Rushi Samirbhai Patel RS397441@dal.ca
B00886157

Assignment 5

Problem 1:

SQL Queries for Sales Database (I have also attached the sql file for your reference):

a) **Which customers are in a different city than their sales representative?**

select customerName from customers table1 join employees table2 on table1.salesRepEmployeeNumber = table2.employeeNumber join offices table3 on table2.officeCode = table3.officeCode where table1.city != table3.city;

b) **Which orders included sales that are below the manufacturer's suggested retail price (MSRP)?**

select orderNumber from orderdetails table1 join products table2 on table1.productCode = table2.productCode where table1.priceEach<table2.MSRP group by table1.orderNumber;

c) **Which product line has the highest profit margin in 2003. Include the profit margin. The profit of an item is the sales of the item less the cost of the item. The profit margin is the profit divided by the total base cost.**

```
with table1 as
(select orderdetails.productCode, sum(priceEach * quantityOrdered) as sales,
sum(quantityOrdered) as quantity from orderdetails
join orders on orders.orderNumber = orderdetails.orderNumber where
year(orderDate)=2003 group by orderdetails.productCode)
select productLine, (sales - ( table1.quantity * products.buyPrice))/(table1.quantity *
products.buyPrice) as Profit_Margin from products
join table1
on products.productCode = table1.productCode
order by profit_margin desc limit 1;
```

d) **List the 5 employees with the highest sales in 2004. Include their total sales values and ensure that the top seller is first in the list.**

```
with SalesTable as
(select orderNumber, (quantityOrdered * priceEach) as totalSales from orderdetails),
CustomerTable as
(select orders.customerNumber, final.totalSales from orders join SalesTable as final
on orders.orderNumber = final.orderNumber where year(orders.orderDate) = 2004),
EmployeeTable as
(select customers.salesRepEmployeeNumber, CustomerTable.totalSales from customers
join CustomerTable
on CustomerTable.customerNumber = customers.customerNumber)
select concat(employees.firstName," ",employees.lastName) as Employee_Name,
sum(EmployeeTable.totalSales)as Total_Sales from employees join EmployeeTable
on employees.employeeNumber = EmployeeTable.salesRepEmployeeNumber
group by employees.employeeNumber order by sum(EmployeeTable.totalSales) desc
limit 5;
```

e) Which employees had the value of their 2004 orders exceed the value of their 2003 orders?

```
with SalesTable2004 as
(select orderNumber, (quantityOrdered * priceEach) as totalSales from orderdetails),
CustomerTable2004 as
(select orders.customerNumber, SalesTable2004.totalSales from orders join
SalesTable2004
on orders.orderNumber = SalesTable2004.orderNumber where year(orders.orderDate) =
2004),
EmployeeTable2004 as
(select customers.salesRepEmployeeNumber, CustomerTable2004.totalSales from
customers
join CustomerTable2004
on CustomerTable2004.customerNumber = customers.customerNumber),
Table2004 as
(select employees.employeeNumber, concat(employees.firstName,"
",employees.lastName) as employee_name, sum(EmployeeTable2004.totalSales) as
totalSales from employees join EmployeeTable2004
on employees.employeeNumber = EmployeeTable2004.salesRepEmployeeNumber
group by employees.employeeNumber order by sum(EmployeeTable2004.totalSales)),
SalesTable2003 as
(select orderNumber, (quantityOrdered * priceEach) as totalSales from orderdetails),
CustomerTable2003 as
```

```
(select orders.customerNumber, SalesTable2004.totalSales from orders join
SalesTable2004
on orders.orderNumber = SalesTable2004.orderNumber where year(orders.orderDate) =
2003),
EmployeeTable2003 as
(select customers.salesRepEmployeeNumber, CustomerTable2003.totalSales from
customers
join CustomerTable2003
on CustomerTable2003.customerNumber = customers.customerNumber),
Table2003 as
(select employees.employeeNumber ,concat(employees.firstName,"
",employees.lastName) as employee_name, sum(EmployeeTable2003.totalSales)as
totalSales from employees join EmployeeTable2003
on employees.employeeNumber = EmployeeTable2003.salesRepEmployeeNumber
group by employees.employeeNumber order by sum(EmployeeTable2003.totalSales))
select Table2004.employeeNumber,Table2004.employee_name,Table2004.totalSales
from Table2004 join Table2003
on Table2003.employeeNumber = Table2004.employeeNumber
where Table2003.totalSales < Table2004.totalSales order by
Table2004.employeeNumber;
```

Problem 2:

Overview

The program extracts the information related to manager and product lines. It will fetch the data from the sales database and report it back into an xml file given a particular period of time. This file can hence be reviewed by the Mini-Me Toy Car company.

Files and External Data:

This solution contains 3 java files along with the connection to the sales database. The java files are:

1. Main.java - This class is the main class where the main method is present that calls the implementation methods.
2. DBConnection.java - This class acts as a util class for all the database operations. It has functions to create a connection to the sales database, read data, update data and close the connection.

3. XMLWriter.java - This class acts as a util class for all the xml operations. It writes the data received from the database in a resultset to an xml file using buffered reader and writer.
4. DataExport.java - The implementation of the solution where the data is gathered by running the query is covered in this class. This class further sends the data to the XMLWriter class to write the information in an xml file.

Database Design and Connection:

The solution requires an active connection to the sales database. To achieve this, I have used jdbc driver and sql connector in a DBConnection utilityl class. I have designed my database connection in my code in such a way that only one instance of connection is created for the entire solution. The benefit of using the singleton design pattern is that it is more efficient and is more easy to handle.

Assumptions:

- The database schema is ready to be used with all the necessary information, where the solution is executed.
- The dates (start date and end date) are given in a proper format : yyyy-mm-dd.
- Full path of the file where the file should be saved must be given as user input along with the extension (.xml).

Code ready for deployment argument:

The Mini-Me Toy Car company's requirement stated the need of the company's operation over a period of time as a human readable report in a xml file. The designed solution fetches the required information from the sales database and structures it in a well designed human readable xml file. The data is reported keeping in mind the human readability with proper spacing and tabs where required. The results can be easily analysed and inferred upon.

The information that is fetched from the database contains various details on the database however the report is limited to only the required information on the manager - manager's name, manager's work location, manager's staff, the number of customers served by the manager's staff and the total sales by staff reporting to the manager. It also contains details on the product line - its name, description, customers who ordered some items of the product line, customer's name and the total value of the product lines. The information can be

The code quality is upto the standards and is following the SOLID principles. The efficiency and performance of the solution is good as well because of the design pattern chosen to achieve this solution. Each issue, error and exceptions are handled in the code and the code is now ready to be deployed.

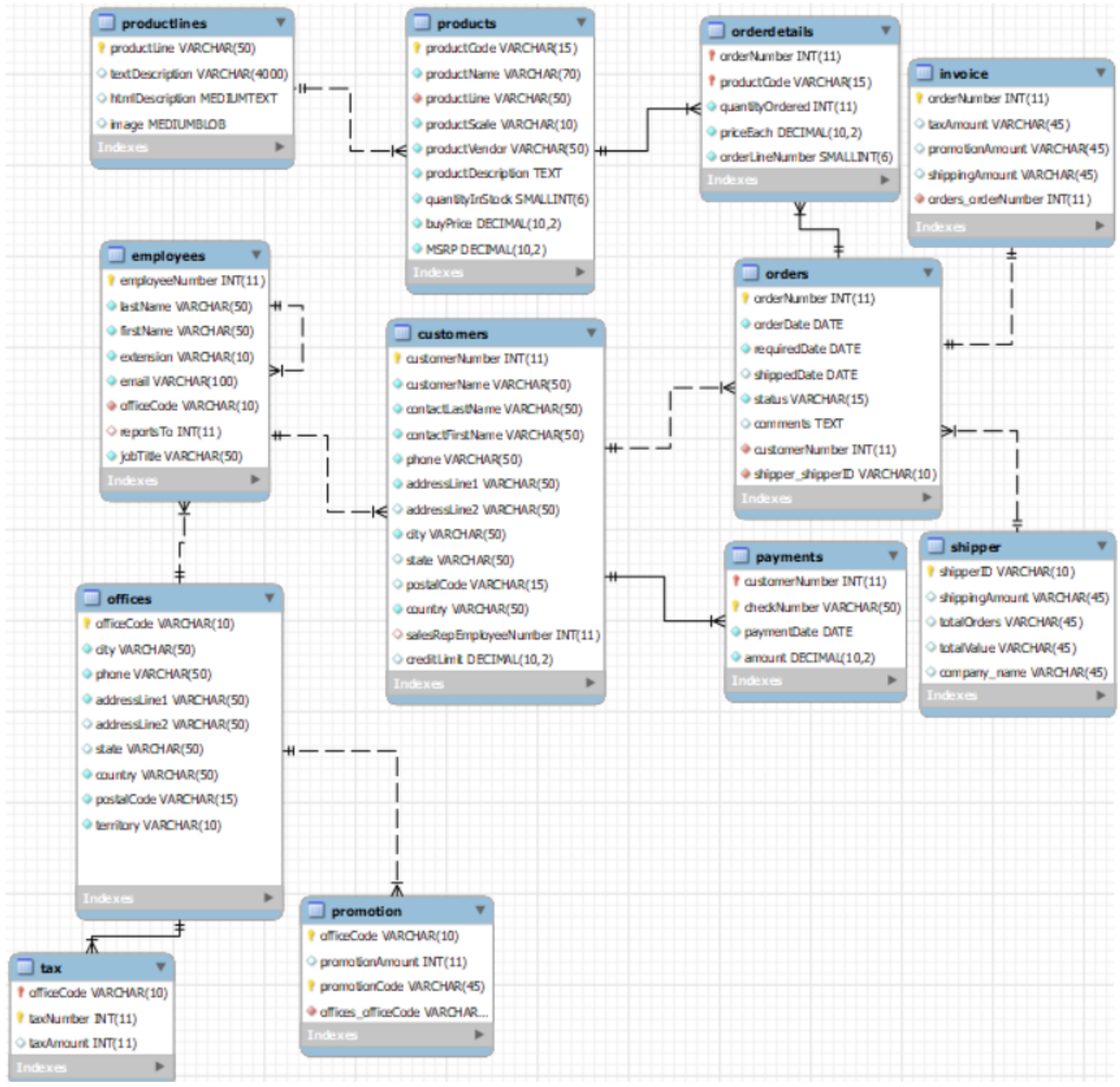
Problem 3:

Overview:

The database design is re-designed to include the tax amount, promotional discount and shipping charges of an order.

Database Design:
Design is displayed on the next page please.

Rushi Samirbhai Patel RS397441@dal.ca
B00886157



Sample SQL to create the structures and populate tables (SQL File is also attached for your reference):

```
use rspatel;
/*My changes for Assignment 5 - Problem 3*/

create table tax(`officeCode` VARCHAR(10) NOT NULL,
  `taxNumber` INT NOT NULL,
  `taxAmount` INT NULL,
  PRIMARY KEY (`officeCode`, `taxNumber`),
  CONSTRAINT `officeCode`
    FOREIGN KEY (`officeCode`)
```

```
   REFERENCES `rspatel`.`offices` (`officeCode`));

create table promotion (`officeCode` VARCHAR(10) NOT NULL,
  `promotionAmount` INT NULL,
  `promotionCode` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`officeCode`, `promotionCode`)
 );

create table invoice (`orderNumber` INT(11) NOT NULL,
  `totalValue` VARCHAR(45) NULL,
  `taxAmount` VARCHAR(45) default 0,
  `promotionAmount` VARCHAR(45) default 0,
  `shippingAmount` VARCHAR(45) default 0,
  `finalValue` VARCHAR(45) NULL,
  PRIMARY KEY (`orderNumber`));

create table shipper (`shipperID` VARCHAR(10) NOT NULL,
 `company_name` VARCHAR(45) NULL,
  `shippingAmount` VARCHAR(45) NULL,
  `totalOrders` VARCHAR(45) NULL,
  `totalValue` VARCHAR(45) NULL,

  PRIMARY KEY (`shipperID`));

alter table orders add column (shipperID VARCHAR(10));

insert into tax values(1, 10, 15);
insert into tax values(2, 11, 16);
insert into tax values(3, 12, 17);
insert into tax values(4, 13, 18);
insert into tax values(5, 14, 19);
insert into tax values(6, 15, 20);
insert into tax values(7, 16, 21);

insert into promotion values(1, 250, 777);
insert into promotion values(2, 300, 888);
insert into promotion values(3, 400, 999);
insert into promotion values(4, 410, 111);
insert into promotion values(5, 420, 222);
insert into promotion values(6, 430, 333);
insert into promotion values(7, 450, 21);

insert into shipper values(1, 'ABC Company',20,3,1000);
insert into shipper values(2, 'XYZ Company',21,2,500);
insert into shipper values(3, 'LKM Company',22,1,100);

update orders set shipperID=1 where orderNumber=10100;
update orders set shipperID=1 where orderNumber=10101;
update orders set shipperID=1 where orderNumber=10102;
update orders set shipperID=2 where orderNumber=10103;
update orders set shipperID=2 where orderNumber=10104;
update orders set shipperID=3 where orderNumber=10105;

/*Populating all the order numbers in the invoice table*/
INSERT INTO invoice (orderNumber)
SELECT orderNumber FROM orders;
```