ? AND BUILDING_NUMBER = ? ");
        stmt.setString(1, category);
        stmt.setInt(2, buildingNumber);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            corridorNumber = rs.getInt("Corridor_Number");
            return corridorNumber;
        } else {
            return 0;
        }

    } catch (Exception e) {
        e.printStackTrace();
        return 0;
    }
}
```

- getAppropriateShelfNumber(): I coded this function to select the last shelf in the corridor that has a product on it, because I organized the products in this way:

Easch product will be on a shelf started from shelf number 1.


```java
public int getAppropriateShelfNumber(Connection con, int corridorNumber) {
    try {
        PreparedStatement stmt = con
                .prepareStatement("SELECT Max(Shelf_Number) FROM Products WHERE Corridor_Number = ? ");
        stmt.setInt(1, corridorNumber);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            int shelfNumber = rs.getInt(1);
            return shelfNumber + 1;
        } else {
            return 0;
        }

    } catch (Exception e) {
        e.printStackTrace();
        return 0;
    }
}
```

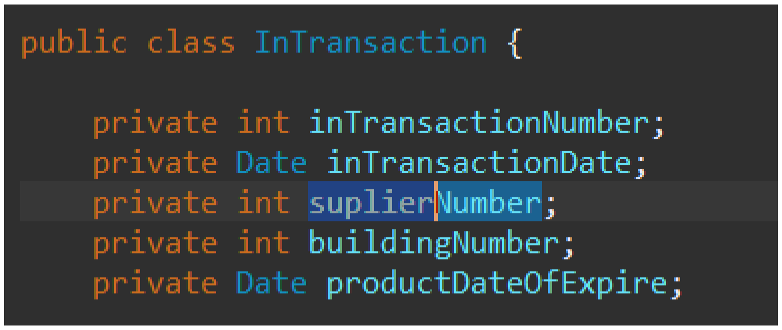
- updateProductPrice(): This function is to update product price by sending to it the product name and the new price, and it will update the price value in all buildings as shown.


```java
public void updateProductPrice(Connection con, String name, int newPrice) {
    try {
        PreparedStatement stmt = con.prepareStatement("UPDATE Products set Price = ? WHERE Product_Name = ?");
        stmt.setInt(1, newPrice);
        stmt.setString(2, name);
        stmt.executeUpdate();
        System.out.println("Products price updated");
        System.out.println("--------------------------------");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

And so on for the other functions in this class.

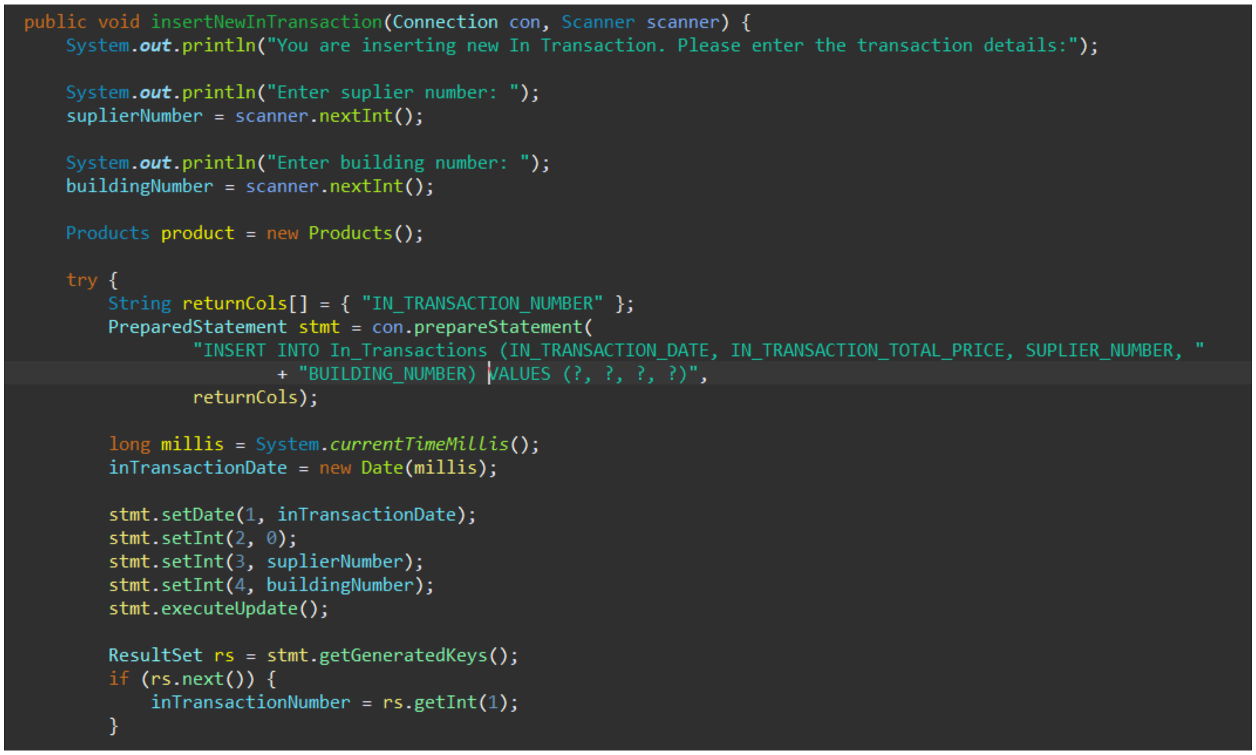
2.3.3 – In Transaction Class:

Firs of all I declare the class attributes as shown below.


```java
public class InTransaction {

    private int inTransactionNumber;
    private Date inTransactionDate;
    private int suplierNumber;
    private int buildingNumber;
    private Date productDateOfExpire;
```

Then I wrote the functions to call the sql commands that I need, like:

- insertNewTransaction(): This function is to insert new In Transaction by asking the user to enter the supplier number and building number, then the function will insert these values into In_Transactions table, the In_Transaction_Date will be the current date, and after finishing the insert, I get the new In_Transaction_Number by assigning the columns name I want to get back after inserting in a String array, and send it as a second parameter to the prepareStatment() function. Then I can get the value by using the ResultSet with the function getGeneratedKeys() that can be called by the PreparedStatment object.



```java
public void insertNewInTransaction(Connection con, Scanner scanner) {
    System.out.println("You are inserting new In Transaction. Please enter the transaction details:");

    System.out.println("Enter suplier number: ");
    suplierNumber = scanner.nextInt();

    System.out.println("Enter building number: ");
    buildingNumber = scanner.nextInt();

    Products product = new Products();

    try {
        String returnCols[] = { "IN_TRANSACTION_NUMBER" };
        PreparedStatement stmt = con.prepareStatement(
                "INSERT INTO In_Transactions (IN_TRANSACTION_DATE, IN_TRANSACTION_TOTAL_PRICE, SUPLIER_NUMBER, "
                        + "BUILDING_NUMBER) VALUES (?, ?, ?, ?)",
                returnCols);

        long millis = System.currentTimeMillis();
        inTransactionDate = new Date(millis);

        stmt.setDate(1, inTransactionDate);
        stmt.setInt(2, 0);
        stmt.setInt(3, suplierNumber);
        stmt.setInt(4, buildingNumber);
        stmt.executeUpdate();

        ResultSet rs = stmt.getGeneratedKeys();
        if (rs.next()) {
            inTransactionNumber = rs.getInt(1);
        }
```

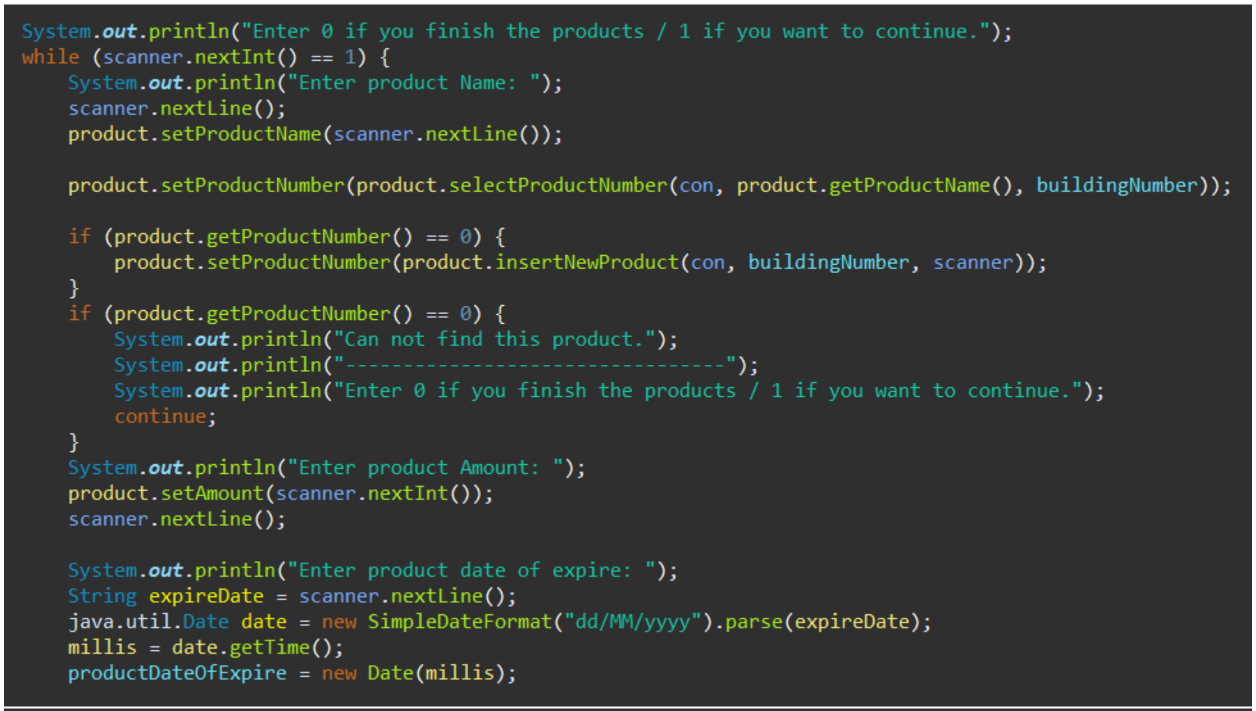
After that, the program will ask the user to enter the products one by one and it will check if the product is in the building or not by calling selectProductNumber() function from Products class and check the returned value.

If the returned value is 0, the program will call insertNewProduct() function which I explained before and then get the new product number and continue the execution.

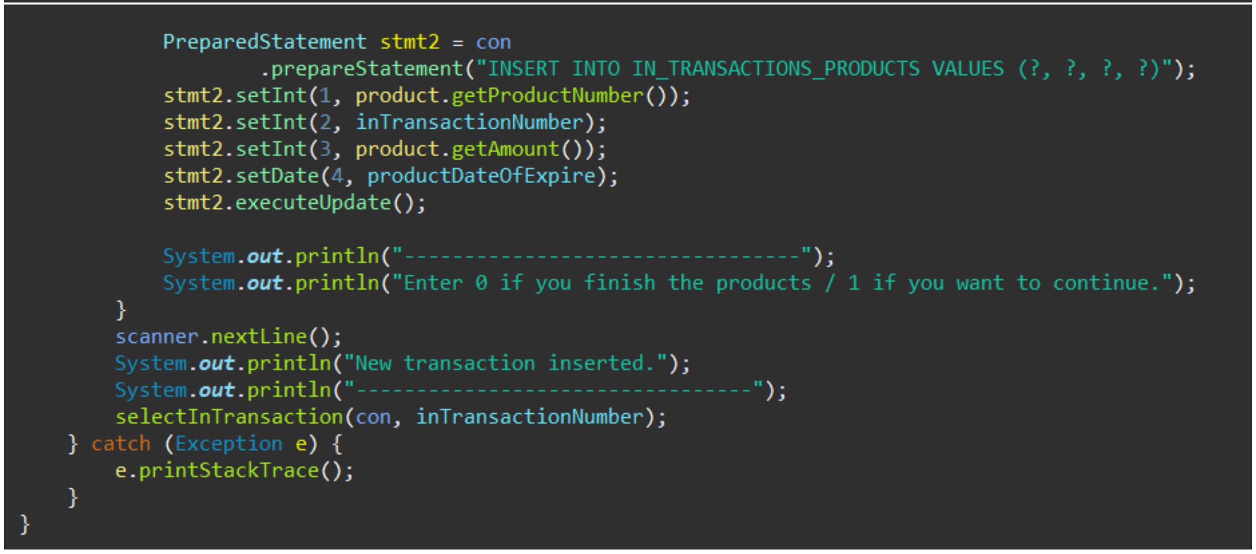
If the returned value is not 0, so we have got the product number and we can continue by inserting the remaining details for the new In_Transaction.

After each product details, the program will insert the values to the In_Transactions_Products table.

When the user finish entering products, the program will show him the In_Transaction bill.

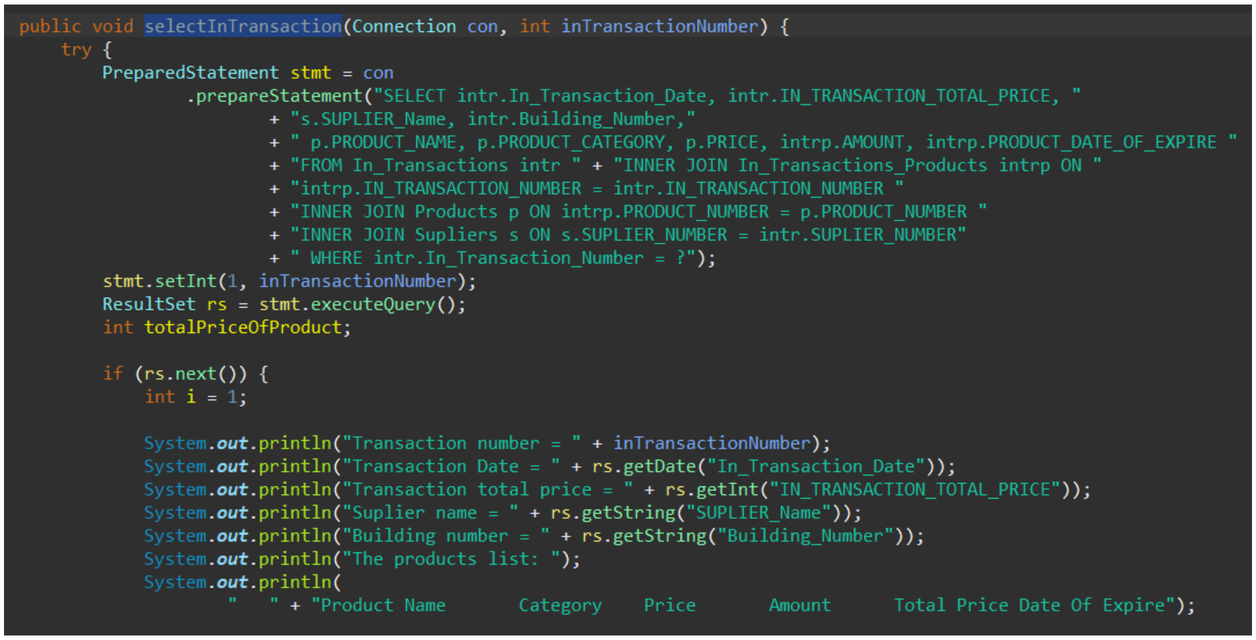


```java
System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
while (scanner.nextInt() == 1) {
    System.out.println("Enter product Name: ");
    scanner.nextLine();
    product.setProductName(scanner.nextLine());

    product.setProductNumber(product.selectProductNumber(con, product.getProductName(), buildingNumber));

    if (product.getProductNumber() == 0) {
        product.setProductNumber(product.insertNewProduct(con, buildingNumber, scanner));
    }
    if (product.getProductNumber() == 0) {
        System.out.println("Can not find this product.");
        System.out.println("--------------------------------");
        System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
        continue;
    }
    System.out.println("Enter product Amount: ");
    product.setAmount(scanner.nextInt());
    scanner.nextLine();

    System.out.println("Enter product date of expire: ");
    String expireDate = scanner.nextLine();
    java.util.Date date = new SimpleDateFormat("dd/MM/yyyy").parse(expireDate);
    millis = date.getTime();
    productDateOfExpire = new Date(millis);
```



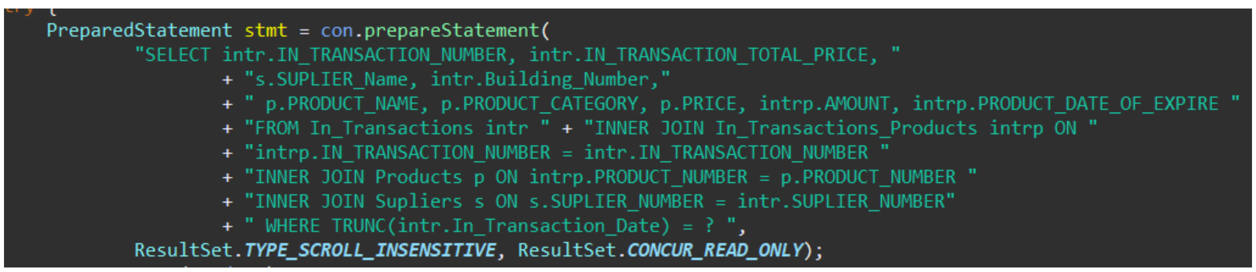
```java
            PreparedStatement stmt2 = con
                    .prepareStatement("INSERT INTO IN_TRANSACTIONS_PRODUCTS VALUES (?, ?, ?, ?)");
            stmt2.setInt(1, product.getProductNumber());
            stmt2.setInt(2, inTransactionNumber);
            stmt2.setInt(3, product.getAmount());
            stmt2.setDate(4, productDateOfExpire);
            stmt2.executeUpdate();

            System.out.println("--------------------------------");
            System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
        }
        scanner.nextLine();
        System.out.println("New transaction inserted.");
        System.out.println("--------------------------------");
        selectInTransaction(con, inTransactionNumber);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

- selectInTransaction(): This function will be executed after the insertNewTransaction() function finish. It will show the bill details by selecting the needed values from the following tables (In_Transactions, In_Transactions_Products, Products, and Supliers) using Inner Join between these tables.



```java
public void selectInTransaction(Connection con, int inTransactionNumber) {
    try {
        PreparedStatement stmt = con
                .prepareStatement("SELECT intr.In_Transaction_Date, intr.IN_TRANSACTION_TOTAL_PRICE, "
                        + "s.SUPLIER_Name, intr.Building_Number,"
                        + " p.PRODUCT_NAME, p.PRODUCT_CATEGORY, p.PRICE, intrp.AMOUNT, intrp.PRODUCT_DATE_OF_EXPIRE "
                        + "FROM In_Transactions intr " + "INNER JOIN In_Transactions_Products intrp ON "
                        + "intrp.IN_TRANSACTION_NUMBER = intr.IN_TRANSACTION_NUMBER "
                        + "INNER JOIN Products p ON intrp.PRODUCT_NUMBER = p.PRODUCT_NUMBER "
                        + "INNER JOIN Supliers s ON s.SUPLIER_NUMBER = intr.SUPLIER_NUMBER"
                        + " WHERE intr.In_Transaction_Number = ?");
        stmt.setInt(1, inTransactionNumber);
        ResultSet rs = stmt.executeQuery();
        int totalPriceOfProduct;

        if (rs.next()) {
            int i = 1;

            System.out.println("Transaction number = " + inTransactionNumber);
            System.out.println("Transaction Date = " + rs.getDate("In_Transaction_Date"));
            System.out.println("Transaction total price = " + rs.getInt("IN_TRANSACTION_TOTAL_PRICE"));
            System.out.println("Suplier name = " + rs.getString("SUPLIER_Name"));
            System.out.println("Building number = " + rs.getString("Building_Number"));
            System.out.println("The products list: ");
            System.out.println(
                    "    " + "Product Name       Category    Price       Amount      Total Price Date Of Expire");
```

- selectInTransactionByDate(): It is the same as the previous function, but here I sent to the PreparedStatement these parameters so I can use ResultSet.previous() function.



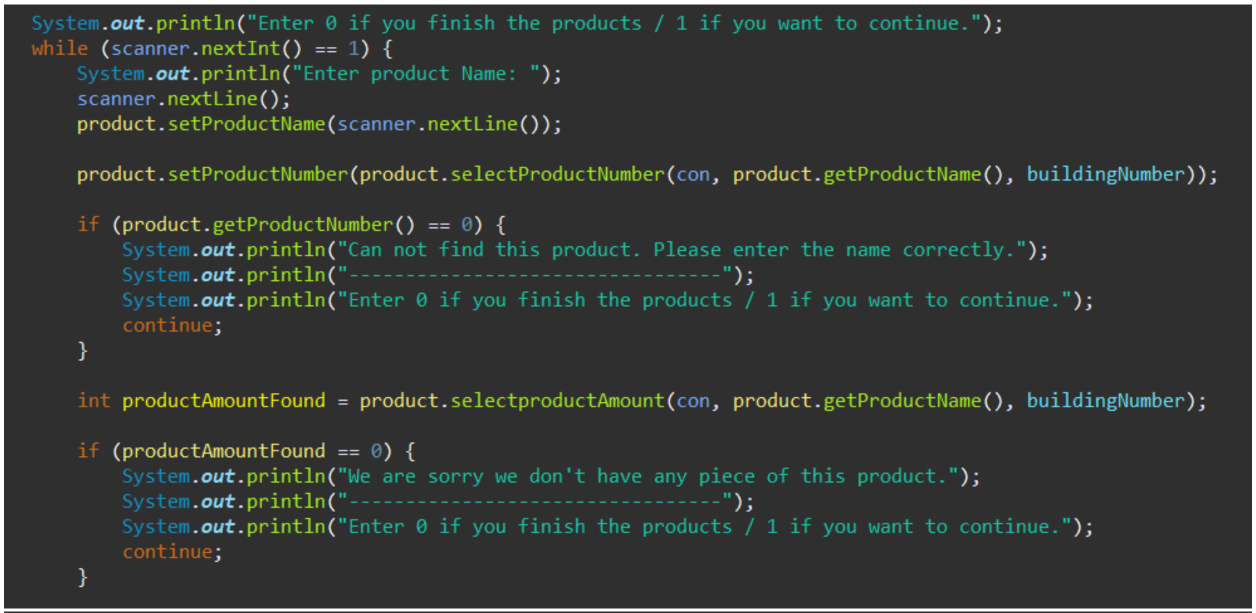
```java
    PreparedStatement stmt = con.prepareStatement(
            "SELECT intr.IN_TRANSACTION_NUMBER, intr.IN_TRANSACTION_TOTAL_PRICE, "
                    + "s.SUPLIER_Name, intr.Building_Number,"
                    + " p.PRODUCT_NAME, p.PRODUCT_CATEGORY, p.PRICE, intrp.AMOUNT, intrp.PRODUCT_DATE_OF_EXPIRE "
                    + "FROM In_Transactions intr " + "INNER JOIN In_Transactions_Products intrp ON "
                    + "intrp.IN_TRANSACTION_NUMBER = intr.IN_TRANSACTION_NUMBER "
                    + "INNER JOIN Products p ON intrp.PRODUCT_NUMBER = p.PRODUCT_NUMBER "
                    + "INNER JOIN Supliers s ON s.SUPLIER_NUMBER = intr.SUPLIER_NUMBER"
                    + " WHERE TRUNC(intr.In_Transaction_Date) = ? ",
            ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
```

2.3.4 – Out Transaction Class:

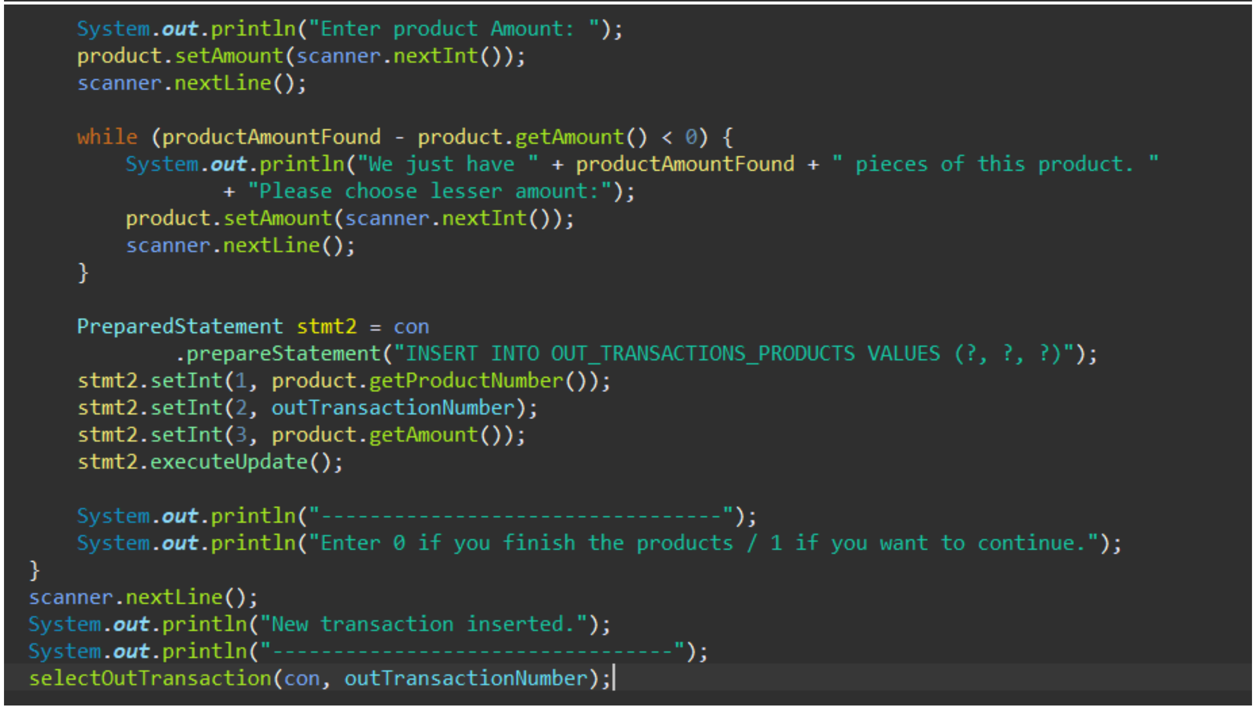
It is the same as the In Transaction class, but when the user wants to insert new Out Transaction, the function will check if the entered product is in the building or not.

If the entered product does not exist in the building the function will ask the user to choose another product.

If the product is in the building the function will compare the amount of the product in the building with the amount that the user entered. If the user entered an amount bigger than the amount found in the building, the program will tell the user to choose lesser amount, and the function will continue the execution.



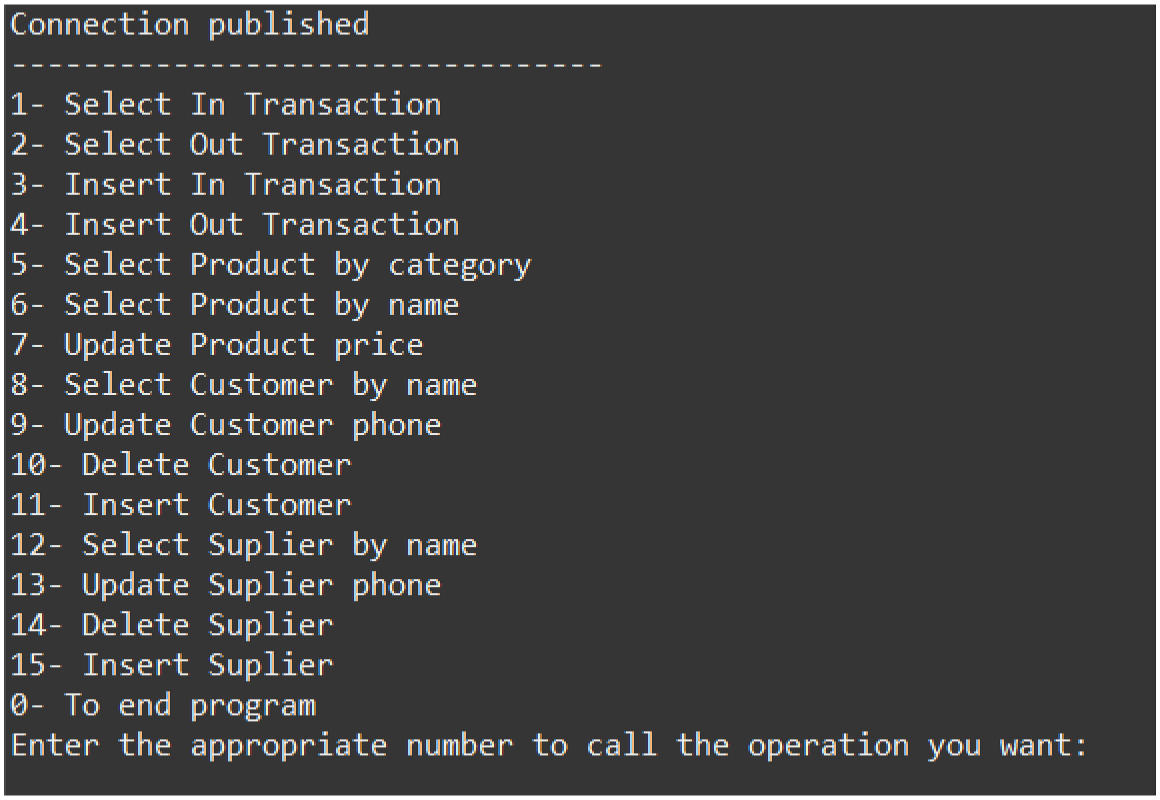
```
System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
while (scanner.nextInt() == 1) {
    System.out.println("Enter product Name: ");
    scanner.nextLine();
    product.setProductName(scanner.nextLine());

    product.setProductNumber(product.selectProductNumber(con, product.getProductName(), buildingNumber));

    if (product.getProductNumber() == 0) {
        System.out.println("Can not find this product. Please enter the name correctly.");
        System.out.println("---------------------------------");
        System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
        continue;
    }

    int productAmountFound = product.selectproductAmount(con, product.getProductName(), buildingNumber);

    if (productAmountFound == 0) {
        System.out.println("We are sorry we don't have any piece of this product.");
        System.out.println("---------------------------------");
        System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
        continue;
    }
```



```
    System.out.println("Enter product Amount: ");
    product.setAmount(scanner.nextInt());
    scanner.nextLine();

    while (productAmountFound - product.getAmount() < 0) {
        System.out.println("We just have " + productAmountFound + " pieces of this product. "
                + "Please choose lesser amount:");
        product.setAmount(scanner.nextInt());
        scanner.nextLine();
    }

    PreparedStatement stmt2 = con
            .prepareStatement("INSERT INTO OUT_TRANSACTIONS_PRODUCTS VALUES (?, ?, ?)");
    stmt2.setInt(1, product.getProductNumber());
    stmt2.setInt(2, outTransactionNumber);
    stmt2.setInt(3, product.getAmount());
    stmt2.executeUpdate();

    System.out.println("---------------------------------");
    System.out.println("Enter 0 if you finish the products / 1 if you want to continue.");
}
scanner.nextLine();
System.out.println("New transaction inserted.");
System.out.println("---------------------------------");
selectOutTransaction(con, outTransactionNumber);
```

3 – Testing:

This is the first thing which will appear to the user.

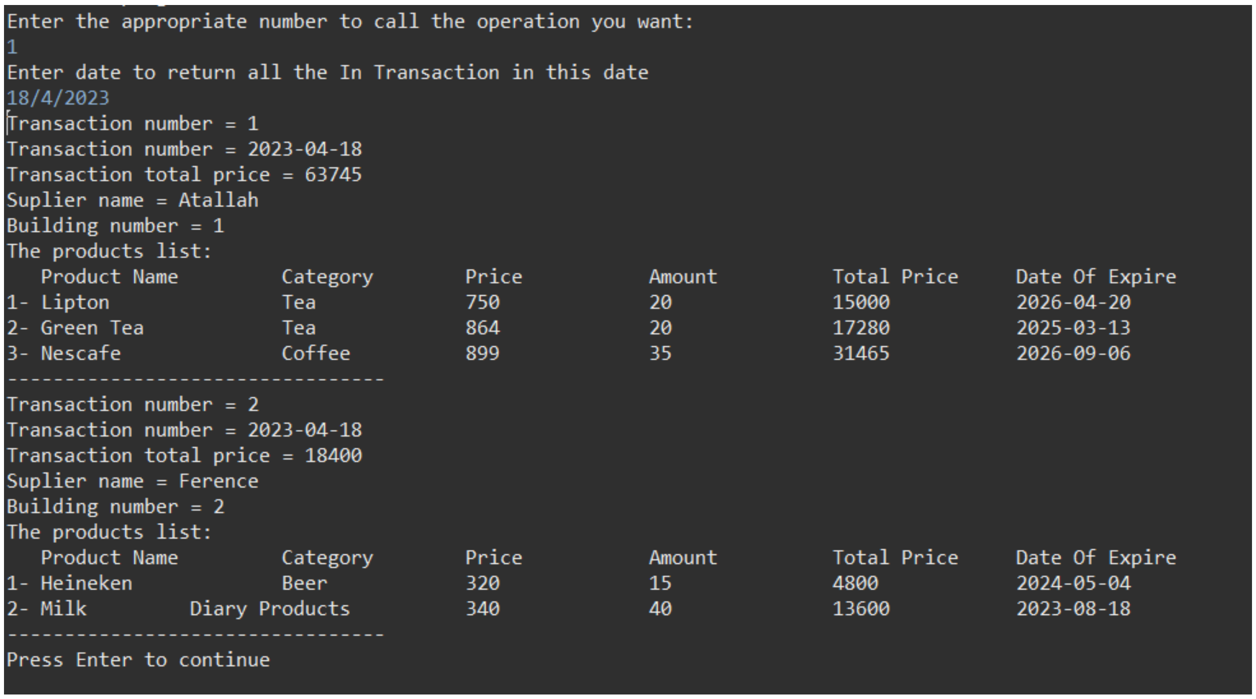


```
Connection published
---------------------------------
1- Select In Transaction
2- Select Out Transaction
3- Insert In Transaction
4- Insert Out Transaction
5- Select Product by category
6- Select Product by name
7- Update Product price
8- Select Customer by name
9- Update Customer phone
10- Delete Customer
11- Insert Customer
12- Select Suplier by name
13- Update Suplier phone
14- Delete Suplier
15- Insert Suplier
0- To end program
Enter the appropriate number to call the operation you want:
```

The user can choose any operation by enter the number shown next to it.

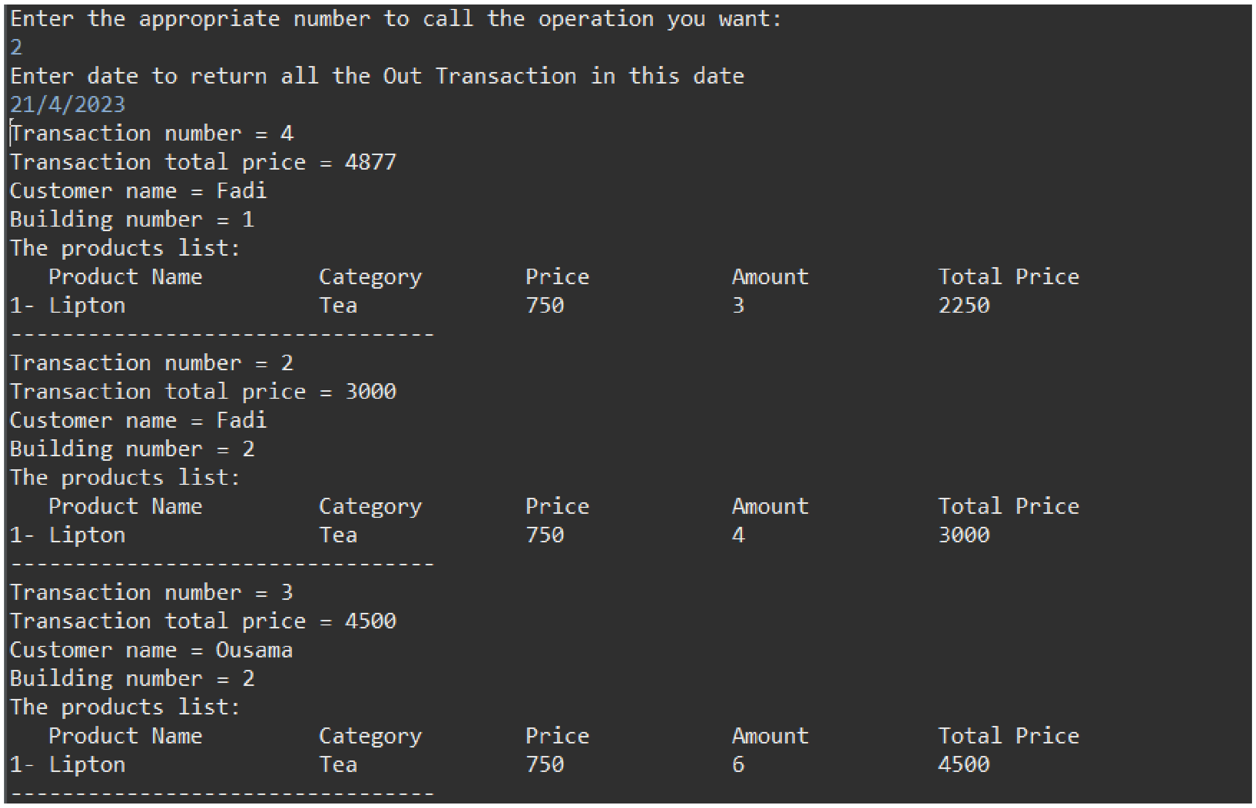
For example, if the user enters 1, the program will ask him to enter the date which he wants to see all the in transactions done in this date.

The result will be ass shown.

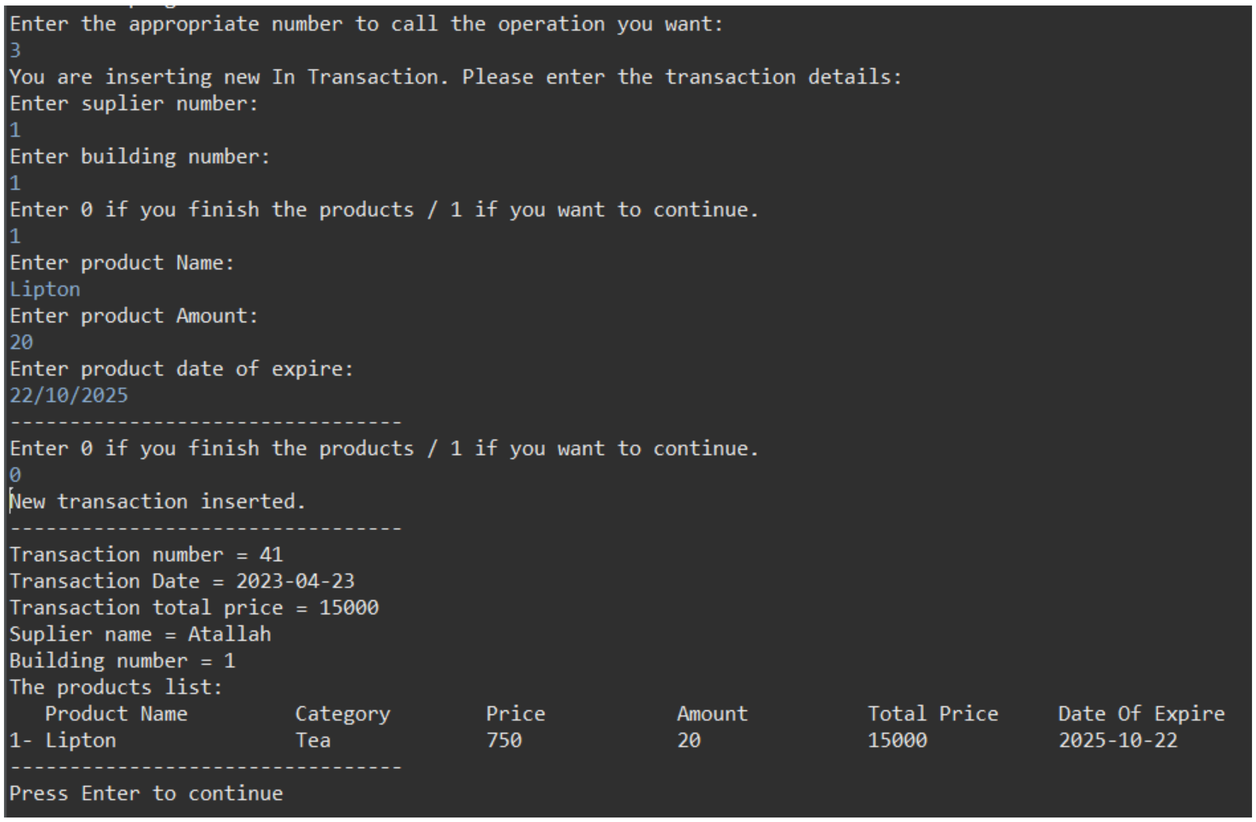

```
Enter the appropriate number to call the operation you want:
1
Enter date to return all the In Transaction in this date
18/4/2023
Transaction number = 1
Transaction number = 2023-04-18
Transaction total price = 63745
Suplier name = Atallah
Building number = 1
The products list:
   Product Name         Category        Price           Amount          Total Price     Date Of Expire
1- Lipton               Tea             750             20              15000           2026-04-20
2- Green Tea            Tea             864             20              17280           2025-03-13
3- Nescafe              Coffee          899             35              31465           2026-09-06
---------------------------------
Transaction number = 2
Transaction number = 2023-04-18
Transaction total price = 18400
Suplier name = Ference
Building number = 2
The products list:
   Product Name         Category        Price           Amount          Total Price     Date Of Expire
1- Heineken             Beer            320             15              4800            2024-05-04
2- Milk         Diary Products          340             40              13600           2023-08-18
---------------------------------
Press Enter to continue
```

The program will ask the user again about what he wants to do.

- This is the result if the user enters 2:


```
Enter the appropriate number to call the operation you want:
2
Enter date to return all the Out Transaction in this date
21/4/2023
Transaction number = 4
Transaction total price = 4877
Customer name = Fadi
Building number = 1
The products list:
   Product Name         Category        Price           Amount          Total Price
1- Lipton               Tea             750             3               2250
---------------------------------
Transaction number = 2
Transaction total price = 3000
Customer name = Fadi
Building number = 2
The products list:
   Product Name         Category        Price           Amount          Total Price
1- Lipton               Tea             750             4               3000
---------------------------------
Transaction number = 3
Transaction total price = 4500
Customer name = Ousama
Building number = 2
The products list:
   Product Name         Category        Price           Amount          Total Price
1- Lipton               Tea             750             6               4500
---------------------------------
```

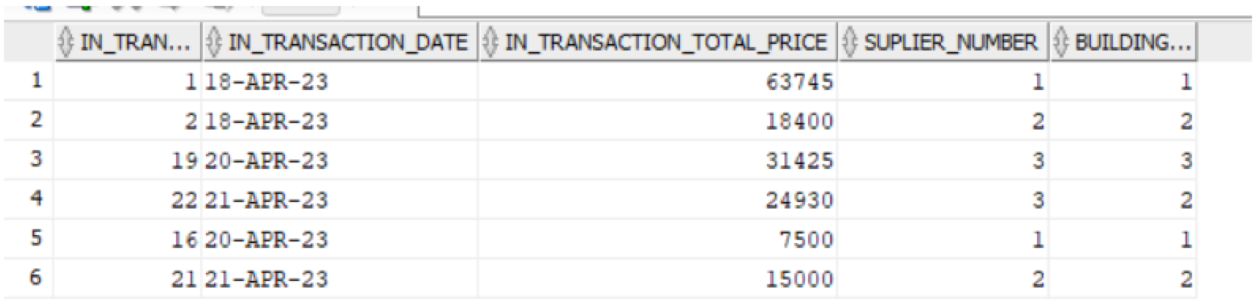
- If the user wants to insert new in transaction:



```
Enter the appropriate number to call the operation you want:
3
You are inserting new In Transaction. Please enter the transaction details:
Enter suplier number:
1
Enter building number:
1
Enter 0 if you finish the products / 1 if you want to continue.
1
Enter product Name:
Lipton
Enter product Amount:
20
Enter product date of expire:
22/10/2025
--------------------------------
Enter 0 if you finish the products / 1 if you want to continue.
0
New transaction inserted.
--------------------------------
Transaction number = 41
Transaction Date = 2023-04-23
Transaction total price = 15000
Suplier name = Atallah
Building number = 1
The products list:
   Product Name         Category        Price           Amount          Total Price     Date Of Expire
1- Lipton               Tea             750             20              15000           2025-10-22
--------------------------------
Press Enter to continue
```

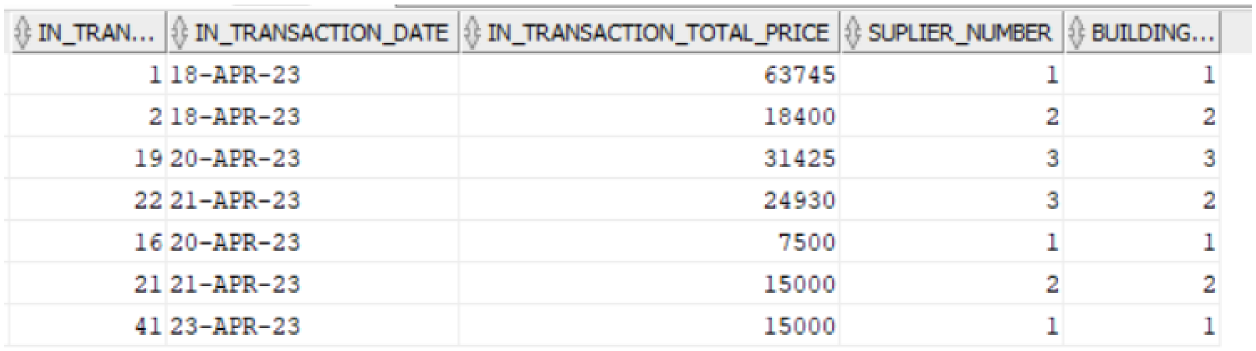
This is the database values before and after the insert:

In_Transactions table before inserting:



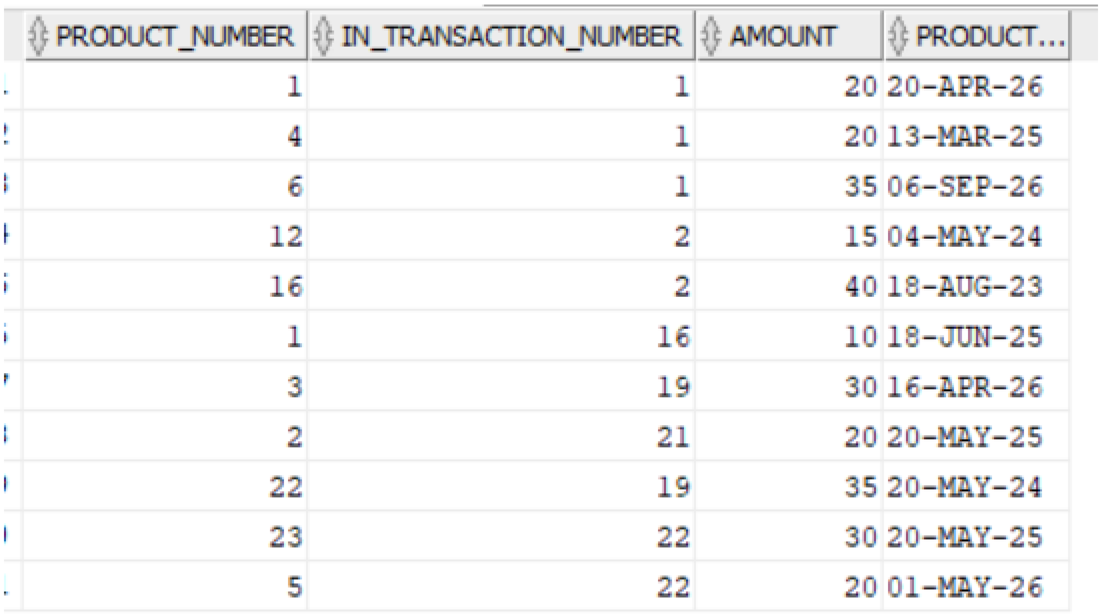
	IN_TRAN...	IN_TRANSACTION_DATE	IN_TRANSACTION_TOTAL_PRICE	SUPLIER_NUMBER	BUILDING...
1	1	18-APR-23	63745	1	1
2	2	18-APR-23	18400	2	2
3	19	20-APR-23	31425	3	3
4	22	21-APR-23	24930	3	2
5	16	20-APR-23	7500	1	1
6	21	21-APR-23	15000	2	2

In_Transactions table after inserting:



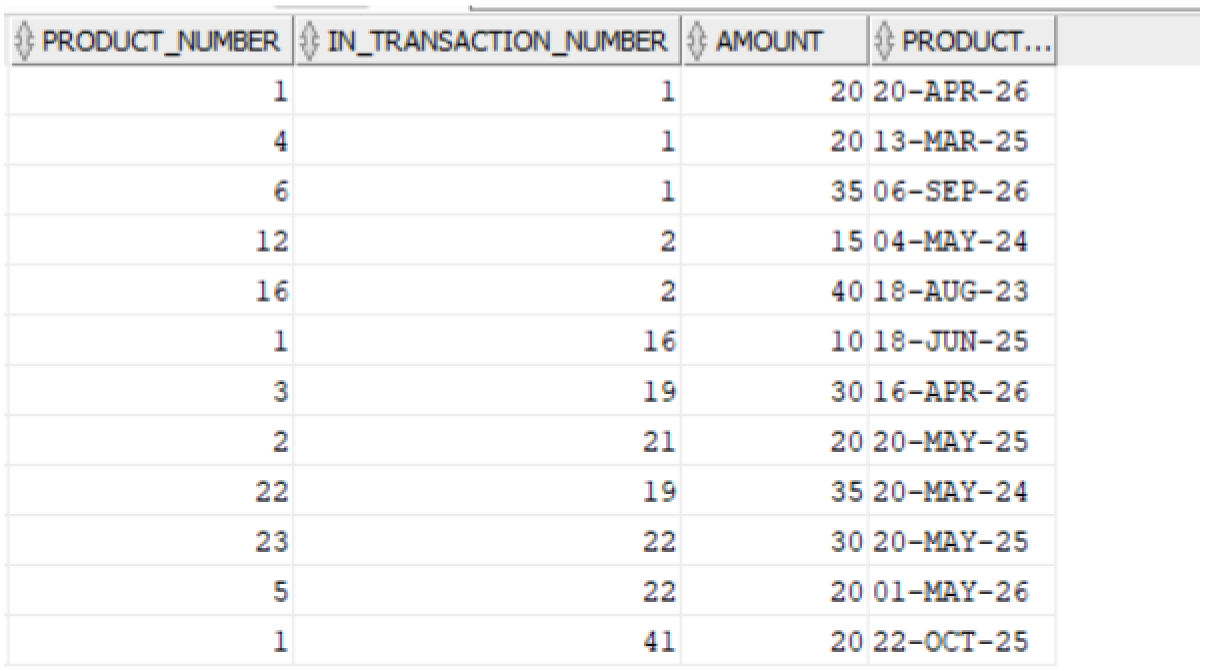
IN_TRAN...	IN_TRANSACTION_DATE	IN_TRANSACTION_TOTAL_PRICE	SUPLIER_NUMBER	BUILDING...
1	18-APR-23	63745	1	1
2	18-APR-23	18400	2	2
19	20-APR-23	31425	3	3
22	21-APR-23	24930	3	2
16	20-APR-23	7500	1	1
21	21-APR-23	15000	2	2
41	23-APR-23	15000	1	1

In_Transactions_Products table before inserting:



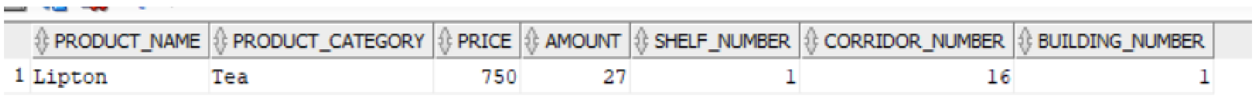
PRODUCT_NUMBER	IN_TRANSACTION_NUMBER	AMOUNT	PRODUCT...
1	1	20	20-APR-26
4	1	20	13-MAR-25
6	1	35	06-SEP-26
12	2	15	04-MAY-24
16	2	40	18-AUG-23
1	16	10	18-JUN-25
3	19	30	16-APR-26
2	21	20	20-MAY-25
22	19	35	20-MAY-24
23	22	30	20-MAY-25
5	22	20	01-MAY-26

In_Transactions_Products table after inserting:



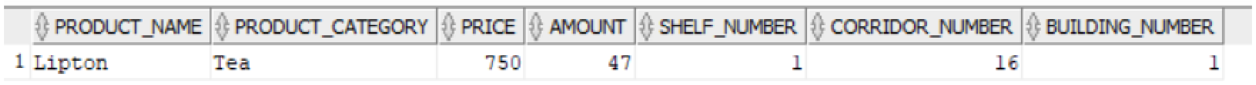
PRODUCT_NUMBER	IN_TRANSACTION_NUMBER	AMOUNT	PRODUCT...
1	1	20	20-APR-26
4	1	20	13-MAR-25
6	1	35	06-SEP-26
12	2	15	04-MAY-24
16	2	40	18-AUG-23
1	16	10	18-JUN-25
3	19	30	16-APR-26
2	21	20	20-MAY-25
22	19	35	20-MAY-24
23	22	30	20-MAY-25
5	22	20	01-MAY-26
1	41	20	22-OCT-25

The product details before inserting:



	PRODUCT_NAME	PRODUCT_CATEGORY	PRICE	AMOUNT	SHELF_NUMBER	CORRIDOR_NUMBER	BUILDING_NUMBER
1	Lipton	Tea	750	27	1	16	1

The product details after inserting:



	PRODUCT_NAME	PRODUCT_CATEGORY	PRICE	AMOUNT	SHELF_NUMBER	CORRIDOR_NUMBER	BUILDING_NUMBER
1	Lipton	Tea	750	47	1	16	1