DATABASE SYSTEMS – GROUP 3 – SCRIPTS FOR QUERIES, PROCEDURES, TRIGGERS

Vishnu Muthyalu | Tien Vo | Robert Sepulveda | Zahid Laguna

***A report query that uses a JOIN (any type) to report on some aggregate value based on a group by clause.***

***SQL CODE:***

SELECT Employees\_EmployeeID AS 'employee', COUNT(Employees\_EmployeeID) AS 'number of service sales', AVG(TotalCost) AS 'average service ammount', SUM(TotalCost) AS 'total service sales'

FROM servicereceipts INNER JOIN serviceappointments

ON AppointmentID = ServiceAppointments\_AppointmentID

GROUP BY Employees\_EmployeeID

ORDER BY sum(TotalCost) DESC

***A query that includes a subquery.***

***SQL CODE:***

SELECT RecieptID, servicereceipts.ServiceDate AS "Service Date", Employees\_EmployeeID AS "Employee ID", TotalCost

FROM servicereceipts INNER JOIN serviceappointments

ON AppointmentID = ServiceAppointments\_AppointmentID

WHERE TotalCost > (

SELECT AVG(TotalCost)

FROM servicereceipts

)

ORDER BY RecieptID;

***(Please go to next Page)***

***A query that creates a view.***

***SQL CODE:***

CREATE OR REPLACE VIEW InProgress\_Services AS

SELECT AppointmentID, Employees\_EmployeeID, serviceappointments.ServiceDate, ServiceStatus, TotalCost

FROM serviceappointments JOIN servicereceipts

ON AppointmentID = ServiceAppointments\_AppointmentID

WHERE ServiceStatus = "In Progress";

***A query that demonstrates the view's use.***

***SQL CODE:***

SELECT Employees\_EmployeeID, Count(Employees\_EmployeeID), Sum(TotalCost)

FROM inprogress\_services

GROUP BY Employees\_EmployeeID

ORDER BY Sum(TotalCost) DESC;

***(Please go to next Page)***

***A stored procedure that can be called by a query to perform some math operation on the data and returns a value(s).***

***SQL CODE:***

DELIMITER $$

CREATE PROCEDURE GetAveragePrice()

BEGIN

SELECT AVG(Price) AS AveragePrice

FROM vehicleinventory;

END$$

DELIMITER ;

-- Example calling the procedure

CALL GetAveragePrice();

***A stored procedure that uses a cursor to access and manipulate (update/change) data.***

***SQL CODE: (code continues onto next page)***

DELIMITER $$

CREATE PROCEDURE UpdateVehiclePrices()

BEGIN

DECLARE finished INTEGER DEFAULT 0;

DECLARE vehicle\_id INTEGER;

DECLARE vehicle\_price DECIMAL(10,0);

-- Declare cursor

DECLARE vehicle\_cursor CURSOR FOR

SELECT VehicleID, Price FROM vehicleinventory;

-- Declare continue handler

DECLARE CONTINUE HANDLER

FOR NOT FOUND SET finished = 1;

OPEN vehicle\_cursor;

get\_prices: LOOP

FETCH vehicle\_cursor INTO vehicle\_id, vehicle\_price;

IF finished = 1 THEN

LEAVE get\_prices;

END IF;

-- Update prices

UPDATE vehicleinventory

SET Price = ROUND(Price \* 1.05, 0)

WHERE VehicleID = vehicle\_id;

END LOOP get\_prices;

CLOSE vehicle\_cursor;

END $$

DELIMITER ;

***(Please go to next Page)***

***A trigger that updates/inserts data based on an insert***

***SQL CODE:***

DELIMITER $$

CREATE TRIGGER price\_changes

AFTER UPDATE ON vehicleinventory

FOR EACH ROW

BEGIN

INSERT INTO price\_audit

(VehicleID, old\_price, new\_price, changed\_on)

VALUES

(OLD.VehicleID, OLD.Price, NEW.Price, CURRENT\_TIMESTAMP);

END$$

/\*This is for Creating the table that will track the events done by the procedure and trigger\*/

DELIMITER ;

CREATE TABLE price\_audit (

VehicleID INT,

old\_price DECIMAL(10,0),

new\_price DECIMAL(10,0),

changed\_on TIMESTAMP

);

***(Please go to next Page)***

***A query that demos the trigger.***

***SQL CODE:***

/\*These are some statements to use to test out the procedure and trigger

- uncomment each statement individually when testing them , not all at once

- the scripts below are first commented but please uncomment individually before testing them out\*/

/\*SELECT VehicleID, Price

FROM vehicleinventory

WHERE VehicleID IN (401, 402);\*/

/\*CALL UpdateVehiclePrices();\*/

/\*SELECT VehicleID, Price

FROM vehicleinventory

WHERE VehicleID IN (401, 402);\*/

/\*SELECT \*

FROM price\_audit;\*/

***The reason this is important would be in case the auto dealerships change the value of their cars, or the car market changes their prices completely and we would have to create a trigger to update it based on these factors. That’s why we provided a trigger to update the vehicle inventory and price. We also added an audit table to keep a trail of price changes and to analyze the trends over time.***