CS258 Database Systems: Assignment (2019/2020)

A department store has recently expanded its services to increase the number of ways of purchasing items from the store. The store's managers and technical team have decided that the store's business can be stored within the following tables.

INVENTORY
The store keeps a track of its inventory. To do so, it stores an item description, the item price and the number of items currently in stock.

ORDERS
As well as being able to purchase in-store, a customer can place an order and collect from the store or arrange a home delivery. A store also tracks whether an order has been completed. For an in-store purchase, completion is immediate, for collections and deliveries an order is only marked as complete once the item is in the hands of the customer.

ORDER_PRODUCTS
For an order of any type, the amount of each product that is in the order must be stored, and the amount in stock must be reduced correspondingly. Multiple different products can be part of the same order.

COLLECTIONS
If the customer purchases an item in-store, then no customer details are needed, just the information about the products in the order as specified above. If either of the other ordering options is followed, then the amount of stock must still be reduced, but it is also necessary to store some additional details. For in-store collection, the store keeps track of an order-id, a customer name (as first and last name) and the expected collection date, but does not keep the customer address.

DELIVERIES
For a delivery, the store keeps track of not only an order id, but the customer name, a delivery address and the requested delivery date.

STAFF and **STAFF_ORDERS**
The store also decided to refine its analytics by refining its inventory information to include details of its staff (for the purposes of testing, assume that all existing and future orders are associated with one and only one member of staff). Each member of staff has a staff id, and a staff name. Every order is associated with a member of staff who is responsible for the order. Any staff member can sell any particular item in the store, and they can sell more than one kind of item.

The goal of this coursework is for you to develop a Java application using JDBC, and a database for this system. You will need to provide SQL statements for creating the tables (see Section 2), develop a selection of functions (Section 4) and test your application using test data you come up with yourself. **Where possible, you should aim to perform any operations using SQL, rather than extracting and processing data within the Java application.**

1 Submission

Your final submission should consist of three files; Assignment.java, README.md and schema.sql.

All schema related SQL (including any tables/views/functions/procedures/sequences/triggers) should be in the schema.sql file.

Your submitted Java code **should not** actively perform the following:

1. Create any of the database schema
2. Insert test data

We will destroy and recreate the tables with test data as part of the marking process. Your submission does however have to interact with the tables described in this assignment, and it is expected that you will perform the process of creating the tables and inserting data for your own testing purposes to ensure your solution functions correctly.

Submit three files: Assignment.java, README.md and schema.sql via Tabula by noon *on Thursday 5th December 2019*.

2 Initialization [20%]

Write SQL code to create the tables from Figure 1. Use the following types for the respective columns:

- integer (ProductID, OrderID, ProductQuantity, ProductStockAmount, StaffID)
- varchar(30) (ProductDesc, OrderType, FName, LName, House, Street, City)
- numeric(8,2) (ProductPrice)
- integer (OrderCompleted)
- Date (OrderPlaced, DeliveryDate, CollectionDate)

For OrderType, acceptable values are "InStore", "Collection" or "Delivery".
For OrderCompleted, 0 is treated as an uncompleted order, and 1 as a completed order.
Ensure the table names and attribute names match those provided.

Specify key and referential integrity constraints so as to capture the description given in the introduction as accurately as possible. Also specify any additional constraints you believe apply to the system. Include the SQL code in your schema.sql file, with justification for your choices in the README.md file.

3 Design Choices [5%]

The tables as designed in Fig. 1 should not be changed for this coursework. However, if you were to make any modifications, what changes might you make to improve the table structure? Write your answer in a "Design Choices" section in the README.md file.

4 Application [75%]

Write (and test) a Java program that repeatedly displays the menu below and allows the user to select one of the options. Each option is specified below, along with the expected output format. A skeleton file is provided for you to use, and it is expected that your final submission will be a completed version of this skeleton file that completes any indicated methods, and expands upon the existing main() method to provided the application functionality. You may add additional methods should you so wish, but you must implement the provided incomplete methods to implement the functionality

INVENTORY

ProductID	ProductDesc	ProductPrice	ProductStockAmount

ORDERS

OrderID	OrderType	OrderCompleted	OrderPlaced

ORDER_PRODUCTS

OrderID	ProductID	ProductQuantity

DELIVERIES

OrderID	FName	LName	House	Street	City	DeliveryDate

COLLECTIONS

OrderID	FName	LName	CollectionDate

STAFF

StaffID	FName	LName

STAFF_ORDERS

StaffID	OrderID

Figure 1: Table Schemas

described in the file and in this document. Once an option has completed, the program should return to the menu (until Quit is selected).

You should not read any command line inputs in these methods, the data they use should be passed as parameters to the respective methods. As part of the menu functionality, the data should be read in via prompts in the main method (use readEntry) and put into appropriate data structures for passing to these methods. The methods do however handle any data printing to the screen. When testing your code, you can create a separate testing method that contains hard-coded test cases passed as arguments without the need to enter data at a prompt.

Document your solutions, any decisions that you make should be justified in the README.md file. It is expected that your program appropriately handle any error

cases that might occur, such as invalid inputs or failures that arise out of violating constraints applied on the database.

MENU:

- (1) In-Store Purchases
- (2) Collection
- (3) Delivery
- (4) Biggest Sellers
- (5) Reserved Stock
- (6) Staff Life-Time Success
- (7) Staff Contribution
- (8) Employees of the Year
- (0) Quit
- Enter your choice:

4.1 Purchases: Options 1-3

Options 1-3 deal with storing order information. When someone makes a purchase, it is necessary to create a new order and store the details. Anytime a new order is created you must:

- Store the date the order was made (as specified by the user).
- Store a productID and the quantity of that product sold, for each and every product that is part of the order.
- Reduce the amount of stock of the product by the amount sold, for each and every product that is part of the order.
- Store whether an order has been completed or not
- Store the Staff ID of the employee who added the order

4.1.1 In-Store Orders (Method option1)

If an order is an instore order (Order type 'InStore'), it is automatically considered a completed order, and this should be stored alongside the information above that applies to all orders - no additional customer information is needed.

This option should be implemented using the method 'option1'. It should take a Database Connection (conn), an int array of productIDs that belong to the order, an int array of quantities (the index of the quantity array corresponds to the index of a matching productID in productIDs), the date which the order was made as a string and the ID of the staff member as an int. This string should correspond to the default Oracle Date formatting (i.e. 'DD-Mon-YY' such as '09-Nov-17').

Once any data storage or updates are complete, the option should print out a list of updated product quantities with their product ID, one per line.

The below is an example of the formatting to be expected when running Option 1:

Example Input

```
Enter your choice: 1
Enter a product ID: 1049
Enter the quantity sold: 5
Is there another product in the order?: Y
Enter a productID: 1048
Enter the quantity sold: 2
Is there another product in the order ?: N
Enter the date sold: 01-Jan-10
Enter your staff ID: 5
```

Example Output (Database should also be updated)

```
Product ID 1049 stock is now at 20.
Product ID 1048 stock is now at 48.
```

4.1.2 Collection Orders (Method option2)

If an order is a collection order (OrderType 'Collection'), it defaults to uncompleted. In addition to the information that must be stored for all orders, it is also necessary to find out and store:

- The date (same formatting as above) that the item will be collected.
- The first and last name of the person collecting the order.

This option should be implemented using the method 'option2'. It should take a Database Connection (conn), an int array of productIDs that belong to the order, an int array of quantities (the index of the quantity array corresponds to the index of a matching productID in productIDs), the date which the order was made as a string, the date of collection, the first and last name of the collector as strings and the ID of the staff member as an int. Any dates should correspond to the default Oracle Date

formatting (i.e. 'DD-Mon-YY' such as '09-Nov-17').

Once any data storage or updates are complete, the option should print out a list of updated product quantities with their product ID, one per line.

The below is an example of the formatting to be expected when running Option 2:

Example Input

```
Enter your choice: 2
Enter a product ID: 1049
Enter the quantity sold: 5
Is there another product in the order?: Y
Enter a productID: 1048
Enter the quantity sold: 2
Is there another product in the order ?: N
Enter the date sold: 01-Jan-10
Enter the date of collection: 10-Jan-10
Enter the first name of the collector: Brian
Enter the last name of the collector: Smith
Enter your staff ID : 5
```

Example Output (Database should also be updated)

```
Product ID 1049 stock is now at 20.
Product ID 1048 stock is now at 48.
```

4.1.3 Delivery Orders (Method option3)

If an order is a delivery order (OrderType "Delivery"), it defaults to uncompleted. In addition to the information needed for all orders, It is also necessary to find out and store:

- The date (same formatting as above) that the item will be delivered.
- The first and last name of the person receiving the order.
- The house, street and city that the delivery is going to.

This option should be implemented using the method 'option3'. It should take a Database Connection (conn), an int array of productIDs that belong to the order, an int array of quantities (the index of the quantity array corresponds to the index of a matching productID in productIDs), the date which the order was made (as a string), the date of the delivery, the first and last name of the recipient as strings, the house, street and city of the address as strings and the ID of the staff member as an int. Any dates should correspond to the default Oracle Date formatting (i.e. 'DD-Mon-YY' such

as '09-Nov-17').

The below is an example of the formatting to be expected when running Option 3:

Example Input

```
Enter your choice: 3
Enter a product ID: 1049
Enter the quantity sold: 5
Is there another product in the order?: Y
Enter a productID: 1048
Enter the quantity sold: 2
Is there another product in the order ?: N
Enter the date sold: 01-Jan-10
Enter the date of collection: 10-Jan-10
Enter the first name of the recipient: Brian
Enter the last name of the recipient: Smith
Enter the house name/no : Corner Cottage
Enter the street: End Lane
Enter the City: Liverpool
Enter your staff ID : 5
```

Example Output (Database should also be updated)

```
Product ID 1049 stock is now at 20.
Product ID 1048 stock is now at 48.
```

4.2 Biggest Sellers (Method option4)

The store is interested in knowing which items are their biggest sellers of all time by value. Produce a comma separated output that lists the product id, the item description and the total amount of money made by selling them (from all orders ever), sorted in descending order. List one product per line.

This option should be implemented using the method 'option4'. It should take a Database Connection (conn) as a parameter.

The below is an example of the formatting to be expected when running Option 4:

Example Input

```
Enter your choice: 4
```

Example Output

```
ProductID,  ProductDesc,    TotalValueSold
2049,       Oak Chair,      £50000
108,        Pine Chair,     £49323
1078,       Turquoise Lamp, £47934
```

4.3 Reserved Stock (Method option5)

While the store reserves items for in-store collection, it places a limit on how long it will wait for a customer. If the order is uncompleted (i.e. not picked up) and is a collection order, then for a given date the program must (a) identify all orders that have a *collection date* 8 days or older than the provided date, (b) re-add the items from the order to the product stock amount and (c) delete the order and any data pertaining to that order (we will assume someone else will take care of the customers refund). Write an option that given a date, finds any such violations of the rule and applies the above process. As output you should print out the order id of the orders that have been canceled.

This option should be implemented using the method 'option5'. It should take a Database Connection (conn) and a target date as a string parameter. This string should correspond to the default Oracle Date formatting (i.e. 'DD-Mon-YY' such as '09-Nov-17').

The below is an example of the formatting to be expected when running Option 5:

Example Input

```
Enter your choice: 5
Enter the date: 07-Nov-17
```

Example Output (Database should also be updated)

```
Order 1034 has been cancelled
Order 1078 has been cancelled
```

4.4 Staff Analytics

4.4.1 Staff Life-Time Success (Method option6)

The staff employee lifetime awards are coming up! Obtain a list of staff who have sold at least £50000 of items during the store's lifetime and print their names alongside the amount. Print one employee per line, and sort in descending order by the total value sold.

This option should be implemented using the method 'option6'. It should take a Database Connection (conn) as a parameter.

The below is an example of the formatting to be expected when running Option 6:

Example Input

```
Enter your choice: 6
```

Example Output

```
EmployeeName,  TotalValueSold
Brian Jackson, £51042
Susan Smith,   £50001
```

4.4.2 Staff Contribution (Method option7)

The stores wishes to know who is responsible for selling the most of the highest rated products. For any item that has sold over £20000 in the store's lifetime, print the name of any member of staff who has sold at least one of any of these products, alongside the amount they have sold of each of these products (0 is possible), one employee per line.

The output should be formatted such that it lists the quantity of each of the products sold by that employee sold over the store lifetime, with a heading line that lists the product IDs. Sort the results in descending order by the total monetary value of the high-selling products only (greater than £20000 in the store's lifetime) sold by each employee.

This option should be implemented using the method 'option7'. It should take a Database Connection (conn) as a parameter.

The below is an example of the formatting to be expected when running Option 7:

Example Input

```
Enter your choice: 7
```

Example Output

```
EmployeeName,   Product 1489, Product 2078, Product 189
Brian Jackson,  42,           0,            92
Susan Smith,    0,            18,           100
Peter Williams, 2,            15,           46
```

4.4.3 Employees of the Year (Method option8)

The store CEO has noted the great success of their employees this year, and decides to hold an employee of the year ceremony. They wish to find the names of staff who have sold at least £30000 of products in a specified year, and sold at least one of each of the items that have sold over £20000 worth of product for the specified year. Print out the employees' first name and last name, one employee per line.

This option should be implemented using the method 'option8'. It should take a Database Connection (conn) as a parameter and a target year as an integer parameter (the integer will be in YYYY format, not YY).

The below is an example of the formatting to be expected when running Option 8:

Example Input

```
Enter your choice: 8
Enter the year: 2017
```

Example Output

```
Brian Jackson
Susan Smith
Peter Williams
```

5 What constitutes a good answer?

Think about good security principles, use prepared statements where you can.

You should handle errors appropriately! This means thinking about what potential errors might occur, and catching them when they do.

Good answers are thoroughly tested answers!

You should favour using SQL over processing data in Java where possible.

For all of your answers, supply appropriate comments (and description in README.md) to describe how your code/SQL works. In addition, as with all programming, you should adhere to programming best practices with respect to formatting etc.

6 Final remarks

Please make sure your programs work on daisy. Keep a copy of everything you submit! You may discuss with fellow students the material covered in lectures and seminars, but you are not allowed to collaborate on the assignment. The University of Warwick takes plagiarism seriously, and penalties will be incurred if any form of plagiarism is detected. Copying, or basing your work on, solutions written by people who have not taken this module is also counted as plagiarism. This includes material that has been downloaded from the internet.

Good luck!