3.1

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

DELIMITER //

CREATE PROCEDURE updateStockPrice(IN stockPrice INTEGER, stockSymbol CHAR(20))

BEGIN

UPDATE Stock

SET Stock.PricePerShare = stockPrice

WHERE Stock.StockSymbol = stockSymbol;

END//

Execution: Call updateStockPrice(400, 'F');

#Add, Edit and Delete information for an employee

#ADD

DELIMITER //

CREATE PROCEDURE addEmployee(IN ID INTEGER, SSN INTEGER, StartDate DATE, HourlyRate INTEGER)

BEGIN

INSERT INTO Employee

VALUES(ID, SSN, StartDate, HourlyRate);

END//

Execution: Call addEmployee(3, 111111111, '2016-03-23', 90);

#EDIT

DELIMITER //

CREATE PROCEDURE editEmployeeSSN(IN ID INTEGER, SSN INTEGER)

BEGIN

UPDATE Employee

SET Employee.SSN = SSN

WHERE Employee.ID = ID;

END//

Execution: call editEmployeeSSN(3, 222222222);

DELIMITER //

CREATE PROCEDURE editEmployeeStartDate(IN ID INTEGER, StartDate DATE)

BEGIN

UPDATE Employee

SET Employee.StartDate = StartDate

WHERE Employee.ID = ID;

END//

Execution: call editEmployeeStartDate(3, '2001-01-03');

DELIMITER //

CREATE PROCEDURE editEmployeeHourlyRate(IN ID INTEGER, HourlyRate INTEGER)

BEGIN

UPDATE Employee

SET HourlyRate = HourlyRate

WHERE Employee.ID = ID;

END//

Execution: call editEmployeeHourlyRate(3, 100);

#delete

DELIMITER //

CREATE PROCEDURE deleteEmployee(IN ID INTEGER)

BEGIN

DELETE FROM employee

Where ID = ID;

END//

Execution: call deleteEmployee(3);

#Obtain a sales report for a particular month

DELIMITER //

CREATE PROCEDURE monthlySalesReport(IN month INTEGER)

BEGIN

SELECT StockOrder.\*

FROM StockOrder

WHERE Month(StockOrder.DateTime) = month;

END//

Execution: call monthlySalesReport(2);

#Produce a comprehensive listing of all stocks

DELIMITER //

CREATE PROCEDURE stockListing()

BEGIN

SELECT\* FROM Stock;

END//

Execution: call stockListing();

#Produce a list of orders by stock symbol

DELIMITER //

CREATE PROCEDURE stockListingBySymbol()

BEGIN

SELECT\* FROM Stock

ORDER BY Stock.StockSymbol;

END //

Execution: call stockListingBySymbol ();

#Determine which customer representative generated most total revenue

DELIMITER //

CREATE PROCEDURE mostRevenue\_CustomerRepresentative()

BEGIN

SELECT E.ID AS 'Employee ID', (R.PricePerShare - S.PricePerShare) \* O.NumShares AS MaxRevenue

FROM Employee E, Transaction R, StockOrder O, Stock S, Trade T

WHERE T.StockId = S.StockSymbol

AND T.TransactionId = R.ID AND T.OrderId = O.ID

AND T.BrokerId = E.ID

ORDER BY MaxRevenue DESC LIMIT 1;

END //

Execution: call mostRevenue\_CustomerRepresentative();

#Determine which customer generated most total revenue

DELIMITER //

CREATE PROCEDURE customer\_mostRevenue()

BEGIN

SELECT C.ID AS 'Client ID', (R.PricePerShare - S.PricePerShare) \* O.NumShares AS MaxRevenue

FROM Client C, Transaction R, StockOrder O, Stock S, Trade T, Account A

WHERE T.StockId = S.StockSymbol AND T.AccountId = A.ID

AND T.TransactionId = R.ID AND T.OrderId = O.ID

AND A.ClientID = C.ID

ORDER BY MaxRevenue DESC LIMIT 1;

END //

Execution: call customer\_mostRevenue();

#Produce a list of most actively traded stocks

DELIMITER //

CREATE PROCEDURE activeStocks()

BEGIN

SELECT Trade.StockId

FROM Trade

GROUP BY Trade.StockId

ORDER BY COUNT(Trade.StockId) DESC;

END //

Execution: call activeStocks();

3.2

# Record an order

DELIMITER //

CREATE PROCEDURE recordOrder(IN NumShares INTEGER, IN ID INTEGER, IN PriceType CHAR(20), IN OrderType CHAR(4))

BEGIN

INSERT INTO stockorder(NumShares, ID, PriceType, OrderType)

VALUES(NumShares, ID, PriceType, OrderType);

END//

Execution: call recordOrder(300, '2014-03-05', 'TrailingStop', 'buy', 0.15, 0);

#Add customer

DELIMITER //

CREATE PROCEDURE addCustomer(IN FirstName CHAR(20), IN LastName CHAR(20), IN Address CHAR(20),

IN ZipCode INTEGER, IN City CHAR(20), IN State CHAR(20), IN Telephone BIGINT,

Email CHAR(32), Rating INTEGER, CreditCardNumber BIGINT,ID INTEGER)

BEGIN

INSERT ignore into location(ZipCode, City,State)

values(ZipCode, City, State);

INSERT INTO Person(FirstName, LastName, Address, ZipCode, Telephone, SSN)

VALUES(FirstName, LastName, Address, ZipCode, Telephone, ID);

INSERT INTO Client(Email, Rating, CreditCardNumber, ID)

VALUES(Email, Rating, CreditCardNumber, ID);

INSERT IGNORE INTO Account(DateOpened, ClientID)

VALUES(NOW(), ID);

END//

Execution: call addCustomer('David', 'Liu', '14 bluetop Rd', 11733, 'stonybrook', 'NY', 9176297036, 'li43@gmail.com', 1, 5216021006125416, 888888888);

# Edit customer information

DELIMITER //

CREATE PROCEDURE editCustomer(E CHAR(32), R INTEGER, CCN BIGINT,I INTEGER)

BEGIN

UPDATE Client

SET Email = E, Rating = R,

CreditCardNumber = CCN,

ID = I;

END//

call editCustomer('1212@stonybrook.edu', 3, 1238283295995321, 221213123);

# Delete people

DELIMITER //

CREATE PROCEDURE deleteCustomer(CustID INTEGER)

BEGIN

DELETE FROM Client

WHERE ID = CustID;

END//

Call deleteCustomer(444444444);

# Produce a list of customer mailing list

DELIMITER //

CREATE PROCEDURE getMailingList()

BEGIN

SELECT Email FROM Client;

END//

Execution: call getMailingListI()