Table Design:

Between the table STAFF\_ORDERS and ORDERS, it’s better to put them into one table. There are 2 advantage: one is to save more storage space, the second one is to select date more conveniently.

Database connection design:

DROP SEQUENCE StuffSequence;

CREATE SEQUENCE StuffSequence START 1 INCREMENT BY 1;

I make use this kind of sequence to increase the id in ORDERS, INVENTORY and STAFF. And all foreign key is ON DELETE CASCADE type. So, there will be no influence when an item is deleted from the table.

Function:

Option1-3

I designed a retry loop when input data is invalid. When user try to input any invalid data (maybe nothing), the program can handle these faults and analyze which kind of invalid type of the input and suggest user to entre a right type of input. For example, in the part of processing product id, the program will get the input and parse it into integer type. If not success, then user will be noticed that’s an invalid input. After user entre a integer, the program will use function isProductID\_Valid (Connection conn,int ProductID) to run “select \* from INVENTORY where ProductID=(input integer)”. If this product id does not exist, program will catch this fault and notify user to enter the product id again. The product quantity also has limitation. When user input a quantity, program will run psql command to check if the stock amount is enough to reduce in this order. And in the part of Y/N judgement, I create Boolean value to build a loop outside of input product id, quantity. Only when user entre ‘N’, the loop will end, which satisfies the requirement. To check if the input date is valid, I use java.util.date to check the formatter of input. Only “dd-Mon-YY” can be stored. Before storing the date, I transfer it into sql.date type because the smallest time unit is day, so do not need to worry the time loss. I use similar method to handle all kind of input (Fname, Lname, street, city and soon) and finally store them into database safely. It’s also necessary to compare the date in option2 and option3.

Option4:

Select Products price multiplies the total sold quantity and combine the table together with a descending order.

Option5:

To achieve this part, I firstly reduce 8 days of the date then put it into the database to process. I use timestamp to compare the date and use while loop in SQL sentences to delete the items in order id one by one. And finally add back the quantity cancelled and delete the order.

Option6:

I select necessary column and join them several times. Find the value of each order, then add the order value to a staff id, lastly convert staff id into stuff name. Eventually, the value sum>=50000 will be selected.

Option7:

In this part, I am able get every information I need but I don’t know to transfer a column to a row in database, so I used a different method: print them directly by Java. I make use of arrayList again to store the distinct value. And I use two loops to print them out. When the quantity number is null in database, it will print ‘0’ instead.

Option8: I tried to find a way to process the view with an integer input, but due to the time reason, I have to put the sql command (all of them are selecting sentences) into the java. So, the problem is just a selection procedure. Find the productIDs that have sold more than £20,000 in the given year. Then count distinct ProductIDs for each StaffID left in this table. Finally, get staffIDs that the count of distinct ProductIDs is equal to the count of ProductIDs