

LiveStock

CSE 305: Assignment 2



Wendy Zheng: wendy.zheng@stonybrook.edu

Ryan Goepfert: [ryan.goepfert@stonybrook.edu](mailto:ryan.goepfert@stonybrook.edu)

Yuliya Astapova: yuliya.astapova@stonybrook.edu

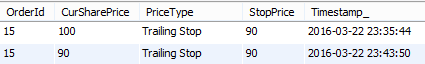
**TABLES WITH DEMO DATA**

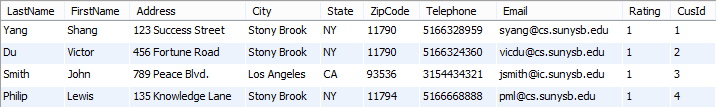
Our database auto-increments all id numbers, so the ids given in the demo data differ from the ones in our database. A customer with the id ‘XXX-XX-XXXX’ now has id ‘X’. Likewise, each employee id, account number, order id, and transaction id auto-increment and are unique. As such, they may differ from the demo data. SSN, telephone number, and credit card no longer have hyphens. All dates have a time set to the time the data was entered, as our database stores both date and time for such fields.

**Account**

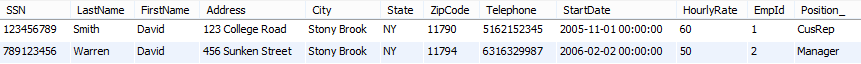
****

**ConditionalPriceHistory**

****

**Customer**

**Employee**

****

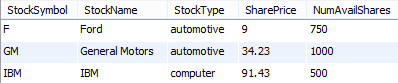
**Order**

****

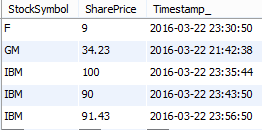
**Portfolio**

****

**Stock**

****

**StockPriceHistory**

****

**Transact**

****

**Transactions**

All fields and values starting with ‘:’ are parameters to be entered by the user. The argument placeholder denotes the required input data. Please note that the manager level transactions dealing with reports have mostly empty views because none of the orders have been recorded yet; therefore they are not considered “reviewed” and do not appear on the resulting table.

**Manager-Level Transactions**

* Set the share price of a stock (for simulating market fluctuations in a stock's share price)

UPDATE STOCK

SET SharePrice = :share\_price

WHERE StockSymbol = :stock\_symbol

UPDATE STOCK  
SET SharePrice = 10  
WHERE StockSymbol = 'GM'

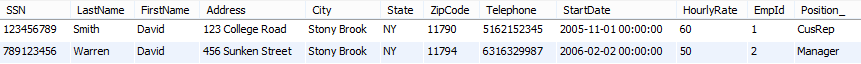


* Add, Edit and Delete information for an employee

INSERT INTO EMPLOYEE(  
SSN, LastName, FirstName, Address, City, State, ZipCode, Telephone, StartDate, HourlyRate, Position)

VALUES (:ssn, :last\_name, :first\_name, :address, :city, :state, :zipcode, :telephone, :start\_date, :hourly\_rate, :position)

INSERT INTO EMPLOYEE(  
SSN, LastName, FirstName, Address, City, State, ZipCode, Telephone, StartDate, HourlyRate, Position\_)

****VALUES ('123456789', 'Smith', 'David', '123 College Road', 'Stony Brook', 'NY', '11790', '5162152345', '2005-11-01 00:00:00', 60, 'CusRep');

UPDATE EMPLOYEE

SET :field = :value

WHERE EmpId = :emp\_id

UPDATE EMPLOYEE

SET FirstName = 'Pablo'

WHERE EmpId = 1



DELETE FROM EMPLOYEE

WHERE EmpId = :emp\_id

DELETE FROM Employee

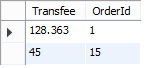
WHERE EmpId = 1



* Obtain a sales report for a particular month

SELECT T.Transfee, T.OrderId

FROM TRANSACT T

 WHERE YEAR(T.Timestamp) = :year

AND MONTH(T.Timestamp) = :month

SELECT T.Transfee, T.OrderId

FROM TRANSACT T

WHERE YEAR(T.Timestamp) = 2016

AND MONTH(T.Timestamp) = 03

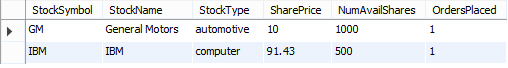
* Produce a comprehensive listing of all stocks

SELECT S.\*, COUNT (O.StockSymbol) AS OrdersPlaced

FROM STOCK S, ORDER O

WHERE S.StockSymbol = O.StockSymbol

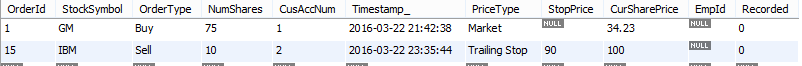
GROUP BY S.StockSymbol



* Produce a list of orders by stock symbol or by customer name

SELECT \*

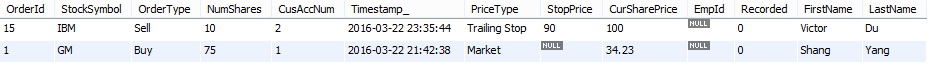
FROM ORDER\_ O

ORDER BY O.StockSymbol

SELECT O.\*, C.FirstName, C.LastName

FROM ORDER O, CUSTOMER C

WHERE O.CusAccNum = C.CusId

 ORDER BY C.LastName, C.FirstName

* Produce a summary listing of revenue generated by a particular stock, stock type, or customer

CREATE VIEW STOCKREVENUE (StockSymbol, TotalRevenue) AS

SELECT O.StockSymbol, SUM(T.TransFee)

FROM Order O, Transact T

WHERE O.Recorded = 1 AND O.OrderId = T.OrderId

GROUP BY O.StockSymbol



CREATE VIEW STOCKTYPEREVENUE (StockType, TotalRevenue) AS

SELECT S.StockType, SUM(T.TransFee)

FROM Stock S, Transact T, Order O

WHERE O.StockSymbol = S.StockSymbol AND O.Recorded = 1

AND O.OrderID=T.OrderId

GROUP BY S.StockType



CREATE VIEW CUSTOMERREVENUE(CusAccNum, TotalRevenue) AS

SELECT O.CusAccNum, SUM(T.TransFee) AS TotalRevenue  
FROM Order O, Transact T  
WHERE O.Recorded = 1 AND O.OrderID = T.OrderId  
GROUP BY O.CusAccNum



* Determine which customer representative generated most total revenue

CREATE VIEW EMPLOYEEREVENUE(EmpId, TotalRevenue) AS

SELECT O.EmpId, SUM(T.TransFee) AS TotalRevenue

FROM ORDER O, Transact T

WHERE O.OrderID=T.OrderId AND O.Recorded = 1 AND O.EmpID <> 0

GROUP BY O.EmpId



SELECT E.EmpId, E.TotalRevenue

FROM EMPLOYEEREVENUE E

WHERE  E.TotalRevenue  =

(SELECT MAX(E.TotalRevenue)

FROM EMPLOYEEREVENUE E)



* Determine which customer generated most total revenue

SELECT C.CusAccNum, C.TotalRevenue

FROM CUSTOMERREVENUE C

WHERE C.TotalRevenue =

(SELECT MAX(C.TotalRevenue)

FROM CUSTOMEREVENUE C)



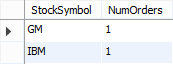
* Produce a list of most actively traded stocks

CREATE VIEW MostTraded(StockSymbol, NumOrders) AS

SELECT O.StockSymbol, COUNT(\*) AS NumOrders

FROM Order O

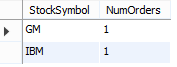
GROUP BY O.StockSymbol



SELECT M.StockSymbol, M.NumOrders

FROM MostTraded M

ORDER BY ASC NumOrders

 LIMIT 25

**Customer-Representative-Level Transactions**

* Record an order

UPDATE ORDER O

SET Recorded = 1

WHERE O.OrderId = :order\_id

UPDATE ORDER O

SET Recorded = 1

WHERE O.OrderId = 1



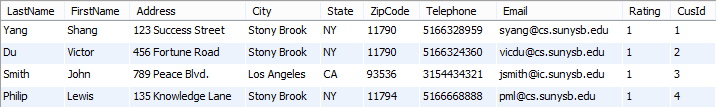
* Add, Edit and Delete information for a customer

INSERT INTO CUSTOMER(  
LastName, FirstName, Address, City, State, ZipCode, Telephone, Email, Rating, CusId)

VALUES (  
:last\_name, :first\_name, :address, :city, :state, :zipcode, :telephone, :email, :rating, :cus\_id)

INSERT INTO Customer(  
LastName, FirstName, Address, City, State, ZipCode, Telephone, Email, Rating)

VALUES (  
'Yang', 'Shang', '123 Success Street', 'Stony Brook', 'NY', '11790', '5166328959', 'syang@cs.sunysb.edu', 1);

****

UPDATE CUSTOMER C

SET field = :value

WHERE CusId = :cus\_id

UPDATE CUSTOMER C

SET C.Address = '600 Circle Road'

WHERE c.CusId = 1

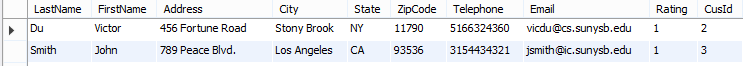


DELETE FROM CUSTOMER

WHERE CusId = :cus\_id

DELETE FROM CUSTOMER

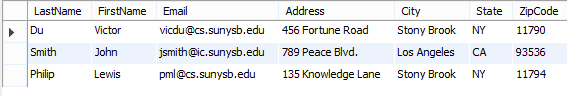
WHERE CusId = 1



* Produce customer mailing lists

SELECT LastName, FirstName, Email, Address, City, State, ZipCode

FROM CUSTOMER



* Produce a list of stock suggestions for a given customer (based on that customer's past orders)

CREATE VIEW Suggest(StockSymbol, SharePrice, NumAvailShares, StockType) AS

SELECT S.StockSymbol, S.SharePrice, S.NumAvailShares, S.StockType

FROM Stock S, Order\_ O

WHERE (O.CusAccNum = :\_cusid AND O.StockSymbol = S.StockSymbol)

CREATE VIEW Suggest(StockSymbol, SharePrice, NumAvailShares, StockType) AS

SELECT S.StockSymbol, S.SharePrice, S.NumAvailShares, S.StockType

FROM Stock S, Order\_ O

WHERE (O.CusAccNum = 1 AND O.StockSymbol = S.StockSymbol)



**Customer-Level Transactions**

* A customer's current stock holdings

SELECT P.AccNum, P.StockSymbol, P.NumShares, P.Stop, P.StopPrice

FROM Portfolio P, Account A, Customer C

WHERE P.AccNum = A.AccNum

AND A.CusId = C.CusId

AND C.CusId = :customer\_id



* The share-price and trailing-stop history for a given conditional order

SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp

FROM CONDITIONALPRICEHISTORY P

WHERE P.OrderId = :\_orderid AND P.PriceType = ‘Trailing Stop’ AND P.StopPrice > 0

SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp

FROM CONDITIONALPRICEHISTORY

WHERE P.OrderId = 15 AND P.PriceType = ‘Trailing Stop’ AND P.StopPrice > 0



* The share-price and hidden-stop history for a given conditional order

SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp

FROM CONDITIONALPRICEHISTORY P, ORDER O

WHERE P.OrderId = :\_orderid AND P.PriceType = “Hidden Stop’ AND P.StopPrice > 0

SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp

FROM CONDITIONALPRICEHISTORY P, ORDER O

WHERE P.OrderId = 15 P.PriceType = “Hidden Stop’ AND P.StopPrice > 0



* The share-price history of a given stock over a certain period of time (e.g., past six months)

SELECT P.SharePrice, P.TimeStamp

FROM STOCKPRICEHISTORY P

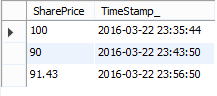
WHERE P.StockSymbol = :stock\_symbol   
 AND MONTH(TIMEDIFF(NOW(), P.TimeStamp)) < :\_month\_past

SELECT P.SharePrice, P.TimeStamp

FROM STOCKPRICEHISTORY P

WHERE P.StockSymbol = ‘IBM’

AND MONTH(TIMEDIFF(NOW(), P.TimeStamp)) < 6



* A history of all current and past orders a customer has placed.

SELECT O.Timestamp, O.CusAccNum, O.OrderType, O.StockSymbol, O.NumShares, O.CurSharePrice, O.PriceType

FROM Order\_ O, Customer C, Account A

WHERE O.CusAccNum = A.AccNum

AND A.CusId = C.CusId AND C.CusId = :customer\_id

SELECT O.Timestamp, O.CusAccNum, O.OrderType, O.StockSymbol, O.NumShares, O.CurSharePrice, O.PriceType

FROM Order\_ O, Customer C, Account A

WHERE O.CusAccNum = A.AccNum

AND A.CusId = C.CusId AND C.CusId = 4



* Stocks available of a particular type and most-recent order info

SELECT O.\*

FROM STOCK S, Order\_ O

WHERE S.StockType LIKE '%:\_stocktype%' AND O.StockSymbol = S.StockSymbol

SELECT O.\*

FROM STOCK S, Order\_ O

WHERE S.StockType LIKE '%auto%' AND O.StockSymbol = S.StockSymbol



* Stocks available with a particular keyword or set of keywords in the stock name, and most-recent order info.

SELECT O.\*

FROM STOCK S, Order\_ O

WHERE S.StockSymbol LIKE '%:\_stocksymbol%' AND O.StockSymbol = S.StockSymbol

SELECT O.\*

FROM STOCK S, Order\_ O

WHERE S.StockSymbol LIKE '%I%' AND O.StockSymbol = S.StockSymbol



SELECT O.\*

FROM ORDER\_ O, CUSTOMER C, ACCOUNT\_ A

WHERE O.CusAccNum = A.AccNum AND A.CusId = C.CusId AND C.CusId = 4

AND MONTH(TIMEDIFF(NOW(), O.Timestamp)) <= 3



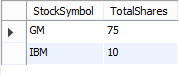
* Best-Seller list of stocks generated based on what stocks have been sold the most.

CREATE VIEW BestSellers(StockSymbol, TotalShares) AS

SELECT O.StockSymbol, SUM(O.NumShares) AS TotalShares

FROM Order O

GROUP BY O.StockSymbol

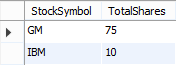


SELECT B.StockSymbol, B.TotalShares

FROM BestSellers B

ORDER BY DESC TotalShares

LIMIT 25



* Personalized stock suggestion list. This finds stocks that are of the same type that the customer currently owns and stocks that the customer once owned, but no longer does.

CREATE VIEW Suggest(StockSymbol, SharePrice, NumAvailShares, StockType) AS

SELECT S.StockSymbol, S.SharePrice, S.NumAvailShares, S.StockType

FROM Stock S, Order\_ O

WHERE (O.CusAccNum = 1 AND O.StockSymbol = S.StockSymbol)

