IST 659 - Data Admin Concepts & Database Management

STEPHEN O OMONDI | Data Science@ Syracuse | SUID:

946934043 Wednesday, December 19, 2018

PROJECT 2 DELIVERABLE – SQL STATEMENTS

CREATE TABLE SQL STATEMENTS

```
/*
    Author:     Stephen Omondi
    Course:     IST 659 M400
    Term:       November 2018
*/


--begin employee category table
CREATE TABLE employee_category(
    --columns for employee category table
    employee_category_id int identity,
    emp_cat_name varchar(25) NOT NULL,
    emp_cat_description varchar(55)
    --constraints on employee category table
    CONSTRAINT PK_empcat_id PRIMARY KEY(employee_category_id)
)
--end employee category table


--begin employee table
CREATE TABLE employee(
    --columns for employee table
    employee_id int identity,
    first_name varchar(25) NOT NULL,
    middle_initial char(1),
    last_name varchar(25) NOT NULL,
    email_address varchar(30) NOT NULL,
    cell_phone char(10) NOT NULL,
    home_address varchar(60),
    emp_category int NOT NULL,
    --constraints on employee table
    CONSTRAINT PK_emp_id PRIMARY KEY(employee_id),
    CONSTRAINT U1_email UNIQUE(email_address),
    CONSTRAINT FK1_emp_cat FOREIGN KEY(emp_category) REFERENCES
employee_category(employee_category_id)
)
--end employee table

--begin route table
CREATE TABLE route(
    --columns for route table
    route_id int identity,
    route_name varchar(25) NOT NULL,
    route_description varchar(55)
    --constraints on route table
    CONSTRAINT PK_routeID PRIMARY KEY(route_id)
)
--end route table


--begin fuel card table
CREATE TABLE fuel_card(
    --columns for fuel card table
    fuel_card_id int identity,
    card_number varchar(10) NOT NULL,
    card_limit char(4) NOT NULL
    --constraints on fuel card table
    CONSTRAINT PK_card_id PRIMARY KEY(fuel_card_id)
)
--end fuel card table


--begin vehicle table
CREATE TABLE vehicle(
    --columns for vehicle table
    vehicle_id int identity,
    vehicle_VIN char(17) NOT NULL,
    vehicle_make varchar(15),
    vehicle_model varchar(15),
    vehicle_mileage int NOT NULL,
    fuel_card int
    --constraints on vehile table
    CONSTRAINT PK_vid PRIMARY KEY(vehicle_id),
    CONSTRAINT U1_vin UNIQUE(vehicle_VIN),
    CONSTRAINT FK1_fuel_cardid FOREIGN KEY(fuel_card) REFERENCES fuel_card(fuel_card_id)
)
--end vehicle table


--begin driver table
CREATE TABLE driver(
    --columns for driver table
    driver_id int identity,
    employee_id int NOT NULL,
    route_id int NOT NULL,
    vehicle_id int NOT NULL,
    manager int NOT NULL
    --constraints on driver table
    CONSTRAINT PK_driveid PRIMARY KEY(driver_id),
    CONSTRAINT FK1_empid FOREIGN KEY(employee_id) REFERENCES employee(employee_id),
    CONSTRAINT FK2_routeid FOREIGN KEY(route_id) REFERENCES route(route_id),
    CONSTRAINT FK3_vehicleid FOREIGN KEY(vehicle_id) REFERENCES vehicle(vehicle_id),
    CONSTRAINT FK4_mngr FOREIGN KEY(manager) REFERENCES employee(employee_id)
)
--end driver table


--begin state table
CREATE TABLE state(
    --columns for state table
    state_id int identity,
    state_name varchar(25) NOT NULL,
    --constraints on state table
    CONSTRAINT PK_stateid PRIMARY KEY(state_id)
)
--end state table


--begin city table
CREATE TABLE city(
    --columns for city table
    city_id int identity,
    city_name varchar(25) NOT NULL,
    --constraints on city table
    CONSTRAINT PK_cityid PRIMARY KEY(city_id)
)
--end city table


--begin package table
CREATE TABLE package(
    --columns for package table
    package_id int identity,
    [weight] int NOT NULL,
    delivery_address varchar(30) NOT NULL,
    city_id int NOT NULL,
    state_id int NOT NULL,
    vehicle_id int NOT NULL,
    zip_code varchar(6) NOT NULL,
    time_dispatched datetime NOT NULL default GetDate(),
    projected_delivery datetime,
    actual_delivery datetime NOT NULL,
    remarks varchar(150),
    --constraints on package table
    CONSTRAINT PK_pkgid PRIMARY KEY(package_id),
    CONSTRAINT FK1_cityid FOREIGN KEY(city_id) REFERENCES city(city_id),
    CONSTRAINT FK2_stateid FOREIGN KEY(state_id) REFERENCES state(state_id),
    CONSTRAINT FK3_vid FOREIGN KEY(vehicle_id) REFERENCES vehicle(vehicle_id)
)
--end package table



/*
       update city table to include
       corresponding state
*/

ALTER TABLE city
ADD state_id int NOT NULL FOREIGN KEY REFERENCES state(state_id)

--end update

/*
UPDATE package actual_delivery column to accept NULL value if the package has not been
delivered yet.
This changes soon as the package is delivered.
*/

ALTER TABLE package
 ALTER COLUMN actual_delivery DATETIME NULL
```

INSERT RECORDS SQL STATEMENTS

```sql
/*
	Author: STEPHEN OMONDI
	Email: soomondi@syr.edu
	INSERT statements
	for records in each table.
	DEC 19 2018
*/


--Add records into state table--
INSERT INTO [dbo].[state](state_name)
VALUES('Alabama'), ('Alaska'), ('Arizona'), ('Arkansas'), ('California')
--end


--Add records into City table
INSERT INTO city(city_name, state_id)
VALUES('Mobile',1)
              , ('Anchorage', 2)
              , ('Phoenix', 3)
              , ('Little Rock',4)
              , ('San Francisco', 5)
--end



/*
	Insert records into employee categories.
	Employees can either be drivers, or managers
*/
INSERT INTO employee_category(emp_cat_name, emp_cat_description)
VALUES('Manager', 'Manages drivers')
              , ('Driver', 'Reports to a manager')
--end



/*
	Add records to the employee table.
	employee table must specify employee category:
	1= Manager, 2=Driver
*/
INSERT INTO employee(first_name, middle_initial, last_name, email_address, cell_phone,
home_address, emp_category)
VALUES('Godfrey', 'O', 'Obongo', 'godfrey@quantum.com', '251456690', '6173 W. Kingdom
Ave, Mobile, AL', 1)
	,('Edwin', 'D', 'Joel', 'edwinjoel@quantum.com', '251456690', '1645 S. Bewy Ave,
Mobile, AL', 2)
	,('Gladys', 'F', 'Joyce', 'gladysjoyce@quantum.com', '251456690', '1630 S. Hunter
Ave, Mobile, AL', 2)
	,('Ellen', 'D', 'Belinda', 'ellenbelinda@quantum.com', '251456690', '1630 S.
Victor Rd, Mobile, AL', 2)
	,('Diana', 'A', 'Omondi', 'dianaomondi@quantum.com', '251456690', '2356 S. Gabriel
Ave, Mobile, AL', 2)
	,('Erick', 'O', 'Omollo', 'erickomollo@quantum.com', '251456690', '8523 S. Shell
Rd, Mobile, AL', 1)
```

```
--end



/*
       Add records to the vehicle table.
       a vehicle must be assigned a unique fuel_card
       That is, one fuel card can only be assigned to one vehicle.
*/

INSERT INTO vehicle(vehicle_VIN, vehicle_make, vehicle_model, vehicle_mileage, fuel_card)
VALUES('5FNRL5H69CB130272', 'Ford', 'Transit 350', '5600', 1)
       , ('19UUA56663AR4CM67', 'Ford', 'Transit 350', '51000', 2)
       , ('JH4DB1640LS003578', 'Ford', 'Transit 350', '69000', 3)
       , ('WB1053202R6496071', 'Ford', 'Transit 350', '6370', 4)
--end



/*
       Add records to the driver table.
       a driver must be assigned a manager,
       a dedicated route and a vehicle. All
       entries are referenced keys.
*/

INSERT INTO driver(employee_id, route_id, vehicle_id, manager)
VALUES(2, 1, 1, 1), (3, 2, 2, 1), (4, 3, 3, 7), (5, 4, 4, 7)
--end




/*
       Add records to the package table.
       a package must be assigned to a city and state,
       and a vehicle for delivery. Actual Delivery and remarks
       are updated later (upon delivery). Projected delivery is 1 hour from the actual
delivery.
*/
INSERT INTO package([weight], delivery_address, city_id, state_id, zip_code, vehicle_id,
time_dispatched, projected_delivery)
VALUES(15, '6173 Shell Drive', 1, 1, '36693', 1, GETDATE(), DATEADD(hour, 1, GETDATE()) )
        , (10, '673 Shell Drive', 1, 1, '36693', 1, GETDATE(), DATEADD(hour, 1,
GETDATE()) )
        , (5, '1212 Oak Drive', 1, 1, '36695', 2, GETDATE(), DATEADD(hour, 1, GETDATE())
)
        , (7, '1256 Oak Drive', 1, 1, '36695', 2, GETDATE(), DATEADD(hour, 1, GETDATE())
)
        , (14, '1256 Foley Drive', 1, 1, '36535', 3, GETDATE(), DATEADD(hour, 1,
GETDATE()) )
        , (2, '1256 Sherman Drive', 1, 1, '36535', 3, GETDATE(), DATEADD(hour, 1,
GETDATE()) )
        , (19, '1256 Oak Drive', 1, 1, '36695', 4, GETDATE(), DATEADD(hour, 1, GETDATE())
)
        , (29, '1256 Oak Drive', 1, 1, '36695', 4, GETDATE(), DATEADD(hour, 1, GETDATE())
)
```

```
--end



--Add records for driving routes table
INSERT INTO route(route_name, route_description)
VALUES('Eastern Block', 'East side')
              , ('Western Block', 'West Side')
              , ('Southern Block', 'South Side')
              , ('Northern Block','North Side')
--end



--Add records into Fuel cards table
INSERT INTO fuel_card(card_number, card_limit)
VALUES ('A0001', 1000)
              , ('B002', 1000)
              , ('C003', 1000)
              , ('D004', 1000)
              , ('E005', 1000)
--end
```

MANIPULATE RECORDS SQL STATEMENTS

```
/*
	Author: STEPHEN OMONDI
	Email: soomondi@syr.edu
	INSERT statements
	for records in each table.
	DEC 19 2018
*/


/*
	Update the actual delivery of a package
	Get the current time when delivery was completed.

*/
UPDATE package
	SET actual_delivery = GETDATE()
WHERE package_id = 3 ---updates the first package in the list.



--Reverse the update in case the package was not delivered
UPDATE package
	SET actual_delivery = NULL
WHERE package_id = 3 ---updates the first package in the list.
```

CREATE VIEWS SQL STATEMENTS

```
/*
       Author: STEPHEN OMONDI
       Email: soomondi@syr.edu
       INSERT statements
       for records in each table.
       DEC 19 2018
*/





/*
       View of Drivers and Assigned
       Vehicles and Routes
 */

CREATE VIEW Drivers_Assigned AS
       SELECT CONCAT(first_name, ' ', last_name) AS THE_DRIVER, -- CONCAT() joins first
and last name
                 route_name AS THE_ROUTE,
                 route_description AS THE_DETAIL,
                 vehicle_VIN AS THE_VEHICLE,
                 email_address AS EMAIL,
                 cell_phone AS TEL#
       FROM route
       INNER JOIN driver ON driver.route_id = route.route_id
       INNER JOIN employee ON employee.employee_id = driver.employee_id
       INNER JOIN vehicle ON vehicle.vehicle_id = driver.vehicle_id
--end view
SELECT * FROM Drivers_Assigned




/*
       View of top 10 Employees by
       category. CONCAT() is used to join an employee first, middle
        and last names into one
 */

 CREATE VIEW TOP10_Employees AS
        SELECT TOP 10 CONCAT(first_name, ' ', middle_initial, ' ', last_name) AS
EMPLOYEE_NAME,
                     email_address AS EMAIL,
                     cell_phone AS TELL#,
                     home_address AS ADDRESS,
                     emp_cat_name AS CATEGORY
        FROM employee
        JOIN employee_category on employee_category.employee_category_id =
employee.emp_category
        ORDER BY emp_cat_name

--end
SELECT * FROM TOP10_Employees
```

```
/*
       View of packages pending delivery
       Show the driver, vehicle and package dispatch time
       and destination. Pending deliveries do not have
       actuall_delivery date

 */

CREATE VIEW Pending_Deliveries AS
       SELECT CONCAT(first_name, ' ', last_name) AS THE_DRIVER, -- CONCAT() joins first
and last name
                 route_name AS THE_ROUTE,
                 route_description AS THE_DETAIL,
                 vehicle_VIN AS THE_VEHICLE,
                 email_address AS EMAIL,
                 cell_phone AS TEL#,
                 time_dispatched AS DISPATCHED,
                 projected_delivery AS PROJECTED,
                 actual_delivery AS DELIVERED
       FROM route
                     INNER JOIN driver ON driver.route_id = route.route_id
                     INNER JOIN employee ON employee.employee_id = driver.employee_id
                     INNER JOIN vehicle ON vehicle.vehicle_id = driver.vehicle_id
                     LEFT OUTER JOIN package ON package.package_id = vehicle.vehicle_id
       WHERE actual_delivery IS NULL -- NULL value means the order has not been delivered
yet.
--END
SELECT * FROM Pending_Deliveries




/*
       View of delivered packages
       Show the driver, vehicle and package dispatch time
       and destination. Delivered packages have
       actuall_delivery date

 */

CREATE VIEW Completed_Deliveries AS
       SELECT CONCAT(first_name, ' ', last_name) AS THE_DRIVER, -- CONCAT() joins first
and last name
                 route_name AS THE_ROUTE,
                 route_description AS THE_DETAIL,
                 vehicle_VIN AS THE_VEHICLE,
                 email_address AS EMAIL,
                 cell_phone AS TEL#,
                 time_dispatched AS DISPATCHED,
                 projected_delivery AS PROJECTED,
                 actual_delivery AS DELIVERED
       FROM route
                     INNER JOIN driver ON driver.route_id = route.route_id
                     INNER JOIN employee ON employee.employee_id = driver.employee_id
                     INNER JOIN vehicle ON vehicle.vehicle_id = driver.vehicle_id
                     LEFT OUTER JOIN package ON package.package_id = vehicle.vehicle_id
       WHERE actual_delivery IS NOT NULL -- NOT NULL value means the order has been
delivered.
--END
SELECT * FROM Completed_Deliveries
```

CREATE FUNCTIONS SQL STATEMENTS

```sql
/*
FUNCTIONS:

Function to tally the number of packages
handled by a vehicle
*/

CREATE FUNCTION package_count(@vehicle_id int)
RETURNS int AS
BEGIN
	---declare variable to temporarily hold the result
	DECLARE @returnValue int
	/*
		Get the count of packages for the provided vehicle ID and
		assign that value to @returValue.
	*/

	SELECT @returnValue = COUNT(package_id) FROM package
	WHERE vehicle_id = @vehicle_id
	RETURN @returnValue

END
GO


--calling the function created above
SELECT
	*
	, dbo.package_count(vehicle_id) AS PackageCountByVehicle
FROM vehicle
ORDER BY PackageCountByVehicle DESC






/*
	Which driver has the highest number of successful packages
	delivered each week? Which driver has the least?
*/

CREATE FUNCTION DriverDelivery(@driver_id int)
RETURNS int AS
BEGIN
	---declare variable to temporarily hold the result
	DECLARE @returnValue int
	/*
		Get the count of packages for the provided driver ID and
		assign that value to @returValue.
	*/

	SELECT @returnValue = COUNT(package_id) FROM package
	INNER JOIN driver ON driver.vehicle_id = package.vehicle_id
	WHERE @driver_id = @driver_id
```

```
       RETURN @returnValue

END
GO



--calling the function created above
SELECT
       *
       , dbo.DriverDelivery(driver_id) AS PackageCountByDriver
FROM driver
ORDER BY PackageCountByDriver DESC
```

CREATE STORED PROCEDURES SQL STATEMENTS

```
/*
PROCEDURES:

Procedure to update the remarks on a package
once it is delivered.
*/


CREATE PROCEDURE UpdateActualDelivery(@remarks varchar(150), @package int)
AS
BEGIN
       /*
       Get the package which will hold the remarks to be added
       */
       UPDATE package SET remarks = @remarks
       WHERE package_id = @package
END
GO

EXEC UpdateActualDelivery 'This was delivered on time', 3

--Show the result of running the procedure
SELECT * FROM package WHERE package_id = 3
```