1. SQL DDL statements to create all the tables in your database scheme.

```
CREATE TABLE Stock (
      StockSymbol       VARCHAR(5) NOT NULL,
      StockName         VARCHAR(20) NOT NULL,
      StockType         VARCHAR(20),
      SharePrice        FLOAT(2) NOT NULL,
      NumAvailShares    INTEGER NOT NULL,
      PRIMARY KEY       (StockSymbol),
      UNIQUE            (StockName)
);

CREATE TABLE Employee (
      SSN               CHAR(9) NOT NULL,
      LastName          VARCHAR(20),
      FirstName         VARCHAR(20),
      Address           VARCHAR(50),
      City              VARCHAR(20),
      State             VARCHAR(20),
      ZipCode           CHAR(5),
      Telephone         CHAR(10),
      StartDate         DATETIME,
      HourlyRate        FLOAT(2),
      EmpId             INTEGER AUTO_INCREMENT NOT NULL,
Position_         VARCHAR(7) NOT NULL,
PRIMARY KEY       (EmpId),
UNIQUE            (SSN)
);

CREATE TABLE Customer (
      LastName          VARCHAR(20) NOT NULL,
      FirstName         VARCHAR(20) NOT NULL,
      Address           VARCHAR(50),
      City              VARCHAR(20),
      State             VARCHAR(20),
      ZipCode           CHAR(5),
      Telephone         CHAR(10),
      Email             VARCHAR(50),
      Rating                  INTEGER NOT NULL,
      CusId             INTEGER AUTO_INCREMENT NOT NULL,
      PRIMARY KEY (CusId)
);

CREATE TABLE Order (
      OrderId                 INTEGER AUTO_INCREMENT,
      StockSymbol       VARCHAR(5),
      OrderType         VARCHAR(4) NOT NULL,
      NumShares         INTEGER NOT NULL,
      CusAccNum         INTEGER DEFAULT 0,
      Timestamp_        DATETIME DEFAULT NOW() NOT NULL,
      PriceType         VARCHAR(15) NOT NULL,
      StopPrice         FLOAT(2) DEFAULT 0,
      StopDiff          FLOAT(2),
      CurSharePrice     FLOAT(2),
```

```
      EmpId             INTEGER DEFAULT 0,
      Recorded          BOOLEAN DEFAULT 0,
      Completed         BOOLEAN DEFAULT 0,
      PRIMARY KEY       (OrderId),
      UNIQUE            (StockSymbol, Timestamp, CusAccNum, EmpId),
      FOREIGN KEY       (StockSymbol) REFERENCES Stock (StockSymbol)
                  ON DELETE SET NULL
                  ON UPDATE CASCADE,
      FOREIGN KEY       (CusAccNum) REFERENCES Account (AccNum)
                  ON DELETE SET NULL
                  ON UPDATE CASCADE,
      FOREIGN KEY       (EmpId) REFERENCES Employee (EmpId)
                  ON DELETE SET NULL
                  ON UPDATE CASCADE
);

CREATE TABLE Transact (
      Id                INTEGER AUTO_INCREMENT,
      OrderId           INTEGER,
      TransFee          FLOAT(2),
      TimeStamp_        DATETIME DEFAULT NOW() NOT NULL,
      PricePerShare     FLOAT(2),
      PRIMARY KEY       (Id),
      FOREIGN KEY       (OrderId) REFERENCES Order (OrderId)
                  ON DELETE SET NULL
                  ON UPDATE CASCADE
 );

CREATE TABLE Login (
      Usr               VARCHAR(20) NOT NULL,
      Pwd               VARCHAR(20) NOT NULL,
      AccType           INTEGER NOT NULL,
      Id                INTEGER NOT NULL,
      PRIMARY KEY       (Usr)
);

CREATE TABLE Account (
      AccNum            INTEGER AUTO_INCREMENT NOT NULL,
      AccCreDate        DATETIME,
      CreditCNum        VARCHAR(16) NOT NULL,
      CusId             INTEGER NOT NULL,
      PRIMARY KEY       (AccNum),
      FOREIGN KEY       (CusId) REFERENCES Customer (CusId)
                  ON DELETE NO ACTION
                  ON UPDATE CASCADE
);

CREATE TABLE ConditionalPriceHistory (
      OrderId           INTEGER,
      CurSharePrice     FLOAT(2),
      PriceType         VARCHAR(15) NOT NULL,
      StopPrice         FLOAT(2),
      Timestamp_        DATETIME DEFAULT NOW(),
      PRIMARY KEY (OrderId, PriceType, Timestamp_),
```

```
      FOREIGN KEY       (OrderId) REFERENCES Order_ (OrderId)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
);

CREATE TABLE StockPriceHistory (
      StockSymbol       VARCHAR(5),
      SharePrice        FLOAT(2),
      Timestamp_        DATETIME DEFAULT NOW(),
      PRIMARY KEY(StockSymbol, Timestamp_),
      FOREIGN KEY(StockSymbol) REFERENCES Stock (StockSymbol)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
);

CREATE TABLE Portfolio (
      AccNum            INTEGER,
      StockSymbol       CHAR(5),
      NumShares         INTEGER,
      Stop_             VARCHAR(8) NOT NULL,
      StopPrice         FLOAT(2),
      PRIMARY KEY       (AccNum, StockSymbol),
      FOREIGN KEY       AccNum) REFERENCES Account_ (AccNum)
                  ON DELETE NO ACTION
                  ON UPDATE CASCADE,
      FOREIGN KEY       (StockSymbol) REFERENCES Stock (StockSymbol)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
);

-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------------------
```

2. SQL DML statement to insert records into all the tables in your database. You are free to insert whatever you wish, but make sure each table has at least 10 records/rows.

```
INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Tomar', 'Prashant', '5510 Alson Drive', 'Norfolk', 'VA', '23508',
'9136055136', 'ptoma001@odu.edu', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Chaudhary', 'Rohit', '5510 Alson Drive', 'Norfolk', 'VA',
'93536', '9134152397', 'rchau004.odu.edu', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Martinez', 'Sarah', '3191 Bradley Parks', 'Samanthafort', 'CT',
'84140', '3952451723', 'danielsdavid@hotmail.com', 1);
```

```
INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Gibson', 'Brenda', '92520 McGuire Ave', 'Schneidermouth', 'IA',
'78506', '2377742976', 'stacy77@gmail.com.com', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Kemp', 'Patricia', '33402 Howard Viaduct', 'Kennethfurt', 'NY',
'99181', '5527212043', 'brianwashington@gmail.com', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Davis', 'Laura', '671 Ward Heights', 'New Denisechester', 'NM',
'78788', '5705259385', 'campbelldaniel@gmail.com', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Wheeler', 'Joe', '222 Mccall Crest', 'Mcgrathbury', 'GA',
'14934', '1993218170', 'davidcolleen@hotmail.com', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Jennings', 'Justin', '69114 Flores Port', 'West Sara', 'DE',
'55309', '5754091899', 'marysmith@gmail.com',  1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Simpson', 'Mark', '658 Cristian Tunnel', 'South Cassandra', 'OH',
'75302', '90168606283', 'yscott@hotmail.com', 1);

INSERT INTO Customer(LastName, FirstName, Address, City, State, ZipCode,
Telephone, Email, Rating)
VALUES ('Harris', 'Justin', '688 Bowers Common', 'Stevenport', 'MN',
'47608', '6956328125', 'hammondelizabeth@yahoo.com', 1);

--------------------------------------------------------------------------
---------------------------------------


VALUES ('2023-08-25 05:28:47', '2301382194187229', 1);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-03-14 07:40:31', '2293461723062408', 8);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-09-26 18:26:53', '3571126580334965', 1);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-06-28 04:54:17', '180085643733986', 2);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-06-11 15:02:44', '4392821460812461', 3);
```

```
INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-12-20 03:04:57', '4921710018546756', 6);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-02-11 03:18:17', '2714202307639604', 7);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-03-17 20:46:21', '375117991515304', 2);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-02-08 21:29:12', '2275018289197747', 4);

INSERT INTO Account_(AccCreDate, CreditCNum, CusId)
VALUES ('2023-02-08 21:29:12', '2275018289197747', 5);


-------------------------------------------------------------------------
-----------------------------------------

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('Rdx', 'Elliott-Huff', 'Automotive', 81.53, 822);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('qgZ', 'Greene PLC', 'Software', 82.45, 779);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('UjT', 'Lambert, George and Cohen', 'Food', 10.00, 757);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('MlB', 'Santiago, Chen and Chen', 'Retail', 33.61, 194);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('lDE', 'Peck PLC', 'Beverage', 20.60, 212);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('irK', 'Clark Group', 'Retail', 57.24, 658);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('WRx', 'Marshall, Gonzales and Hart', 'Food', 84.95, 490);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('pTd', 'Gould-Chandler', 'Food', 89.63, 971);
```

```
INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('HqE', 'Ferrell, Rodriguez and Pope', 'Software', 31.89, 598);

INSERT INTO Stock (StockSymbol, StockName, StockType, SharePrice,
NumAvailShares)
VALUES ('TYX', 'Weeks, Cortez and Hayden', 'Finance', 75.44, 720);

--------------------------------------------------------------------------
-------------------------------------------

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (4, 'UjT', 250, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (5, 'MlB', 100, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (6, 'WRx', 50, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (7, 'pTd', 50, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (8, 'qgZ', 50, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (9, 'HqE', 50, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (1, 'irK', 250, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (1, 'lDE', 100, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (2, 'TYX', 50, 'None', NULL);

INSERT INTO Portfolio (AccNum, StockSymbol, NumShares, Stop_, StopPrice)
VALUES (3, 'Rdx', 50, 'None', NULL);

--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------------------

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('123456789', 'Tomar', 'Prashant', '5510 Alson Drive', 'Norfolk',
'VA', '23508', '9136055136', '2006-02-02 00:00:00', 50, 'Manager');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('111111111', 'Chaudhary', 'Rohit', '5510 Alson Drive', 'Norfolk',
'VA', '93536', '9134152397', '2007-11-01 00:00:00', 60, 'CusRep');
```

```
INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('222222222', 'Martinez', 'Sarah', '3191 Bradley Parks',
'Samanthafort', 'CT', '84140', '3952451723', '2005-11-01 00:00:00', 60,
'CusRep');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('987654321', 'Gibson', 'Brenda', '92520 McGuire Ave',
'Schneidermouth', 'IA', '78506', '2377742976', '2009-11-01 00:00:00', 60,
'CusRep');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('321654987', 'Kemp', 'Patricia', '33402 Howard Viaduct',
'Kennethfurt', 'NY', '99181', '5527212043', '2010-02-02 00:00:00', 50,
'Manager');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('654321987', 'Davis', 'Laura', '671 Ward Heights', 'New
Denisechester', 'NM', '78788', '5705259385', '2015-11-01 00:00:00', 60,
'CusRep');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('852963741', 'Wheeler', 'Joe', '222 Mccall Crest', 'Mcgrathbury',
'GA', '14934', '1993218170', '2016-02-02 00:00:00', 50, 'Manager');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('963741852', 'Jennings', 'Justin', '69114 Flores Port', 'West
Sara', 'DE', '55309', '5754091899', '2008-02-02 00:00:00', 50, 'Manager');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('741852963', 'Simpson', 'Mark', '658 Cristian Tunnel', 'South
Cassandra', 'OH', '75302', '90168606283', '2015-11-01 00:00:00', 60,
'CusRep');

INSERT INTO Employee (SSN, LastName, FirstName, Address, City, State,
ZipCode, Telephone, StartDate, HourlyRate, Position_)
VALUES ('852741963', 'Harris', 'Justin', '688 Bowers Common',
'Stevenport', 'MN', '47608', '6956328125', '2022-02-02 00:00:00', 50,
'Manager');

-------------------------------------------------------------------------
-------------------------------------------------------------------------
----------------------

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('lDE', 'Sell', 25, 1, NOW(), 'Market', NULL, 1, 0);
```

```
INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('pTd', 'Sell', 40, 7, NOW(), 'Trailing Stop', 10, 1, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('HqE', 'Buy', 150, 3, NOW(), 'Market', null, 5, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('UjT', 'Buy', 150, 4, NOW(), 'Market', null, 7, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('qgZ', 'Sell', 60, 8, NOW(), 'Hidden Stop', 10, 1, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('HqE', 'Sell', 30, 9, NOW(), 'Trailing Stop', 10, 3, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('qgZ', 'Sell', 210, 1, NOW(), 'Trailing Stop', 5, 3, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('MlB', 'Sell', 35, 5, NOW(), 'Market', null, 9, 0);

INSERT INTO Order_ (StockSymbol, OrderType, NumShares, CusAccNum,
Timestamp_, PriceType, StopPrice, EmpId, Recorded)
VALUES ('WRx', 'Buy', 120, 6, NOW(), 'Market', null, 1, 0);
-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------------------
```

3. SQL DML statements to support all the transactions listed above.

```
------------------------------------------------------ MANAGER LEVEL
TRANSACTIONS ------------------------------------------------------------
----------

delimiter |
-- Sets the share price of a stock.
CREATE PROCEDURE setSharePrice(IN stock_sym VARCHAR(5), IN share_price
FLOAT(2))
      BEGIN
                  UPDATE Stock
                  SET SharePrice = share_price
                  WHERE StockSymbol = stock_sym;
    END
```

```
| delimiter ;

delimiter |
-- Adds an employee to the database.
CREATE PROCEDURE addEmployee(IN ssn CHAR(9), IN last_name VARCHAR(20),
                                                                    IN
first_name VARCHAR(20), IN address VARCHAR(50),
                                                                    IN
city VARCHAR(20), IN state VARCHAR(20),
                                                                    IN
zipcode CHAR(5), IN telephone CHAR(10),
                                                                    IN
start_date DATETIME, IN hourly_rate FLOAT(2),
                                                                    IN
pos VARCHAR(7))
      BEGIN
                  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, pos);
    END
| delimiter ;

delimiter |
-- Updates info for a given employee.
-- Only does the char attributes < does not work, can't put field in set
CREATE PROCEDURE updateEmployee(IN field CHAR(10), IN val CHAR(50), IN
emp_id INT)
      BEGIN
                  UPDATE Employee
                  SET field = val
                  WHERE EmpId = emp_id;
    END
| delimiter ;

delimiter |
-- Updates info for a given employee.
-- Only does the Timestamp_ attribute.
CREATE PROCEDURE updateEmployee2(IN val DATETIME, IN emp_id INT)
      BEGIN
                  UPDATE Employee
                  SET StartDate = val
                  WHERE EmpId = emp_id;
    END
| delimiter ;

delimiter |
-- Updates info for a given employee.
-- Only does the HourlyRate attribute.
CREATE PROCEDURE updateEmployee3(IN val FLOAT(2), IN emp_id INT)
      BEGIN
                  UPDATE Employee
                  SET HourlyRate = val
                  WHERE EmpId = emp_id;
```

```
    END
| delimiter ;

delimiter |
-- Deletes a given employee from the database.
CREATE PROCEDURE deleteEmployee(IN emp_id INT)
      BEGIN
                  DELETE FROM Employee
        WHERE EmpId = emp_id;
    END
| delimiter ;

delimiter |
-- Gets a sales report for a particular month and year.
CREATE PROCEDURE getSalesReport(IN yr INT, IN mon INT)
      BEGIN
                  SELECT T.Transfee, T.OrderId
                  FROM Transact T
                  WHERE YEAR(T.Timestamp_) = yr
                  AND MONTH(T.Timestamp_) = mon;
    END
| delimiter ;

delimiter |
-- Produces a list of all stocks and how many orders are
-- placed for each one.
CREATE PROCEDURE listAllStocks()
      BEGIN
                  SELECT S.*, COUNT(O.StockSymbol) AS OrdersPlaced
                  FROM Stock S, Order_ O
                  WHERE S.StockSymbol = O.StockSymbol
                  GROUP BY S.StockSymbol;
      END
| delimiter ;

delimiter |
-- Produces a list of all stocks and how many orders are
-- placed for each one.
CREATE PROCEDURE getNumOrders(IN stock_symbol VARCHAR(5))
      BEGIN
                  SELECT COUNT(O.StockSymbol) AS OrdersPlaced
                  FROM Order_ O
                  WHERE stock_symbol = O.StockSymbol
                  GROUP BY O.StockSymbol;
      END
| delimiter ;

delimiter |
-- Creates a list of orders that are ordered by stock symbol
-- or customer name.
CREATE PROCEDURE listOrdersBy(IN field CHAR(5))
      BEGIN
                  IF(field = 'order')
        THEN SELECT *
```

```
                            FROM Order_ O
                            ORDER BY O.StockSymbol;
                END IF;
                IF(field = 'name')
        THEN SELECT O.*, C.FirstName, C.LastName
                            FROM Order_ O, Customer C
                            WHERE O.CusAccNum = C.CusId
                            ORDER BY C.LastName, C.FirstName;
                END IF;
    END
| delimiter ;

delimiter |
-- Shows the total revenue produced by each stock.
CREATE PROCEDURE showStockRevenue()
      BEGIN
                SELECT S.StockSymbol, S.StockName, S.TotalRevenue
                FROM StockRevenue S;
    END
| delimiter ;

delimiter |
-- Shows the total revenue produced by each stock type.
CREATE PROCEDURE showStockTypeRevenue()
      BEGIN
                SELECT S.StockType, S.TotalRevenue
                FROM StockTypeRevenue S;
    END
| delimiter ;

delimiter |
-- Shows the total revenue produced by each customer.
CREATE PROCEDURE showCustomerRevenue()
      BEGIN
                SELECT C.CusAccNum, C.FirstName, C.LastName,
S.TotalRevenue
                FROM CustomerRevenue C;
    END
| delimiter ;

delimiter |
-- Shows the customer rep that generated the most
-- total revenue.
CREATE PROCEDURE showMaxEmployeeRevenue()
      BEGIN
                SELECT E.EmpId, E.TotalRevenue
                FROM EmployeeRevenue E
                WHERE E.TotalRevenue =
                            (SELECT MAX(E.TotalRevenue)
                            FROM EmployeeRevenue E);
    END
| delimiter ;

delimiter |
```

```
-- Shows the customer that generated the most total
-- revenue.
CREATE PROCEDURE showMaxCustomerRevenue()
      BEGIN
                  SELECT C.CusAccNum, C.TotalRevenue
                  FROM CustomerRevenue C
                  WHERE C.TotalRevenue =
                              (SELECT MAX(C.TotalRevenue)
                              FROM CustomerRevenue C);
    END
| delimiter ;

delimiter |
-- Shows a list of the top n most actively traded stocks.
CREATE PROCEDURE showMostTradedStocks(IN n INT)
      BEGIN
                  SELECT M.StockSymbol, M.StockName, M.NumOrders
                  FROM MostTraded M
                  ORDER BY NumOrders ASC
                  LIMIT n;
    END
| delimiter ;

-- ----------------------------------------------------------------CUSTOMER
REPRESENTATIVE LEVEL TRANSACTIONS ---------------------------------------
-----
delimiter |
-- Records an order, meaning it is ready to be executed.
CREATE PROCEDURE recordOrder(IN order_id INT)
      BEGIN
                  UPDATE Order_ O
                  SET Recorded = 1
                  WHERE O.OrderId = order_id;
    END
| delimiter ;

delimiter |
-- Adds a customer to the database.
CREATE PROCEDURE addCustomer(IN last_name VARCHAR(20), IN first_name
VARCHAR(20),
                                                                        IN
address VARCHAR(50), IN city VARCHAR(20),
                              IN state VARCHAR(20), IN zipcode CHAR(5),
                              IN telephone CHAR(10), IN email VARCHAR(50),
                                                                        IN
rating INT)
      BEGIN
                  INSERT INTO Customer(LastName, FirstName, Address, City,
State, ZipCode, Telephone, Email, Rating)
                  VALUES (last_name, first_name, address, city, state,
zipcode, telephone, email, rating);
    END
| delimiter ;
```

```
-- Update customer in db - doesn't work, see update emp for reason
delimiter |
CREATE PROCEDURE updateCustomer(IN field VARCHAR(10), IN val VARCHAR(50),
IN cus_id INT)
      BEGIN
                  UPDATE Customer C
                  SET field = val
                  WHERE CusId = cus_id;
    END
| delimiter ;

-- Delete a customer from db
delimiter |
CREATE PROCEDURE deleteCustomer(IN cus_id INT)
      BEGIN
                  DELETE FROM Customer
         WHERE CusId = cus_id;
    END
| delimiter ;
-- Create the customer mailing list
delimiter |
CREATE PROCEDURE createCustomerMailingList()
      BEGIN
                  SELECT CusId, LastName, FirstName, Email, Address, City,
State, ZipCode
                  FROM Customer;
    END
| delimiter ;


-----------------------------------------------------------------------
CUSTOMER LEVEL TRANSACTIONS ---------------------------------------------
-----------
-- Get customers current stock holdings
delimiter |
CREATE PROCEDURE getCustomerStockHoldings(IN customer_id INT)
      BEGIN
                  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;
    END
| delimiter ;

-- The share-price and trailing-stop history for a given conditional order
delimiter |
CREATE PROCEDURE getConditonalOrderTrailingStop(IN orderid INT)
      BEGIN
                  SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp_
                  FROM CONDITIONALPRICEHISTORY P
                  WHERE P.OrderId = orderid  AND P.PriceType = 'Trailing
Stop' AND P.StopPrice > 0;
```

```
    END
| delimiter ;

-- The share-price and hidden-stop history for a given conditional order
delimiter |
CREATE PROCEDURE getConditonalOrderHiddenStop(IN orderid INT)
      BEGIN
                  SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp_
                  FROM CONDITIONALPRICEHISTORY P
                  WHERE P.OrderId = orderid  AND P.PriceType = 'Hidden
Stop' AND P.StopPrice > 0;
    END
| delimiter ;

-- The share-price and hidden-stop history for a given conditional order
delimiter |
CREATE PROCEDURE getConditonalOrderHistory(IN orderid INT)
      BEGIN
                  SELECT P.CurSharePrice, P.StopPrice, P.TimeStamp_
                  FROM CONDITIONALPRICEHISTORY P
                  WHERE P.OrderId = orderid AND P.StopPrice > 0;
    END
| delimiter ;

-- The share-price history of a given stock over a certain period of time
(e.g., past six months)
delimiter |
CREATE PROCEDURE getSharePriceHistory(IN stock_symbol VARCHAR(5), IN
start_date DATE, IN end_date DATE)
      BEGIN
                  SELECT P.SharePrice, P.Timestamp_
                  FROM STOCKPRICEHISTORY P
                  WHERE P.StockSymbol = stock_symbol
                  AND DATE(P.Timestamp_) BETWEEN start_date AND end_date;
    END
| delimiter ;

-- A history of all current and past orders a customer has placed.
delimiter |
CREATE PROCEDURE getCustomerOrdersHistory(IN customer_id INT)
      BEGIN
                  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;
    END
| delimiter ;

-- Stocks available of a particular type
delimiter |
CREATE PROCEDURE getStockUsingType(IN stock_type CHAR(20))
      BEGIN
                  SELECT S.*
```

```
                FROM STOCK S
                WHERE S.StockType LIKE CONCAT('%', stock_type, '%');

    END
| delimiter ;

-- Most-recent order info of a customer
delimiter |
CREATE PROCEDURE getMostRecentOrderInfo(IN cus_id INTEGER, IN stock_sym
VARCHAR(5))
      BEGIN
                SELECT O.*
                FROM ORDER_ O, CUSTOMER C, ACCOUNT_ A
                WHERE O.CusAccNum = A.AccNum AND A.CusId = C.CusId
                            AND C.CusId = cus_id AND O.StockSymbol =
stock_sym
                            AND TIMESTAMPDIFF(MONTH, NOW(), O.Timestamp_)
<= 3;

    END
| delimiter ;

-- Stocks available with a particular keyword or set of keywords in the
stock name
delimiter |
CREATE PROCEDURE getStockUsingKeyword(IN stock_keyword CHAR(20))
      BEGIN
                SELECT S.*
                FROM STOCK S
                WHERE S.StockName LIKE CONCAT('%', stock_keyword, '%');
    END
| delimiter ;

-- Get the list of best selling stocks
delimiter |
CREATE PROCEDURE getBestSellers()
      BEGIN
                SELECT B.StockSymbol, B.TotalShares
                FROM BestSellers B
                ORDER BY TotalShares desc
                LIMIT 25;
    END
| delimiter ;

-- Get the list of best selling stocks in stock table format
delimiter |
CREATE PROCEDURE getBestSellers2()
      BEGIN
                SELECT B.StockSymbol, B.StockName, B.StockType,
B.SharePrice, B.NumAvailShares
                FROM BestSellers2 B
                ORDER BY TotalShares desc
                LIMIT 25;
    END
```

```
| delimiter ;

-- Produce a list of stock suggestions for a given customer (based on that
customer's past orders)
delimiter |
CREATE PROCEDURE Suggest(IN cus_id INT)
      BEGIN
                  SELECT DISTINCT S.StockSymbol, S.StockName, S.StockType,
S.SharePrice, S.NumAvailShares
                  FROM Stock S, Order_ O, Account_ A
                  WHERE (A.CusId = cus_id AND O.CusAccNum = A.AccNum AND
O.StockSymbol = S.StockSymbol)
        GROUP BY S.StockSymbol;
      END
| delimiter ;

delimiter |

-- Adds entry to CondPriceHist for a hidden stop when a stock price
changes.
-- Must call after updating stock price.
CREATE Procedure UpdateHiddenStop(IN new_stock_price FLOAT(2), IN
old_stock_price FLOAT(2), IN stock_symbol CHAR(5))
      BEGIN
                  IF (new_stock_price <> old_stock_price)
                              THEN INSERT IGNORE INTO
ConditionalPriceHistory(OrderId, PriceType, StopPrice, CurSharePrice,
Timestamp_)
                                    SELECT DISTINCT O.OrderId, O.PriceType,
O.StopPrice, new_stock_price, NOW()
                                    FROM Order_ O
                                    WHERE stock_symbol = O.StockSymbol
                                    AND O.PriceType IN ('Hidden Stop')
                                    AND O.Completed = 0;
                  END IF;
        IF (new_stock_price <> old_stock_price)
                              THEN SELECT DISTINCT O.OrderId,
new_stock_price, O.StopPrice
                                    FROM Order_ O
                                    WHERE stock_symbol = O.StockSymbol
                                    AND O.PriceType IN ('Hidden Stop')
                                    AND O.Completed = 0;
        END IF;
    END;
|
delimiter ;

delimiter |
-- Adds entry to CondPriceHist for a trailing stop when a stock price
changes.
-- Updates the current stop price if the current share goes up.
-- Must call after updating stock price.
CREATE PROCEDURE UpdateTrailingStop(IN new_stock_price FLOAT(2), IN
old_stock_price FLOAT(2), IN stock_symbol CHAR(5))
```

```
      BEGIN
                  IF (new_stock_price > old_stock_price)
          THEN INSERT IGNORE INTO ConditionalPriceHistory(OrderId,
PriceType, StopPrice, CurSharePrice)
                               SELECT DISTINCT O.OrderId, O.PriceType,
new_stock_price - O.StopDiff, new_stock_price
                               FROM Order_ O, ConditionalPriceHistory C
                               WHERE stock_symbol = O.StockSymbol
              AND C.Timestamp_= (SELECT DISTINCT MAX(H.Timestamp_)

FROM ConditionalPriceHistory H

WHERE O.OrderId = H.OrderId)
              AND O.PriceType = 'Trailing Stop'
              AND O.StopDiff < new_stock_price - C.StopPrice
              AND O.Completed = 0;
          END IF;
          IF (new_stock_price > old_stock_price)
                               THEN SELECT DISTINCT O.OrderId,
new_stock_price, C.StopPrice
                               FROM Order_ O, ConditionalPriceHistory C
                               WHERE stock_symbol = O.StockSymbol
              AND C.Timestamp_= (SELECT DISTINCT MAX(H.Timestamp_)

FROM ConditionalPriceHistory H

WHERE O.OrderId = H.OrderId)
              AND O.PriceType = 'Trailing Stop'
              AND O.StopDiff < new_stock_price - C.StopPrice
              AND O.Completed = 0;
          END IF;
          IF (new_stock_price < old_stock_price)
                    THEN INSERT IGNORE INTO ConditionalPriceHistory(OrderId,
PriceType, StopPrice, CurSharePrice, Timestamp_)
                               SELECT DISTINCT O.OrderId, O.PriceType,
C.StopPrice, new_stock_price, NOW()
              FROM Order_ O, ConditionalPriceHistory C
              WHERE stock_symbol = O.StockSymbol
              AND C.Timestamp_= (SELECT DISTINCT MAX(H.Timestamp_)

FROM ConditionalPriceHistory H

WHERE O.OrderId = H.OrderId)
              AND O.PriceType IN ('Trailing Stop')
              AND O.Completed = 0;
                    END IF;
          IF (new_stock_price < old_stock_price)
                               THEN SELECT DISTINCT O.OrderId,
new_stock_price, C.StopPrice
              FROM Order_ O, ConditionalPriceHistory C
              WHERE stock_symbol = O.StockSymbol
              AND C.Timestamp_= (SELECT DISTINCT MAX(H.Timestamp_)

FROM ConditionalPriceHistory H
```

```
WHERE O.OrderId = H.OrderId)
            AND O.PriceType IN ('Trailing Stop')
            AND O.Completed = 0;
        END IF;
      END;
|
delimiter ;

delimiter |

CREATE PROCEDURE MarkComplete(IN order_id INT, IN cur_share_price
FLOAT(2), IN stop_price FLOAT(2))
      BEGIN
                  IF (cur_share_price <= stop_price
                              AND 1 = (SELECT O.Recorded
                                          FROM ORDER_ O
                                          WHERE order_id = O.OrderId)
                              AND (SELECT O.NumShares
                                    FROM Order_ O
                  WHERE O.OrderId = order_id) <=
                  (SELECT P.NumShares
                   FROM Portfolio P, Order_ O
                   WHERE P.StockSymbol = O.StockSymbol
                   AND O.OrderId = order_id
                   AND O.CusAccNum = P.AccNum))
            THEN UPDATE Order_ O
                              SET O.Completed = 1
                              WHERE O.OrderId = order_id;
                  END IF;
      END;
|
delimiter ;
```