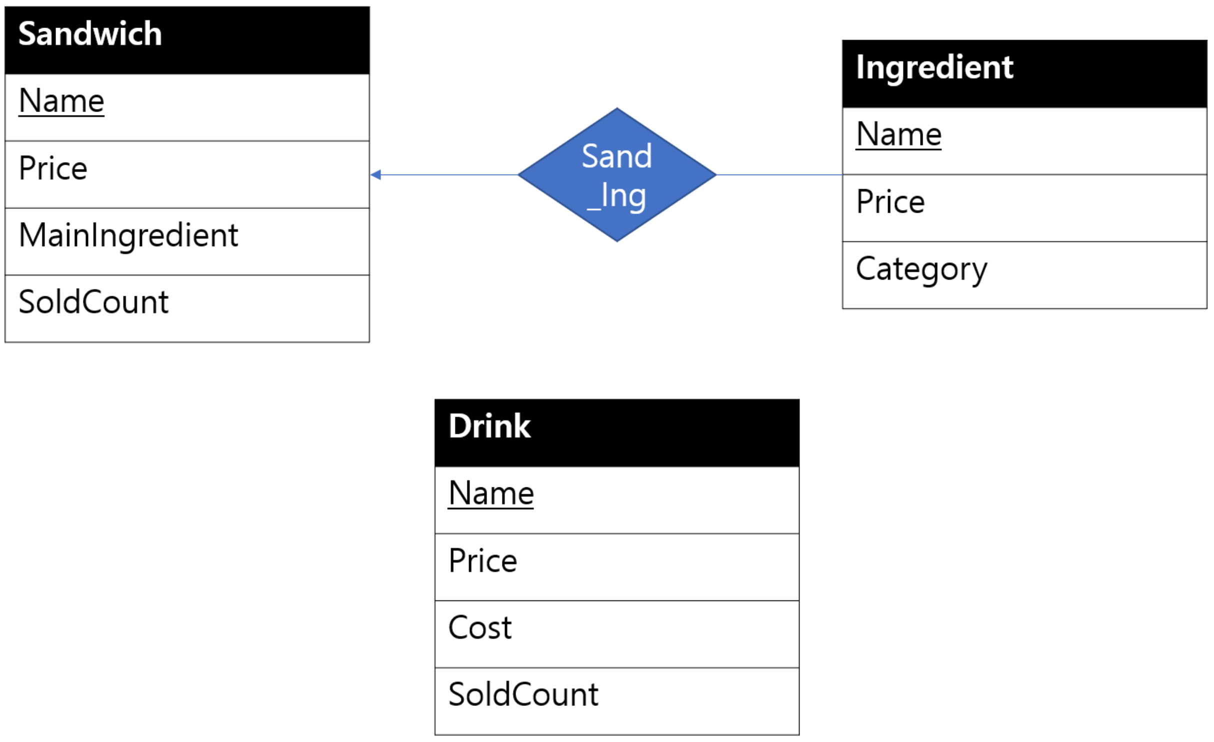
# Database Project

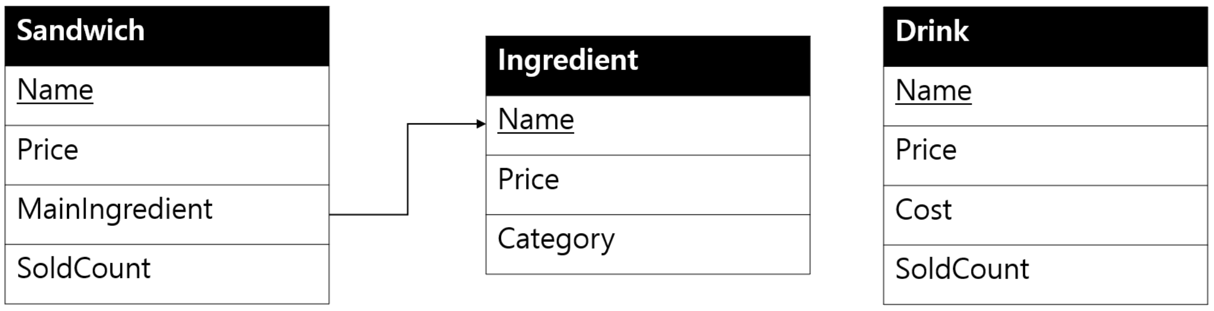
## Menu Management System for Sandwich Store

| | |
|---|---|
| 과목 | Database(02) |
| 이름 | 함초롬 |
| 학번 | 1876442 |
| 전공 | 컴퓨터공학전공 |

## 1. ER diagram



## 2. Database Schema diagram



(Underlined column name means a primary key. Arrow indicates foreign key constraints.)

## 3. Class and method explanation of Java codes

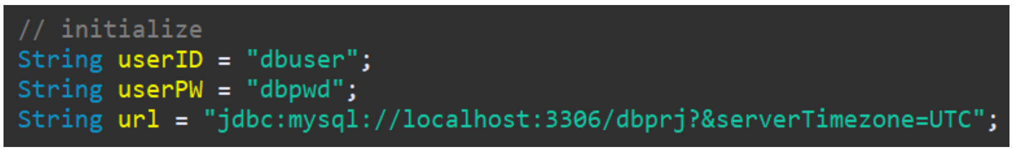
Everything that needs to be done in the project is implemented in the main method of class named 'MyDriver'. Detailed explanation is written in section 5.

## 4. Main class name and how to run, connection configuration instructions

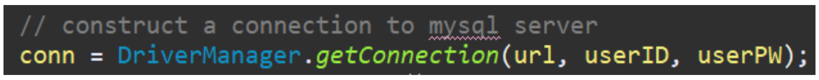
Main class name is 'MyDriver'.

Download the attached jar file(dbproject.jar) and run on CMD. On CMD, change directory where 'dbproject.jar' file is and run with command "java -jar dbproject.jar". Or download source files and run with IDE like Eclipse. There is only one file 'MyDriver'. You can run this file.

In this project, I used "localhost:3306" for host and port name and assigned "dbprj" for database name. I use "dbuser" for user id, "dbpwd" for password.



```java
// initialize
String userID = "dbuser";
String userPW = "dbpwd";
String url = "jdbc:mysql://localhost:3306/dbprj?&serverTimezone=UTC";
```



```java
// construct a connection to mysql server
conn = DriverManager.getConnection(url, userID, userPW);
```

## 5. Detailed Explanation about 16 Requirements

**[ DB Schema ]**

**(1) Should have at least 3 tables with each table having at least 3 columns.**

**The 3 tables should have at least 3 columns, and others may have any number of columns.**

**(3) Should include primary key in every table, and also foreign key, not null constraints should exist in some tables**

**(4) Tables should be in 3rd Normal Form (3NF)**

I made 3 tables having at least 3 columns as schema diagram. All tables are in 3rd Normal Form.

All tables have primary keys and not null constraints. Sandwich table has foreign key references primary key in Ingredient table.

Category divides into main and sub. All of sub categorized ingredients should be included in every sandwich.

code (createdb.sql)

```
#create table as schema design - (1), (3), (4)

CREATE TABLE Drink(
            Name VARCHAR(10) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            Cost NUMERIC(2,1),
            SoldCount INT,
            PRIMARY KEY (Name)
);

CREATE TABLE Ingredient(
            Name VARCHAR(20) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            Category VARCHAR(10) NOT NULL,
            PRIMARY KEY (Name)
);

CREATE TABLE Sandwich(
            Name VARCHAR(20) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            MainIngredient VARCHAR(20) NOT NULL,
            SoldCount INT,
            PRIMARY KEY (Name),
            FOREIGN KEY (MainIngredient) REFERENCES Ingredient(Name)
);
```

**(2) Should have at least 30 records inserted for initialization (total records for all tables)**

**For example, A table may have 5 records, B table 10 records, and C table 15 records**

I initialized with 30(at total) records. I inserted data with INSERT and UPDATE queries.

code (createdb.sql)

```
#insert data into tables - (2)

INSERT INTO Drink (Name, Price) values ('Coke', 2);
INSERT INTO Drink (Name, Price) values ('Sprite', 2);
INSERT INTO Drink (Name, Price) values ('Juice', 4);
INSERT INTO Drink (Name, Price) values ('Water', 1);
INSERT INTO Drink (Name, Price) values ('Coffee', 3);
INSERT INTO Drink (Name, Price) values ('Milk', 2);
UPDATE Drink SET SoldCount = 0;
UPDATE Drink SET Cost = Price * 0.5;

INSERT INTO Ingredient (Name, Price, Category) values ('egg', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('beef', 3, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('chicken', 2, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('ham', 0.8, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('avocado', 1.5, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('turkey', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('tuna', 0.7, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('bacon', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('meatball', 1.2, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('shrimp', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('bread', 0.3, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('vegetable', 0.7, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('cheese', 0.5, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('sauce', 0.1, 'sub');

INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Egg', 5.5, 'egg');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Roasted Beef', 9, 'beef');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Roasted Chicken', 8, 'chicken');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Ham', 6, 'ham');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Avocado', 7, 'avocado');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Turkey', 6.5, 'turkey');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Tuna', 6.3, 'tuna');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Bacon', 6.8, 'bacon');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Meatball', 7.5, 'meatball');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Shrimp', 7.5, 'shrimp');
UPDATE Sandwich SET SoldCount = 0;
```

**(5) At least 1 index should be defined on the tables (The primary key(PK) columns have index automatically created, so do not create an index on a PK)**

I created index on Sandwich table's SoldCount column and Drink table's SoldCount column. Because this is frequently used in the program.

code (createdb.sql)

```
#create index - (5)

CREATE INDEX SandwichSoldCountIndex ON Sandwich(SoldCount);
CREATE INDEX DrinkSoldCountIndex ON Drink(SoldCount);
```

**(6) 1 view should be defined, and the view should be defined using at least two other tables**

The program was implemented to calculate the net profit. To facilitate this net profit calculation, a view is created as shown below.

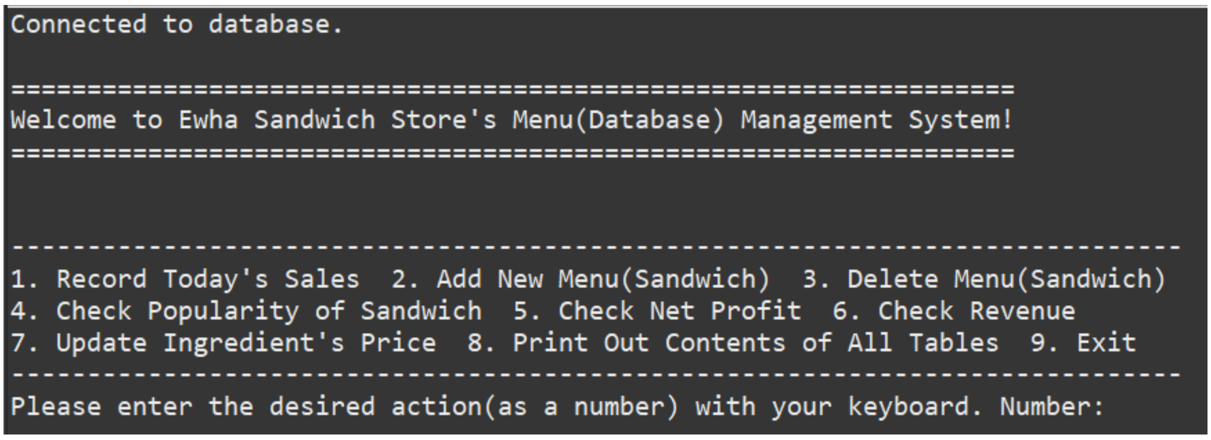
Net profit of sandwich = (sandwich's price – sum of ingredients' price) * (number sold)

Ingredients for sandwich = (all of sub categorized ingredients + main ingredient of the menu) code (createdb.sql)

```
#create view - (6)

CREATE VIEW DefaultCost AS
SELECT sum(Price) as Cost
FROM Ingredient
WHERE Category='sub';

CREATE VIEW Profit AS
SELECT (Sandwich.Name) AS Name, (Sandwich.Price - Ingredient.Price - DefaultCost.Cost) as Profit
FROM Sandwich, Ingredient, DefaultCost
WHERE Sandwich.MainIngredient = Ingredient.Name;
```

**[ DB Queries and Program ]**

- Basic screen at program start

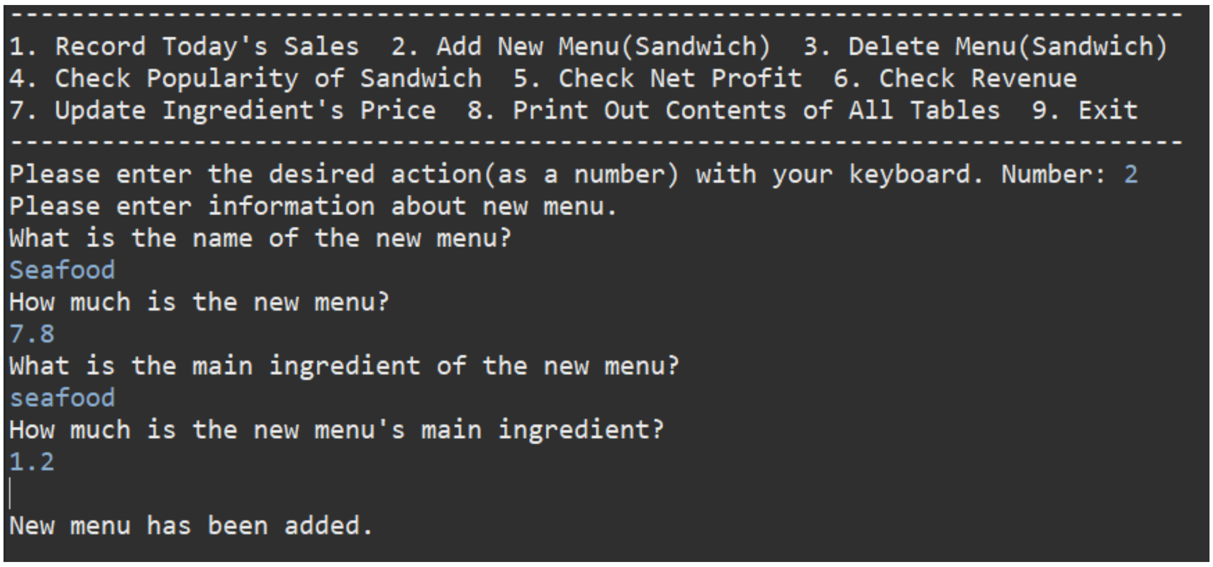


```
Connected to database.

=================================================================
Welcome to Ewha Sandwich Store's Menu(Database) Management System!
=================================================================



----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number:
```

**(7) All queries (in 8 to 14 below) should have parameterized variables.**

**(8) Should have at least 1 interface (menu and user input) and query to insert into 1 table**

[2. Add New Menu(Sandwich)] is interface to insert data into table.



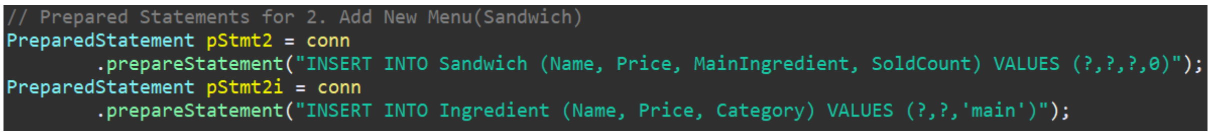
```
----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 2
Please enter information about new menu.
What is the name of the new menu?
Seafood
How much is the new menu?
7.8
What is the main ingredient of the new menu?
seafood
How much is the new menu's main ingredient?
1.2

New menu has been added.
```

When user wants to insert new sandwich menu, user can enter number 2 with keyboard.

The program asks for input value from the user and creates a query using the user input value. The program asks new menu's name, price, main ingredient and the price of main ingredient. After getting all this information from user, then insert these data into database with using prepared statement as below.

In the above example, the seafood menu and ingredients are inserted into the sandwich table and the ingredient table after receiving user input.

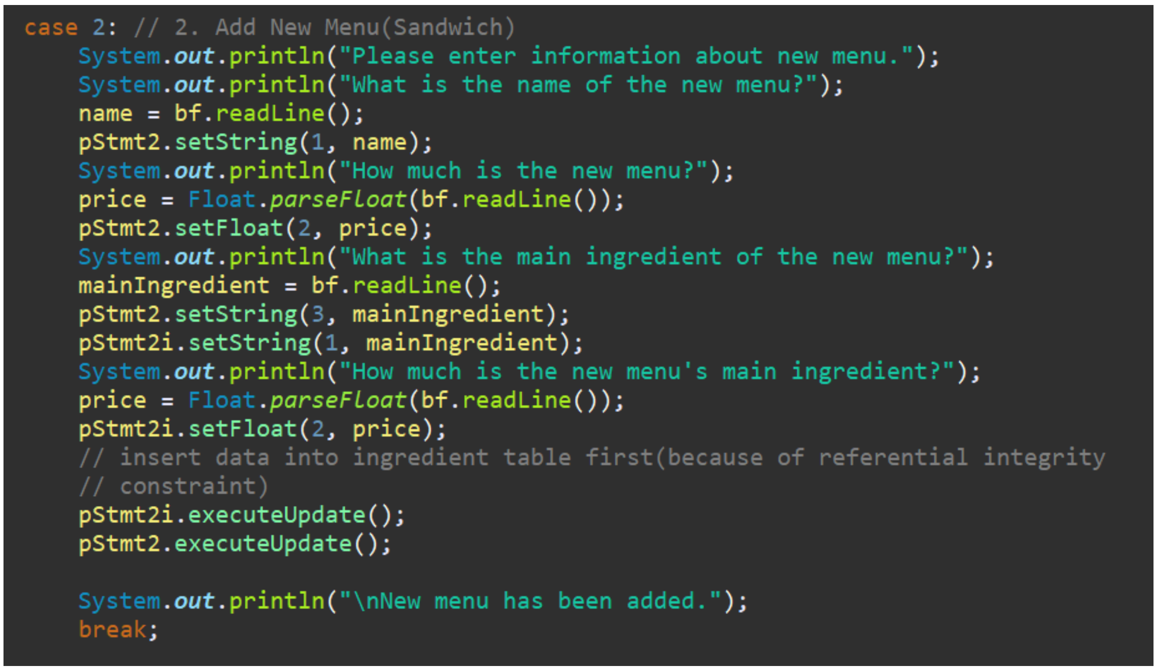
- Prepared statement



```
// Prepared Statements for 2. Add New Menu(Sandwich)
PreparedStatement pStmt2 = conn
        .prepareStatement("INSERT INTO Sandwich (Name, Price, MainIngredient, SoldCount) VALUES (?,?,?,0)");
PreparedStatement pStmt2i = conn
        .prepareStatement("INSERT INTO Ingredient (Name, Price, Category) VALUES (?,?,'main')");
```

- pStmt2 is used for insertion to Sandwich table.
- pStmt2i is used for insertion to Ingredient table.

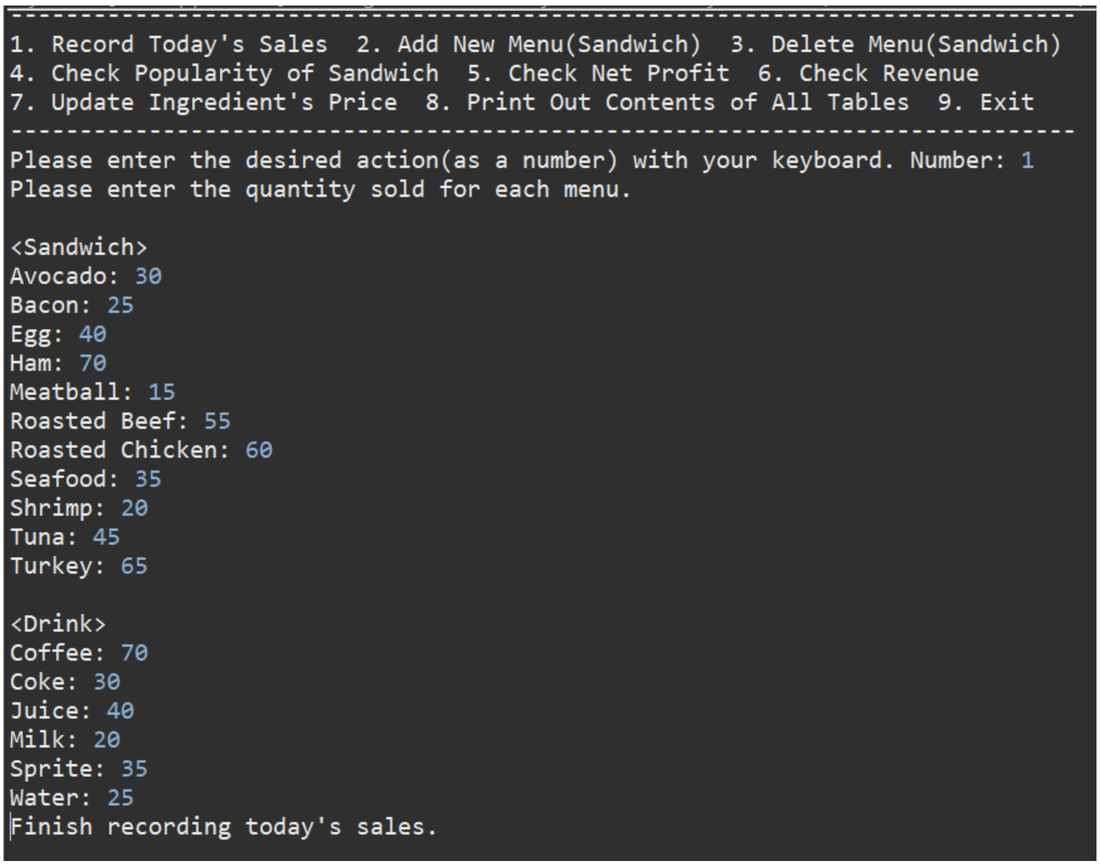
- Part of the code that use above prepared statements (actual execution part)



```
case 2: // 2. Add New Menu(Sandwich)
    System.out.println("Please enter information about new menu.");
    System.out.println("What is the name of the new menu?");
    name = bf.readLine();
    pStmt2.setString(1, name);
    System.out.println("How much is the new menu?");
    price = Float.parseFloat(bf.readLine());
    pStmt2.setFloat(2, price);
    System.out.println("What is the main ingredient of the new menu?");
    mainIngredient = bf.readLine();
    pStmt2.setString(3, mainIngredient);
    pStmt2i.setString(1, mainIngredient);
    System.out.println("How much is the new menu's main ingredient?");
    price = Float.parseFloat(bf.readLine());
    pStmt2i.setFloat(2, price);
    // insert data into ingredient table first(because of referential integrity
    // constraint)
    pStmt2i.executeUpdate();
    pStmt2.executeUpdate();

    System.out.println("\nNew menu has been added.");
    break;
```

**(9) Should have at least 1 interface (menu and user input) and query to update on 1 or 2 tables**

[1. Record Today's Sales] is interface to update data on two tables(Sandwich, Drink).



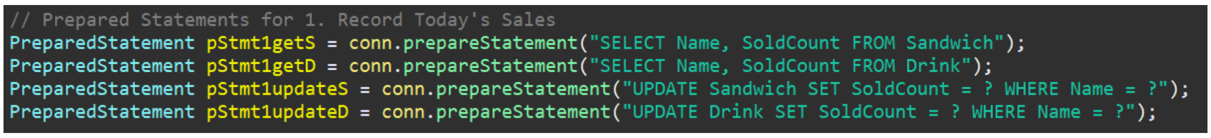
```
-----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
-----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 1
Please enter the quantity sold for each menu.

<Sandwich>
Avocado: 30
Bacon: 25
Egg: 40
Ham: 70
Meatball: 15
Roasted Beef: 55
Roasted Chicken: 60
Seafood: 35
Shrimp: 20
Tuna: 45
Turkey: 65

<Drink>
Coffee: 70
Coke: 30
Juice: 40
Milk: 20
Sprite: 35
Water: 25
Finish recording today's sales.
```

When user wants to record today's sales, user can enter number 1 with keyboard.

The program asks for input value from the user and creates a query using the user input value. For each menu, the program asks how much the menu has been sold.

First, get the menu name and recorded sales data(SoldCount) stored in the database. Secondly, update SoldCount of each menu by adding the today's sales quantity entered by the user. The program doing same job on Sandwich table and Drink table.

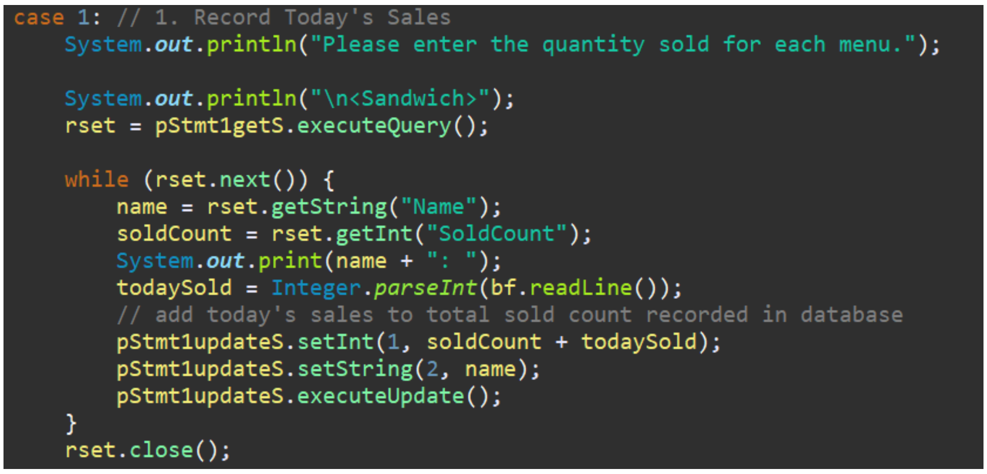
- Prepared statement



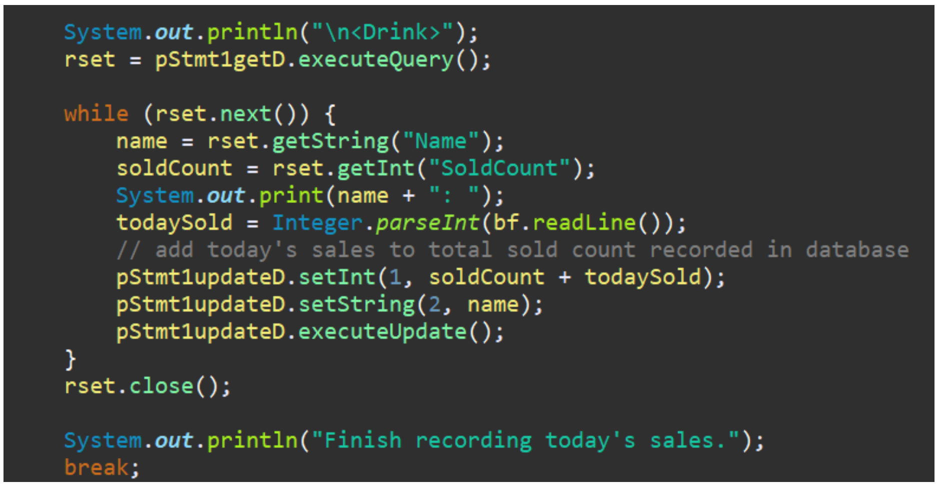
```java
// Prepared Statements for 1. Record Today's Sales
PreparedStatement pStmt1getS = conn.prepareStatement("SELECT Name, SoldCount FROM Sandwich");
PreparedStatement pStmt1getD = conn.prepareStatement("SELECT Name, SoldCount FROM Drink");
PreparedStatement pStmt1updateS = conn.prepareStatement("UPDATE Sandwich SET SoldCount = ? WHERE Name = ?");
PreparedStatement pStmt1updateD = conn.prepareStatement("UPDATE Drink SET SoldCount = ? WHERE Name = ?");
```

  - pStmt1getS is used for retrieving data from Sandwich table.
  - pStmt1getD is used for retrieving data from Drink table.
  - pStmt1updateS is used for updating data from Sandwich table.
  - pStmt1updateD is used for updating data from Drink table.

- Part of the code that use above prepared statements (actual execution part)



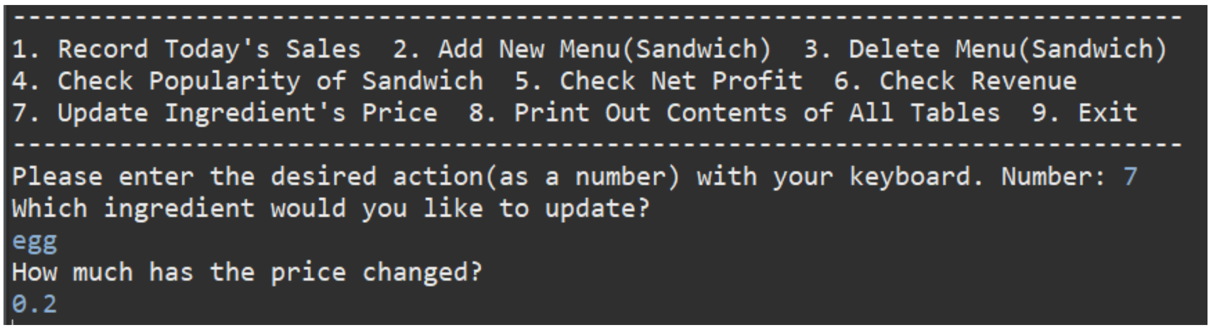
```java
case 1: // 1. Record Today's Sales
    System.out.println("Please enter the quantity sold for each menu.");

    System.out.println("\n<Sandwich>");
    rset = pStmt1getS.executeQuery();

    while (rset.next()) {
        name = rset.getString("Name");
        soldCount = rset.getInt("SoldCount");
        System.out.print(name + ": ");
        todaySold = Integer.parseInt(bf.readLine());
        // add today's sales to total sold count recorded in database
        pStmt1updateS.setInt(1, soldCount + todaySold);
        pStmt1updateS.setString(2, name);
        pStmt1updateS.executeUpdate();
    }
    rset.close();
```



```
    System.out.println("\n<Drink>");
    rset = pStmt1getD.executeQuery();

    while (rset.next()) {
        name = rset.getString("Name");
        soldCount = rset.getInt("SoldCount");
        System.out.print(name + ": ");
        todaySold = Integer.parseInt(bf.readLine());
        // add today's sales to total sold count recorded in database
        pStmt1updateD.setInt(1, soldCount + todaySold);
        pStmt1updateD.setString(2, name);
        pStmt1updateD.executeUpdate();
    }
    rset.close();

    System.out.println("Finish recording today's sales.");
    break;
```

**(10) One of the updates should occur on 2 tables by using transactions**

[7. Update Ingredient's Price] is interface to update data on two tables(Sandwich, Ingredient) by transaction.



```
----------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
----------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 7
Which ingredient would you like to update?
egg
How much has the price changed?
0.2
```

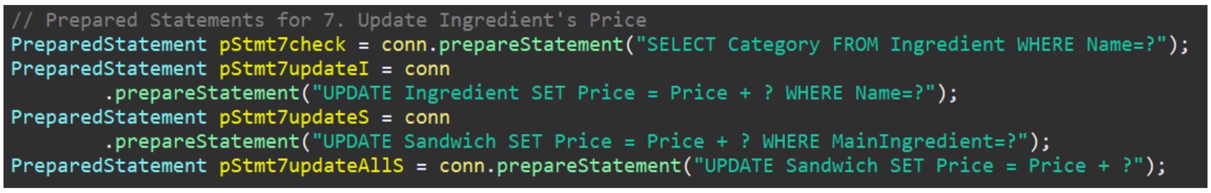
When user wants to update the price of ingredient(cost), user can enter number 7 with keyboard.

The program asks for input value from the user and creates a query using the user input value. The program asks which ingredient to update price and how much the price changed.

If the price of the ingredient increases, the price of the menu using that ingredient should also increase. Otherwise, the owner of sandwich store will lose money. Therefore, updating price of ingredient should be done by transaction.

If the category of the ingredient to update the price is 'main', you only need to update the price of the menu using that ingredient. As in the above example, ingredient egg's price has changed by 0.2. Egg sandwich is the sandwich whose main ingredient is egg. Therefore, Egg sandwich's price should be updated. After running this query, ingredient egg's price and sandwich Egg's price changed by 0.2. Else if the category of the ingredient to update the price is 'sub', which means every sandwich uses all of 'sub' categorized ingredients, you need to update the price of every menu. So, I separate the two cases. When the category of the ingredient to be updated is main, I use pStmt7updateS. When the category of the ingredient to be updated is sub, I use pStmt7updateAllS which updates every sandwich menu's price.

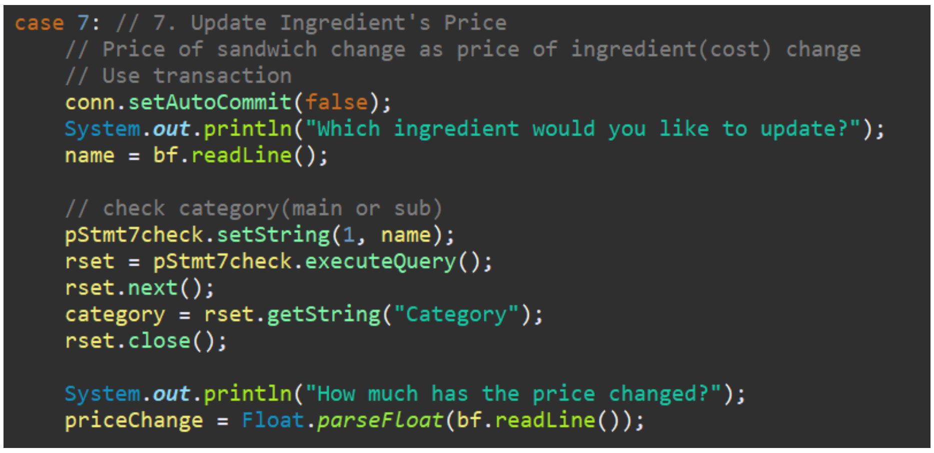
- Prepared statement



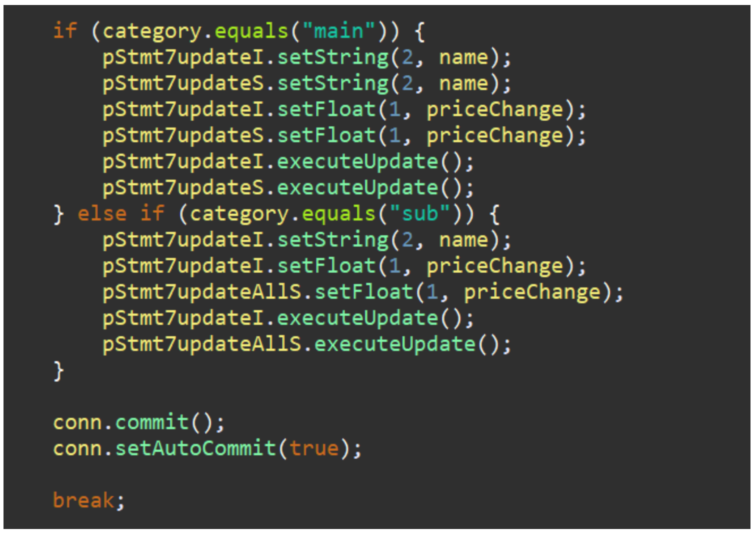
```java
// Prepared Statements for 7. Update Ingredient's Price
PreparedStatement pStmt7check = conn.prepareStatement("SELECT Category FROM Ingredient WHERE Name=?");
PreparedStatement pStmt7updateI = conn
        .prepareStatement("UPDATE Ingredient SET Price = Price + ? WHERE Name=?");
PreparedStatement pStmt7updateS = conn
        .prepareStatement("UPDATE Sandwich SET Price = Price + ? WHERE MainIngredient=?");
PreparedStatement pStmt7updateAllS = conn.prepareStatement("UPDATE Sandwich SET Price = Price + ?");
```

  - pStmt7check is used for checking which category ingredient belongs to.
  - others used for updating data

- Part of the code that use above prepared statements (actual execution part)



```java
case 7: // 7. Update Ingredient's Price
    // Price of sandwich change as price of ingredient(cost) change
    // Use transaction
    conn.setAutoCommit(false);
    System.out.println("Which ingredient would you like to update?");
    name = bf.readLine();

    // check category(main or sub)
    pStmt7check.setString(1, name);
    rset = pStmt7check.executeQuery();
    rset.next();
    category = rset.getString("Category");
    rset.close();

    System.out.println("How much has the price changed?");
    priceChange = Float.parseFloat(bf.readLine());
```

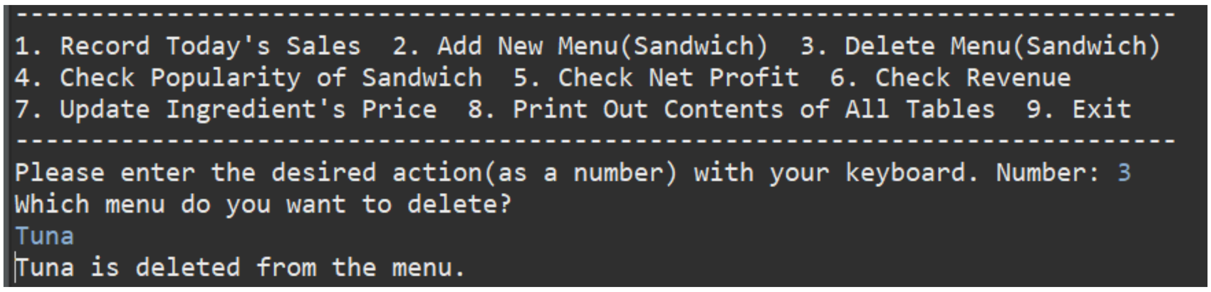


```java
    if (category.equals("main")) {
        pStmt7updateI.setString(2, name);
        pStmt7updateS.setString(2, name);
        pStmt7updateI.setFloat(1, priceChange);
        pStmt7updateS.setFloat(1, priceChange);
        pStmt7updateI.executeUpdate();
        pStmt7updateS.executeUpdate();
    } else if (category.equals("sub")) {
        pStmt7updateI.setString(2, name);
        pStmt7updateI.setFloat(1, priceChange);
        pStmt7updateAllS.setFloat(1, priceChange);
        pStmt7updateI.executeUpdate();
        pStmt7updateAllS.executeUpdate();
    }

    conn.commit();
    conn.setAutoCommit(true);

    break;
```

  - Since this task should be performed by transaction, we first change autocommit to false and then run queries.

**(11) Should have at least 1 interface (menu and user input) and queries to delete from 1 table**

[3. Delete Menu] is interface to delete data on Sandwich table.



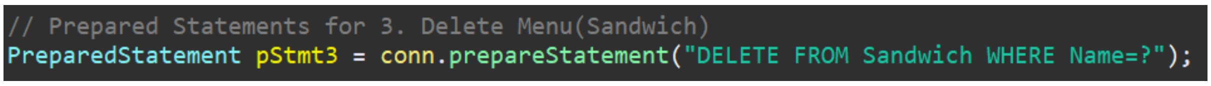
```
-----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
-----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 3
Which menu do you want to delete?
Tuna
Tuna is deleted from the menu.
```

When user wants to delete one of the menus from sandwich table, user can enter number 3 with keyboard.

The program asks for input value from the user and creates a query using the user input value. The program asks which menu to delete.

In the above example, Tuna sandwich is deleted.

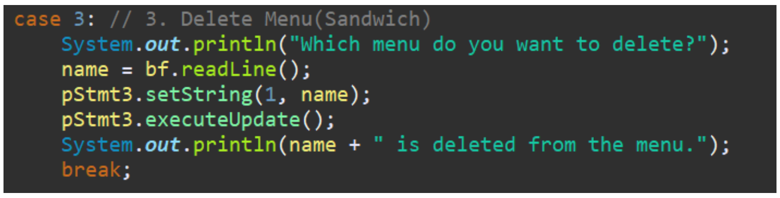
- Prepared statement



```
// Prepared Statements for 3. Delete Menu(Sandwich)
PreparedStatement pStmt3 = conn.prepareStatement("DELETE FROM Sandwich WHERE Name=?");
```

  - pStmt3 is used for deletion.

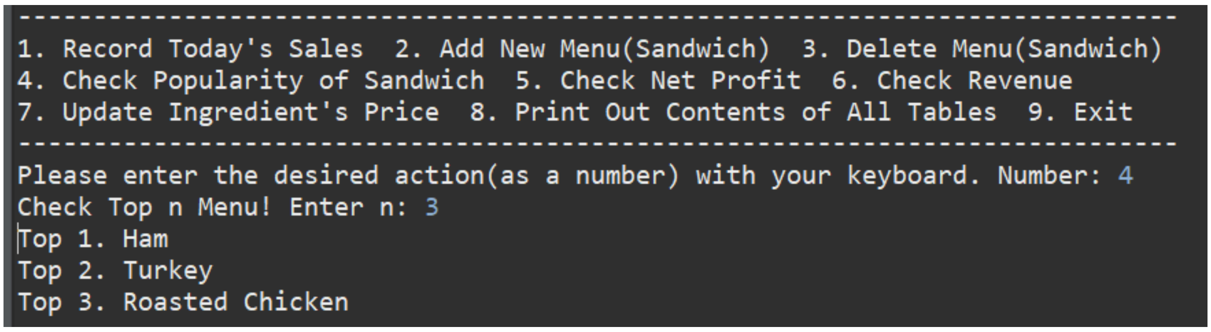
- Part of the code that use above prepared statements (actual execution part)



```
case 3: // 3. Delete Menu(Sandwich)
    System.out.println("Which menu do you want to delete?");
    name = bf.readLine();
    pStmt3.setString(1, name);
    pStmt3.executeUpdate();
    System.out.println(name + " is deleted from the menu.");
    break;
```

**(12) Should have at least 1 interface (menu and user input) and queries to select from database.**

[4. Check Popularity of Sandwich] is interface to retrieve(select) data from Sandwich table.

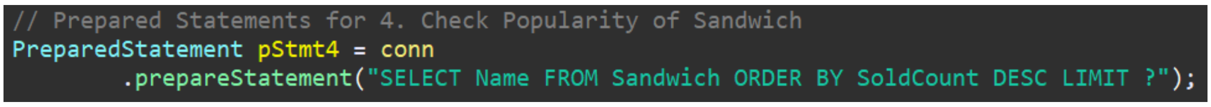


```
-----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
-----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 4
Check Top n Menu! Enter n: 3
Top 1. Ham
Top 2. Turkey
Top 3. Roasted Chicken
```

When user wants to check popularity of the menu from sandwich table, user can enter number 4 with keyboard.

The program asks for input value from the user and creates a query using the user input value. The program asks how many of the top menus you want to see. Then show top n menus. The popularity is based on number sold. In the above example, the ham sandwich sold the most.

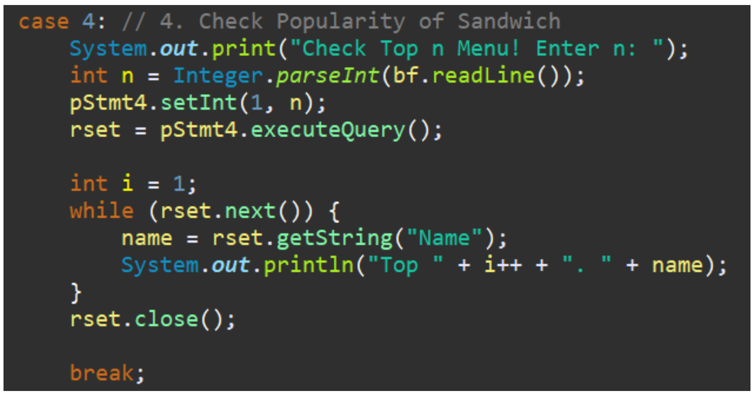
- Prepared statement



```java
// Prepared Statements for 4. Check Popularity of Sandwich
PreparedStatement pStmt4 = conn
        .prepareStatement("SELECT Name FROM Sandwich ORDER BY SoldCount DESC LIMIT ?");
```

  - pStmt4 is used for selection. To select data by descending order, I used "ORDER BY SoldCount DESC".

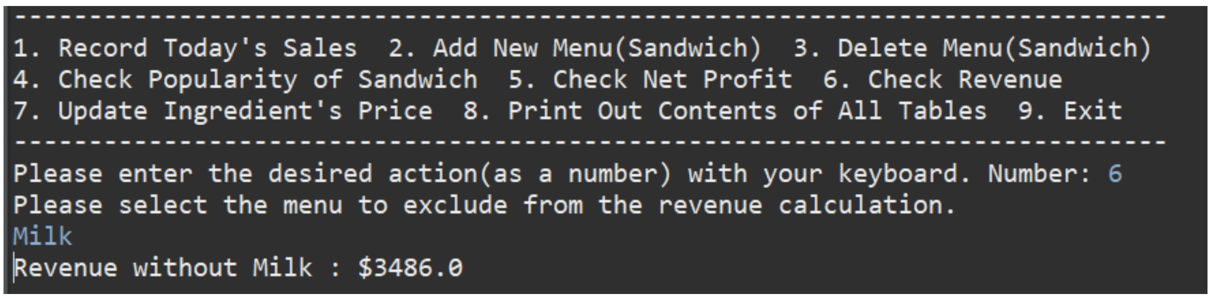
- Part of the code that use above prepared statements (actual execution part)



```java
case 4: // 4. Check Popularity of Sandwich
    System.out.print("Check Top n Menu! Enter n: ");
    int n = Integer.parseInt(bf.readLine());
    pStmt4.setInt(1, n);
    rset = pStmt4.executeQuery();

    int i = 1;
    while (rset.next()) {
        name = rset.getString("Name");
        System.out.println("Top " + i++ + ". " + name);
    }
    rset.close();

    break;
```

**(13) Should have at least 1 interface (menu and user input) and queries to select using nested queries and join.**

[6. Check Revenue] is interface to retrieve(select) data from Sandwich table and Drink table by using nested queries and join.



```
-----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
-----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 6
Please select the menu to exclude from the revenue calculation.
Milk
Revenue without Milk : $3486.0
```

When user wants to check revenue, user can enter number 6 with keyboard.

The program asks for input value from the user and creates a query using the user input value. The program asks which menu to exclude from the revenue calculation. In the above example, the calculated revenue without Milk is shown.

Revenue = sum of (price * sold count) of each menu

- Prepared statement

```
// Prepared Statements for 6. Check Revenue
PreparedStatement pStmt6 = conn.prepareStatement(
                                    "SELECT SUM(Price * SoldCount)+DrinkRevenue
AS Revenue FROM Sandwich, (SELECT SUM(Price * SoldCount) AS DrinkRevenue FROM
Drink WHERE Name!=?)DrinkR WHERE Name!=?");
```

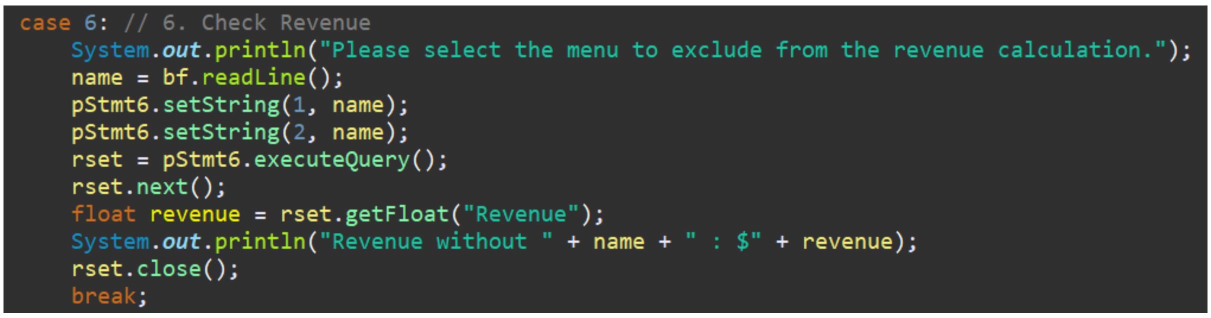
  - pStmt6 is uses nested query and join as below.

    *SELECT SUM(Price * SoldCount)+DrinkRevenue AS Revenue*

    *FROM Sandwich, (SELECT SUM(Price * SoldCount) AS DrinkRevenue FROM Drink WHERE Name!=?)DrinkR*

    *WHERE Name!=?*

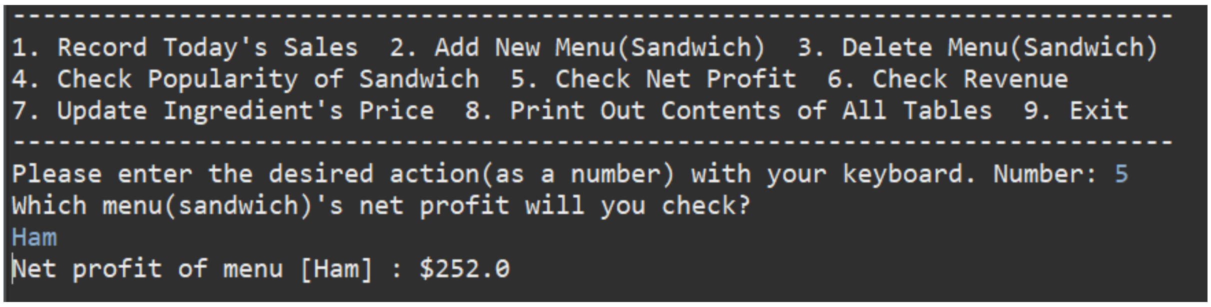
- Part of the code that use above prepared statements (actual execution part)



```
case 6: // 6. Check Revenue
    System.out.println("Please select the menu to exclude from the revenue calculation.");
    name = bf.readLine();
    pStmt6.setString(1, name);
    pStmt6.setString(2, name);
    rset = pStmt6.executeQuery();
    rset.next();
    float revenue = rset.getFloat("Revenue");
    System.out.println("Revenue without " + name + " : $" + revenue);
    rset.close();
    break;
```

**(14) Should have at least 1 interface (menu and user input) and queries to select from view**

[5. Check Net Profit] is interface to retrieve(select) data from view ‘Profit’.



```
----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 5
Which menu(sandwich)'s net profit will you check?
Ham
Net profit of menu [Ham] : $252.0
```

When user wants to check net profit of menu, user can enter number 5 with keyboard.

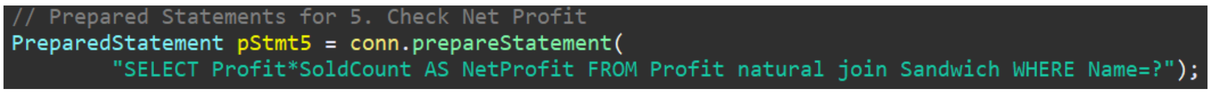
The program asks for input value from the user and creates a query using the user input value. The program asks which menu to check net profit.

Net profit of sandwich = (sandwich’s price – sum of ingredients’ price) * (number sold)

Ingredients for sandwich = (all of sub categorized ingredients + main ingredient of the menu)

In the above example, the net profit of Ham sandwich is $252.

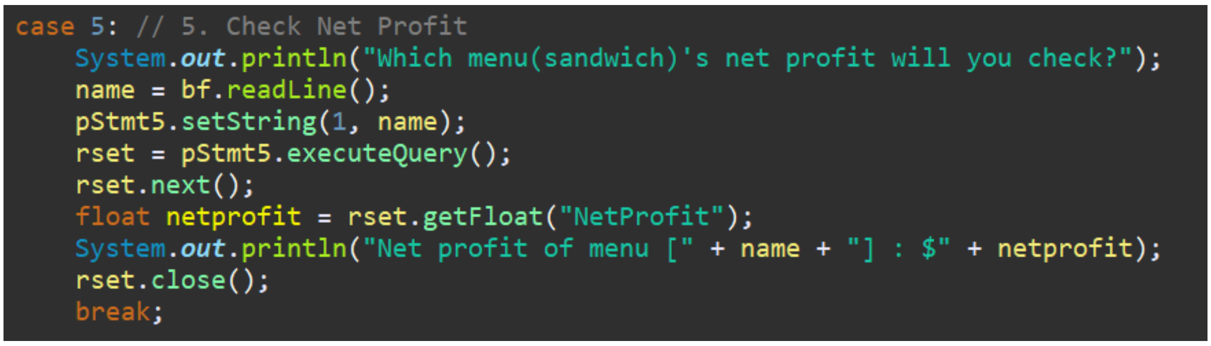
- Prepared statement



```
// Prepared Statements for 5. Check Net Profit
PreparedStatement pStmt5 = conn.prepareStatement(
        "SELECT Profit*SoldCount AS NetProfit FROM Profit natural join Sandwich WHERE Name=?");
```

  - ‘Profit’ is view. pStmt5 is used for selection(calculation is done by aggregate function).

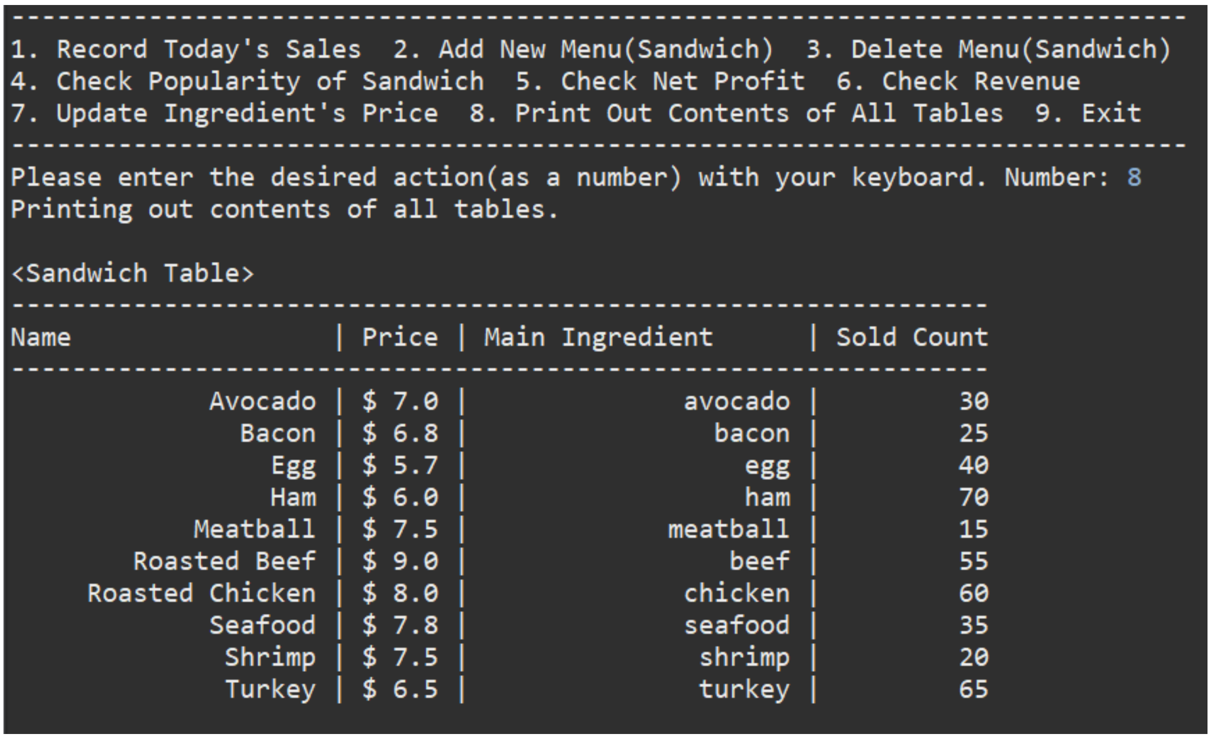
- Part of the code that use above prepared statements (actual execution part)



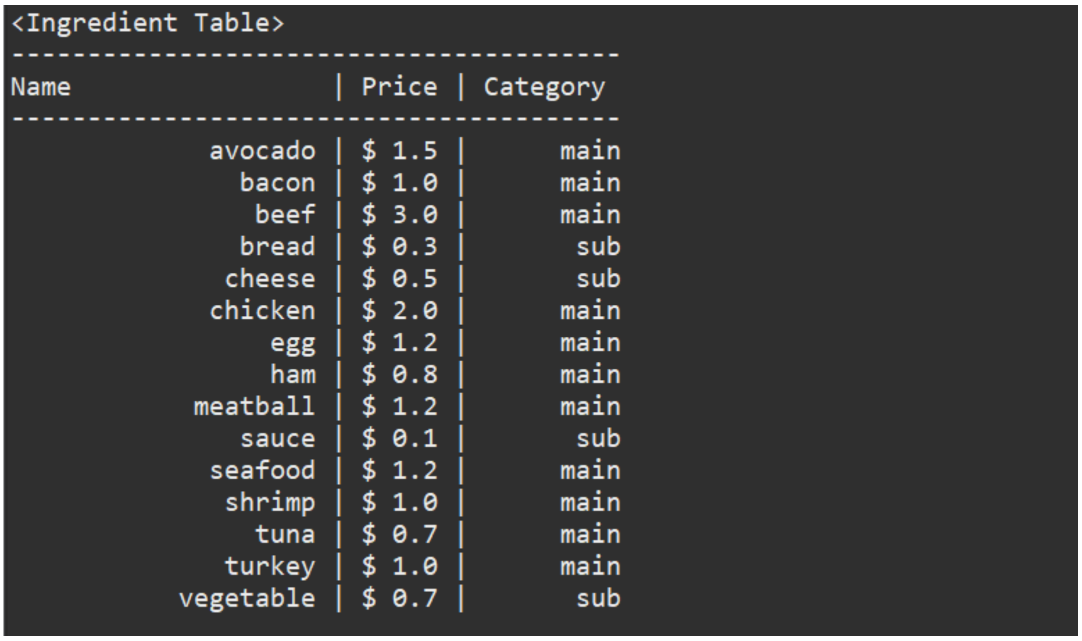
```
case 5: // 5. Check Net Profit
    System.out.println("Which menu(sandwich)'s net profit will you check?");
    name = bf.readLine();
    pStmt5.setString(1, name);
    rset = pStmt5.executeQuery();
    rset.next();
    float netprofit = rset.getFloat("NetProfit");
    System.out.println("Net profit of menu [" + name + "] : $" + netprofit);
    rset.close();
    break;
```

**(15) Should have interface (menu) to print out contents of all tables**

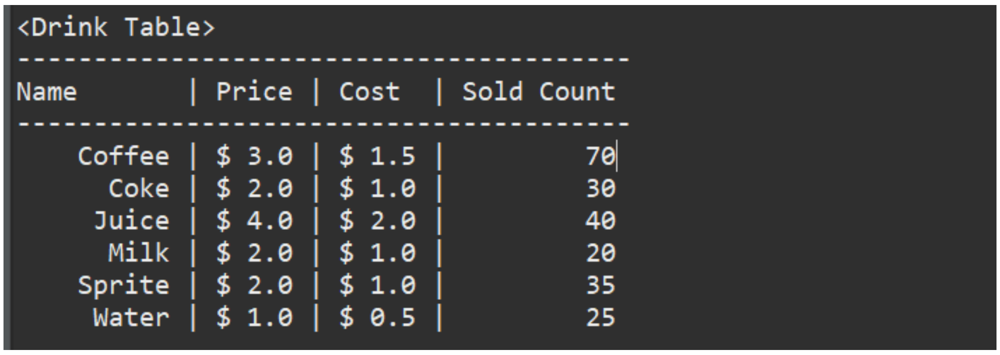
[8. Print Out Contents of All Tables] is interface to retrieve(select) data from all the tables.



```
----------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
----------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 8
Printing out contents of all tables.

<Sandwich Table>
----------------------------------------------------------------
Name                    | Price | Main Ingredient       | Sold Count
----------------------------------------------------------------
             Avocado | $ 7.0 |                avocado |         30
               Bacon | $ 6.8 |                  bacon |         25
                 Egg | $ 5.7 |                    egg |         40
                 Ham | $ 6.0 |                    ham |         70
            Meatball | $ 7.5 |               meatball |         15
        Roasted Beef | $ 9.0 |                   beef |         55
     Roasted Chicken | $ 8.0 |                chicken |         60
             Seafood | $ 7.8 |                seafood |         35
              Shrimp | $ 7.5 |                 shrimp |         20
              Turkey | $ 6.5 |                 turkey |         65
```



```
<Ingredient Table>
----------------------------------------
Name                    | Price | Category
----------------------------------------
             avocado | $ 1.5 |       main
               bacon | $ 1.0 |       main
                beef | $ 3.0 |       main
               bread | $ 0.3 |        sub
              cheese | $ 0.5 |        sub
             chicken | $ 2.0 |       main
                 egg | $ 1.2 |       main
                 ham | $ 0.8 |       main
            meatball | $ 1.2 |       main
               sauce | $ 0.1 |        sub
             seafood | $ 1.2 |       main
              shrimp | $ 1.0 |       main
                tuna | $ 0.7 |       main
              turkey | $ 1.0 |       main
           vegetable | $ 0.7 |        sub
```

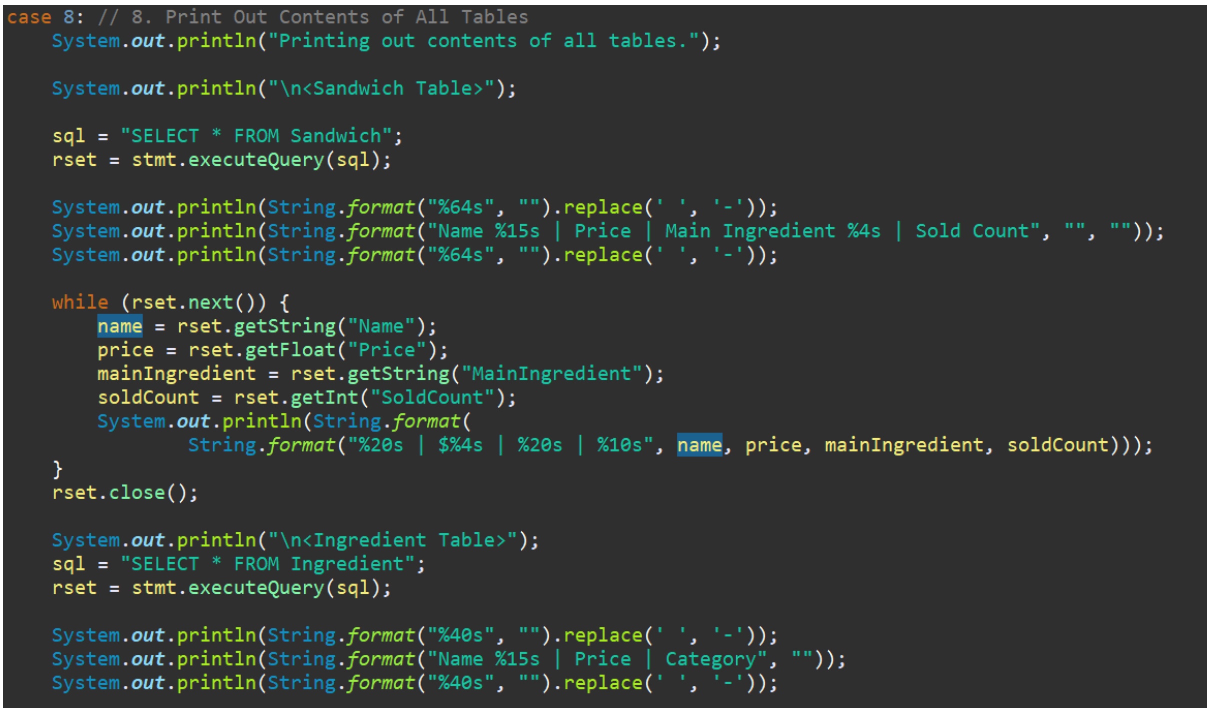


```
<Drink Table>
----------------------------------------
Name       | Price | Cost  | Sold Count
----------------------------------------
    Coffee | $ 3.0 | $ 1.5 |         70
      Coke | $ 2.0 | $ 1.0 |         30
     Juice | $ 4.0 | $ 2.0 |         40
      Milk | $ 2.0 | $ 1.0 |         20
    Sprite | $ 2.0 | $ 1.0 |         35
     Water | $ 1.0 | $ 0.5 |         25
```

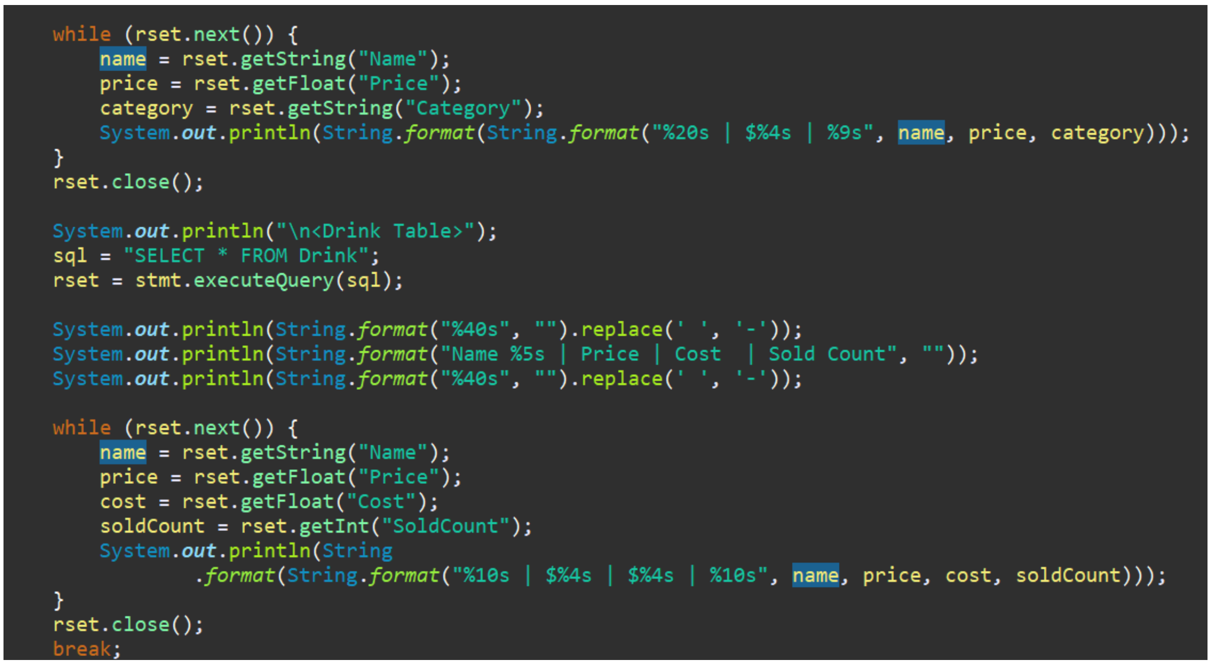
When user wants to see all records from all tables, user can enter number 8 with keyboard.

The program prints out contents of all tables.

- Statement & Part of the code that use above prepared statements (actual execution part)



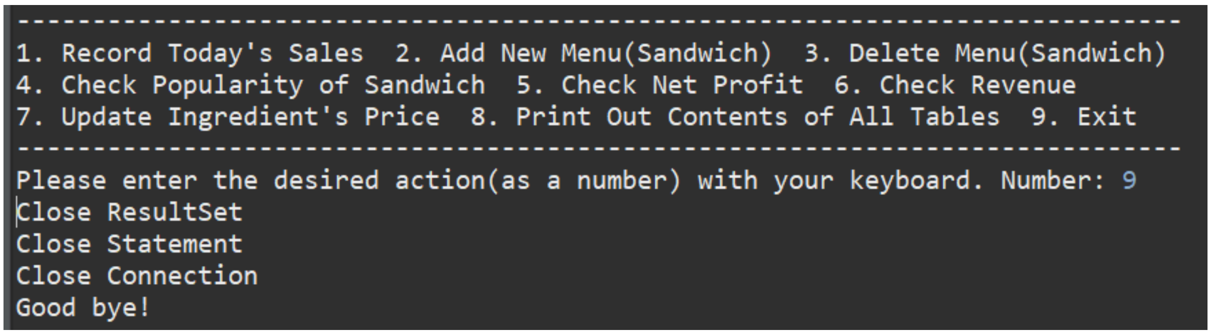
```
case 8: // 8. Print Out Contents of All Tables
    System.out.println("Printing out contents of all tables.");

    System.out.println("\n<Sandwich Table>");

    sql = "SELECT * FROM Sandwich";
    rset = stmt.executeQuery(sql);

    System.out.println(String.format("%64s", "").replace(' ', '-'));
    System.out.println(String.format("Name %15s | Price | Main Ingredient %4s | Sold Count", "", ""));
    System.out.println(String.format("%64s", "").replace(' ', '-'));

    while (rset.next()) {
        name = rset.getString("Name");
        price = rset.getFloat("Price");
        mainIngredient = rset.getString("MainIngredient");
        soldCount = rset.getInt("SoldCount");
        System.out.println(String.format(
                String.format("%20s | $%4s | %20s | %10s", name, price, mainIngredient, soldCount)));
    }
    rset.close();

    System.out.println("\n<Ingredient Table>");
    sql = "SELECT * FROM Ingredient";
    rset = stmt.executeQuery(sql);

    System.out.println(String.format("%40s", "").replace(' ', '-'));
    System.out.println(String.format("Name %15s | Price | Category", ""));
    System.out.println(String.format("%40s", "").replace(' ', '-'));
```



```
    while (rset.next()) {
        name = rset.getString("Name");
        price = rset.getFloat("Price");
        category = rset.getString("Category");
        System.out.println(String.format(String.format("%20s | $%4s | %9s", name, price, category)));
    }
    rset.close();

    System.out.println("\n<Drink Table>");
    sql = "SELECT * FROM Drink";
    rset = stmt.executeQuery(sql);

    System.out.println(String.format("%40s", "").replace(' ', '-'));
    System.out.println(String.format("Name %5s | Price | Cost  | Sold Count", ""));
    System.out.println(String.format("%40s", "").replace(' ', '-'));

    while (rset.next()) {
        name = rset.getString("Name");
        price = rset.getFloat("Price");
        cost = rset.getFloat("Cost");
        soldCount = rset.getInt("SoldCount");
        System.out.println(String
                .format(String.format("%10s | $%4s | $%4s | %10s", name, price, cost, soldCount)));
    }
    rset.close();
    break;
```

**(16) Should have interface (menu) to finish program gracefully. Otherwise menu should**

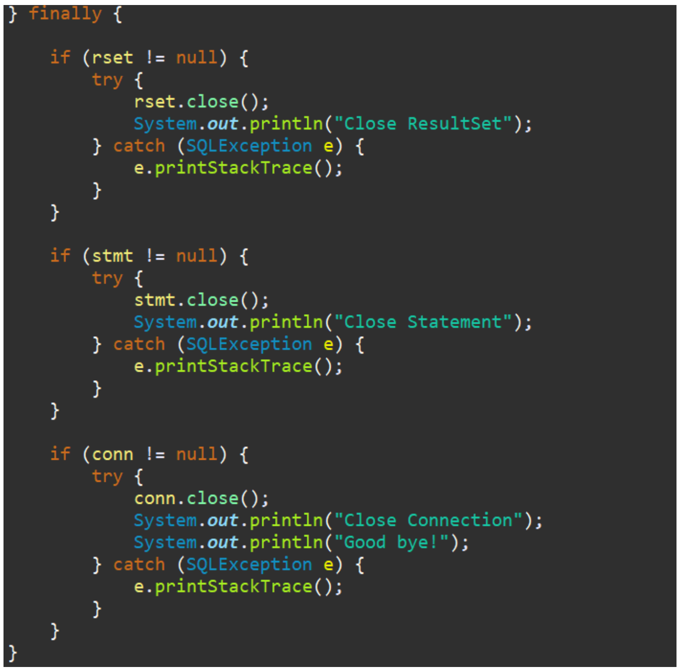
**repeatedly appear automatically.**



```
--------------------------------------------------------------------------
1. Record Today's Sales  2. Add New Menu(Sandwich)  3. Delete Menu(Sandwich)
4. Check Popularity of Sandwich  5. Check Net Profit  6. Check Revenue
7. Update Ingredient's Price  8. Print Out Contents of All Tables  9. Exit
--------------------------------------------------------------------------
Please enter the desired action(as a number) with your keyboard. Number: 9
Close ResultSet
Close Statement
Close Connection
Good bye!
```

The menu repeatedly appear by using while (menu != 9) { … }. The variable menu's value is assigned by user input. By the user input, one of the features is executed. When user enter number 9, the program will be finished gracefully.

- Part of the code that use above prepared statements (actual execution part)



```java
} finally {

    if (rset != null) {
        try {
            rset.close();
            System.out.println("Close ResultSet");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    if (stmt != null) {
        try {
            stmt.close();
            System.out.println("Close Statement");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    if (conn != null) {
        try {
            conn.close();
            System.out.println("Close Connection");
            System.out.println("Good bye!");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

## 6. SQL scripts

**createdb.sql**

```
#create table as schema design - (1), (3), (4)

CREATE TABLE Drink(
            Name VARCHAR(10) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            Cost NUMERIC(2,1),
            SoldCount INT,
            PRIMARY KEY (Name)
);

CREATE TABLE Ingredient(
            Name VARCHAR(20) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            Category VARCHAR(10) NOT NULL,
            PRIMARY KEY (Name)
);

CREATE TABLE Sandwich(
            Name VARCHAR(20) NOT NULL,
            Price NUMERIC(2,1) NOT NULL,
            MainIngredient VARCHAR(20) NOT NULL,
            SoldCount INT,
            PRIMARY KEY (Name),
            FOREIGN KEY (MainIngredient) REFERENCES Ingredient(Name)
);

#insert data into tables - (2)

INSERT INTO Drink (Name, Price) values ('Coke', 2);
INSERT INTO Drink (Name, Price) values ('Sprite', 2);
INSERT INTO Drink (Name, Price) values ('Juice', 4);
INSERT INTO Drink (Name, Price) values ('Water', 1);
INSERT INTO Drink (Name, Price) values ('Coffee', 3);
INSERT INTO Drink (Name, Price) values ('Milk', 2);
UPDATE Drink SET SoldCount = 0;
UPDATE Drink SET Cost = Price * 0.5;
```

```
INSERT INTO Ingredient (Name, Price, Category) values ('egg', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('beef', 3, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('chicken', 2, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('ham', 0.8, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('avocado', 1.5, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('turkey', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('tuna', 0.7, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('bacon', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('meatball', 1.2, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('shrimp', 1, 'main');
INSERT INTO Ingredient (Name, Price, Category) values ('bread', 0.3, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('vegetable', 0.7, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('cheese', 0.5, 'sub');
INSERT INTO Ingredient (Name, Price, Category) values ('sauce', 0.1, 'sub');

INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Egg', 5.5, 'egg');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Roasted Beef', 9, 'beef');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Roasted Chicken', 8, 'chicken');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Ham', 6, 'ham');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Avocado', 7, 'avocado');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Turkey', 6.5, 'turkey');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Tuna', 6.3, 'tuna');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Bacon', 6.8, 'bacon');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Meatball', 7.5, 'meatball');
INSERT INTO Sandwich (Name, Price, MainIngredient) values ('Shrimp', 7.5, 'shrimp');
UPDATE Sandwich SET SoldCount = 0;

#create index - (5)

CREATE INDEX SandwichSoldCountIndex ON Sandwich(SoldCount);
CREATE INDEX DrinkSoldCountIndex ON Drink(SoldCount);

#create view - (6)

CREATE VIEW DefaultCost AS
SELECT sum(Price) as Cost
FROM Ingredient
WHERE Category='sub';
```

```
CREATE VIEW Profit AS
SELECT (Sandwich.Name) AS Name, (Sandwich.Price - Ingredient.Price - DefaultCost.Cost) as Profit
FROM Sandwich, Ingredient, DefaultCost
WHERE Sandwich.MainIngredient = Ingredient.Name;
```

**dropdb.sql**

```
#DROP indices

ALTER TABLE Sandwich DROP INDEX SandwichSoldCountIndex;
ALTER TABLE Drink DROP INDEX DrinkSoldCountIndex;

#DROP tables

DROP TABLE Drink;
DROP TABLE Sandwich;
DROP TABLE Ingredient;

#DROP views

DROP VIEW DefaultCost;
DROP VIEW Profit;
```

## 7. Java codes

```
package dbproject;

import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class MyDriver {
         public static void main(String[] args) throws NumberFormatException,
IOException {
                  // initialize
                  String userID = "dbuser";
                  String userPW = "dbpwd";
                  String url = "jdbc:mysql://localhost:3306/dbprj?&serverTimezone=UTC";
                  Connection conn = null;
                  Statement stmt = null;
                  ResultSet rset = null;
                  String sql = "";
                  BufferedReader bf = new BufferedReader(new
InputStreamReader(System.in));
                  // variables for project
                  String name;
                  float price, cost, priceChange;
                  String mainIngredient;
                  int soldCount, todaySold;
                  String category;
                  try {
                           /*
                            * Class.forName("com.mysql.jdbc.Driver"); This is
deprecated. The new driver
                            * class is `com.mysql.cj.jdbc.Driver'. The driver is
automatically registered
                            * via the SPI and manual loading of the driver class is
generally unnecessary.
                            *
                            */
                           // construct a connection to mysql server
                           conn = DriverManager.getConnection(url, userID, userPW);
                           stmt = conn.createStatement();
                           // Prepared Statements for 1. Record Today's Sales
                           PreparedStatement pStmt1getS = conn.prepareStatement("SELECT
Name, SoldCount FROM Sandwich");
                           PreparedStatement pStmt1getD = conn.prepareStatement("SELECT
Name, SoldCount FROM Drink");
                           PreparedStatement pStmt1updateS =
conn.prepareStatement("UPDATE Sandwich SET SoldCount = ? WHERE Name = ?");
                           PreparedStatement pStmt1updateD =
conn.prepareStatement("UPDATE Drink SET SoldCount = ? WHERE Name = ?");
                           // Prepared Statements for 2. Add New Menu(Sandwich)
                           PreparedStatement pStmt2 = conn
                                             .prepareStatement("INSERT INTO Sandwich
(Name, Price, MainIngredient, SoldCount) VALUES (?,?,?,0)");
                           PreparedStatement pStmt2i = conn
                                             .prepareStatement("INSERT INTO Ingredient
(Name, Price, Category) VALUES (?,?,'main')");
                           // Prepared Statements for 3. Delete Menu(Sandwich)
                           PreparedStatement pStmt3 = conn.prepareStatement("DELETE FROM
Sandwich WHERE Name=?");
                           // Prepared Statements for 4. Check Popularity of Sandwich
```

```
                        PreparedStatement pStmt4 = conn
                                .prepareStatement("SELECT Name FROM
Sandwich ORDER BY SoldCount DESC LIMIT ?");
                        // Prepared Statements for 5. Check Net Profit
                        PreparedStatement pStmt5 = conn.prepareStatement(
                                "SELECT Profit*SoldCount AS NetProfit FROM
Profit natural join Sandwich WHERE Name=?");
                        // Prepared Statements for 6. Check Revenue
                        PreparedStatement pStmt6 = conn.prepareStatement(
                                "SELECT SUM(Price *
SoldCount)+DrinkRevenue AS Revenue FROM Sandwich, (SELECT SUM(Price * SoldCount) AS
DrinkRevenue FROM Drink WHERE Name!=?)DrinkR WHERE Name!=?");
                        // Prepared Statements for 7. Update Ingredient's Price
                        PreparedStatement pStmt7check = conn.prepareStatement("SELECT
Category FROM Ingredient WHERE Name=?");
                        PreparedStatement pStmt7updateI = conn
                                .prepareStatement("UPDATE Ingredient SET
Price = Price + ? WHERE Name=?");
                        PreparedStatement pStmt7updateS = conn
                                .prepareStatement("UPDATE Sandwich SET
Price = Price + ? WHERE MainIngredient=?");
                        PreparedStatement pStmt7updateAllS =
conn.prepareStatement("UPDATE Sandwich SET Price = Price + ?");
                        if (conn != null) {
                                System.out.println("Connected to database.\n");
                                System.out.println(String.format("%66s",
"").replace(' ', '='));
                                System.out.println("Welcome to Ewha Sandwich
Store's Menu(Database) Management System!");
                                System.out.println(String.format("%66s",
"").replace(' ', '='));
                        }
                        int menu = 0;
                        while (menu != 9) {
                                // show menu for user input
                                System.out.println("\n\n" + String.format("%77s",
"").replace(' ', '-'));
                                System.out.println("1. Record Today's Sales  2. Add
New Menu(Sandwich)  3. Delete Menu(Sandwich)");
                                System.out.println("4. Check Popularity of Sandwich
5. Check Net Profit  6. Check Revenue");
                                System.out.println("7. Update Ingredient's Price  8.
Print Out Contents of All Tables  9. Exit ");
                                System.out.println(String.format("%77s",
"").replace(' ', '-'));
                                System.out.print("Please enter the desired
action(as a number) with your keyboard. Number: ");
                                menu = Integer.parseInt(bf.readLine());
                                switch (menu) {
                                case 1: // 1. Record Today's Sales
                                        System.out.println("Please enter the
quantity sold for each menu.");

                                        System.out.println("\n<Sandwich>");
                                        rset = pStmt1getS.executeQuery();
                                        while (rset.next()) {
                                                name = rset.getString("Name");
                                                soldCount =
rset.getInt("SoldCount");

                                                System.out.print(name + ": ");
                                                todaySold =
Integer.parseInt(bf.readLine());
                                                // add today's sales to total
sold count recorded in database
```

```
                                                        pStmt1updateS.setInt(1, soldCount
+ todaySold);
                                                        pStmt1updateS.setString(2, name);
                                                        pStmt1updateS.executeUpdate();
                                              }
                                              rset.close();
                                              System.out.println("\n<Drink>");
                                              rset = pStmt1getD.executeQuery();
                                              while (rset.next()) {
                                                        name = rset.getString("Name");
                                                        soldCount =
rset.getInt("SoldCount");
                                                        System.out.print(name + ": ");
                                                        todaySold =
Integer.parseInt(bf.readLine());
                                                        // add today's sales to total
sold count recorded in database
                                                        pStmt1updateD.setInt(1, soldCount
+ todaySold);
                                                        pStmt1updateD.setString(2, name);
                                                        pStmt1updateD.executeUpdate();
                                              }
                                              rset.close();
                                              System.out.println("Finish recording
today's sales.");
                                              break;
                                     case 2: // 2. Add New Menu(Sandwich)
                                              System.out.println("Please enter
information about new menu.");
                                              System.out.println("What is the name of
the new menu?");
                                              name = bf.readLine();
                                              pStmt2.setString(1, name);
                                              System.out.println("How much is the new
menu?");
                                              price = Float.parseFloat(bf.readLine());
                                              pStmt2.setFloat(2, price);
                                              System.out.println("What is the main
ingredient of the new menu?");
                                              mainIngredient = bf.readLine();
                                              pStmt2.setString(3, mainIngredient);
                                              pStmt2i.setString(1, mainIngredient);
                                              System.out.println("How much is the new
menu's main ingredient?");
                                              price = Float.parseFloat(bf.readLine());
                                              pStmt2i.setFloat(2, price);
                                              // insert data into ingredient table
first(because of referential integrity
                                              // constraint)
                                              pStmt2i.executeUpdate();
                                              pStmt2.executeUpdate();
                                              System.out.println("\nNew menu has been
added.");
                                              break;
                                     case 3: // 3. Delete Menu(Sandwich)
                                              System.out.println("Which menu do you want
to delete?");
                                              name = bf.readLine();
                                              pStmt3.setString(1, name);
                                              pStmt3.executeUpdate();
                                              System.out.println(name + " is deleted
from the menu.");
                                              break;
                                     case 4: // 4. Check Popularity of Sandwich
```

```
                                              System.out.print("Check Top n Menu! Enter 
n: ");
                                              int n = Integer.parseInt(bf.readLine());
                                              pStmt4.setInt(1, n);
                                              rset = pStmt4.executeQuery();
                                              int i = 1;
                                              while (rset.next()) {
                                                       name = rset.getString("Name");
                                                       System.out.println("Top " + i++ + 
". " + name);
                                              }
                                              rset.close();
                                              break;
                                     case 5: // 5. Check Net Profit
                                              System.out.println("Which menu(sandwich)'s 
net profit will you check?");
                                              name = bf.readLine();
                                              pStmt5.setString(1, name);
                                              rset = pStmt5.executeQuery();
                                              rset.next();
                                              float netprofit = 
rset.getFloat("NetProfit");
                                              System.out.println("Net profit of menu [" 
+ name + "] : $" + netprofit);
                                              rset.close();
                                              break;
                                     case 6: // 6. Check Revenue
                                              System.out.println("Please select the menu 
to exclude from the revenue calculation.");
                                              name = bf.readLine();
                                              pStmt6.setString(1, name);
                                              pStmt6.setString(2, name);
                                              rset = pStmt6.executeQuery();
                                              rset.next();
                                              float revenue = rset.getFloat("Revenue");
                                              System.out.println("Revenue without " + 
name + " : $" + revenue);
                                              rset.close();
                                              break;
                                     case 7: // 7. Update Ingredient's Price
                                              // Price of sandwich change as price of 
ingredient(cost) change
                                              // Use transaction
                                              conn.setAutoCommit(false);
                                              System.out.println("Which ingredient would 
you like to update?");
                                              name = bf.readLine();
                                              // check category(main or sub)
                                              pStmt7check.setString(1, name);
                                              rset = pStmt7check.executeQuery();
                                              rset.next();
                                              category = rset.getString("Category");
                                              rset.close();
                                              System.out.println("How much has the price 
changed?");
                                              priceChange = 
Float.parseFloat(bf.readLine());
                                              if (category.equals("main")) {
                                                       pStmt7updateI.setString(2, name);
                                                       pStmt7updateS.setString(2, name);
                                                       pStmt7updateI.setFloat(1, 
priceChange);
                                                       pStmt7updateS.setFloat(1, 
priceChange);
```

```
                                                        pStmt7updateI.executeUpdate();
                                                        pStmt7updateS.executeUpdate();
                                               } else if (category.equals("sub")) {
                                                        pStmt7updateI.setString(2, name);
                                                        pStmt7updateI.setFloat(1,
priceChange);
                                                        pStmt7updateAllS.setFloat(1,
priceChange);
                                                        pStmt7updateI.executeUpdate();
                                                        pStmt7updateAllS.executeUpdate();
                                               }
                                               conn.commit();
                                               conn.setAutoCommit(true);
                                               break;
                                      case 8: // 8. Print Out Contents of All Tables
                                               System.out.println("Printing out contents
of all tables.");
                                               System.out.println("\n<Sandwich Table>");
                                               sql = "SELECT * FROM Sandwich";
                                               rset = stmt.executeQuery(sql);
                                               System.out.println(String.format("%64s",
"").replace(' ', '-'));

         System.out.println(String.format("Name %15s | Price | Main Ingredient %4s |
Sold Count", "", ""));
                                               System.out.println(String.format("%64s",
"").replace(' ', '-'));
                                               while (rset.next()) {
                                                        name = rset.getString("Name");
                                                        price = rset.getFloat("Price");
                                                        mainIngredient =
rset.getString("MainIngredient");
                                                        soldCount =
rset.getInt("SoldCount");
                                                        System.out.println(String.format(

         String.format("%20s | $%4s | %20s | %10s", name, price, mainIngredient,
soldCount)));
                                               }
                                               rset.close();
                                               System.out.println("\n<Ingredient
Table>");
                                               sql = "SELECT * FROM Ingredient";
                                               rset = stmt.executeQuery(sql);
                                               System.out.println(String.format("%40s",
"").replace(' ', '-'));

         System.out.println(String.format("Name %15s | Price | Category", ""));
                                               System.out.println(String.format("%40s",
"").replace(' ', '-'));
                                               while (rset.next()) {
                                                        name = rset.getString("Name");
                                                        price = rset.getFloat("Price");
                                                        category =
rset.getString("Category");

         System.out.println(String.format(String.format("%20s | $%4s | %9s", name,
price, category)));
                                               }
                                               rset.close();
                                               System.out.println("\n<Drink Table>");
                                               sql = "SELECT * FROM Drink";
                                               rset = stmt.executeQuery(sql);
```

```
                                        System.out.println(String.format("%40s",
"").replace(' ', '-'));

                                        System.out.println(String.format("Name %5s

| Price | Cost  | Sold Count", ""));

                                        System.out.println(String.format("%40s",

"").replace(' ', '-'));

                                        while (rset.next()) {
                                                name = rset.getString("Name");
                                                price = rset.getFloat("Price");
                                                cost = rset.getFloat("Cost");
                                                soldCount =

rset.getInt("SoldCount");

                                                System.out.println(String

        .format(String.format("%10s | $%4s | $%4s | %10s", name, price, cost,
soldCount)));
                                        }
                                        rset.close();
                                        break;
                                }
                        }
                } catch (SQLException e) {
                        e.printStackTrace();
                } finally {
                        if (rset != null) {
                                try {
                                        rset.close();
                                        System.out.println("Close ResultSet");
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                        if (stmt != null) {
                                try {
                                        stmt.close();
                                        System.out.println("Close Statement");
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                        if (conn != null) {
                                try {
                                        conn.close();
                                        System.out.println("Close Connection");
                                        System.out.println("Good bye!");
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                }
        }
}
```