**מסדי נתונים מטלה 2**

1.1.

-At first I will create a temporary table to calculate the avg of every product:

drop table if EXISTS avgProd;

create temporary table avgProd

select od.ProductID, avg(od.UnitPrice) as avgPrice

from products as p join `order details` as od on p.ProductID = od.ProductID

group by od.ProductID;

-Now I can define my stored procedure using this temporary table:

CREATE DEFINER=`root`@`localhost` PROCEDURE `myPrecuder`(custumerId nchar(5))

BEGIN

select od.OrderID,

case

when (sum(od.Quantity\* od.UnitPrice) - sum(od.Quantity\* ap. avgPrice)) > 0 then "Loss"

when (sum(od.Quantity\* od.UnitPrice) - sum(od.Quantity\* ap. avgPrice)) < 0 then "Profit"

else "Equivalent"

end as "Profit/Loss",

case

when (sum(od.Quantity\* od.UnitPrice) - sum(od.Quantity\* ap. avgPrice)) >= 0 then (sum(od.Quantity\* od.UnitPrice) - sum(od.Quantity\* ap. avgPrice))

else -1\*(sum(od.Quantity\* od.UnitPrice) - sum(od.Quantity\* ap.avgPrice))

end as "Amount"

from orders as o join `order details` as od on od.OrderID=o.OrderID

join products as pr on pr.ProductID = od.ProductID

join avgProd as ap on ap.ProductID=pr.ProductID

where o.CustomerID = custumerId

group by o.OrderID

order by o.OrderID desc;

END

1.2

import java.sql.\*;

import java.util.Scanner;

public class Main{

public static void main(String[] args){

try{

Class.forName("com.mysql.jdbc.Driver");

try(Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "user", "pwd")){

Statement st = con.createStatement();

system.out.println(“please insert the id(not more then 5 chars)”);

Scanner myScanner = new Scanner(System.in);

string custumerId = myScanner.nextLine();

ResultSet rSet = st.executeQuery("call myPrecuder(custumerId);");

int ColumnsNum = rSet.getMetaData().getColumnCount();

while (rSet.next()){

for (int i = 1; col <= ColumnsNum; i++){

System.out.print(rSet.getString(i) + " ");

}

System.out.println();

}

}} catch (Exception ex){ex.printStackTrace();}

}

}

2. DELIMITER//

create trigger del

after delete on employees as e

for each row

begin

delete from employeeterritories

where EmployeeID not in (e.EmployeeID);

end//

DELIMITER ;

3.1. SELECT

OrderID, RequiredDate

FROM

Orders

WHERE

ShippedDate Is NULL

() AND RequiredDate < NOW

3.2. Returns all the orders id and the required date which they shipped date is unknown or they didn’t shipped and their required date to arrive is already passed, means the date is "smaller" then the date today.

4.1. The candidate keys are: {A,B},{A,C},{B,D},{C,D}.

4.2. The normalized level is NF3 explanation:

NF1 – By the assumption that all the columns has atomic values(as we did on practice).

NF2 – Because there are no non-Prime values.

NF3 – Because there are no non-Prime values.

Stops here because one of the dependencies is D -> A and D is not a super-key