IST 659 - Data Admin Concepts & Database Management

STEPHEN O OMONDI | Data Science@ Syracuse | SUID: 946934043

Wednesday, December 19, 2018

Week 4: Project 1 Deliverable

PROJECT SUMMARY

Company Overview and Objective

Quantum Dispatch is a logistics company that works with companies such as Amazon and Walmart to fulfill their last mile shipping commitments. Quantum Dispatch does this by maintaining a small versatile team comprised of managers and drivers on one hand and a fleet of vans on the other hand. The company uses a file system comprised of ledgers and spreadsheets to keep track of its daily operations. However, it is currently exploring more efficient approaches to keep up with the storage and manipulation of the growing daily data collected from its operations.

Business Operations Overview

On a typical day, a driver reports to work and meets with a designated manager to plan the day's dispatches. The manager begins by dispatching a designated van along with a fuel card to the driver to deliver packages on a predetermined delivery route. The manager records the starting mileage on a mileage tracker and the Driver then loads his van with packages. All loaded packages are tracked on a driver manifest.

At the end of the day, the driver returns to the office to meet with his manager, turns in his manifest which shows whether or not each load was delivered, turns in his van to the van-pool and checks in the fuel card. The driver ensures that the Mileage tracker is updated with the latest odometer reading from the van before signing off duty for the day.

The Case for a Relational Database Model

A relational database approach has been proposed to optimize and streamline storage and retrieval of business operations data that captures the company's business model as described above. A relational database model is deemed beneficial as it can store data in tables structured for the needs of the data, supports ad hoc querying and is scalable to the business volume.

Stakeholders and Expectations

Drivers and *Managers* are the key stakeholders in the proposed project. Drivers need to see a summary of their daily deliveries to include number of packages delivered, mileage covered, and time spent while managers want to see a summary of each driver's brief above along with

vehicle utilization reports, package delivery statuses (whether they were delivered on time, delivered late, or not delivered at all). They also want a route density summary.

Data Questions to be answered from the Relational Database

- What is the total package delivered by each vehicle?
- Which driver has the highest number of successful packages delivered each week? Which driver has the least?
- Which city or zip code has the highest number of packages delivered? Which one has the least

CONCEPTUAL MODELLING

Unrefined representative data

Manager	Driver	Vehicle	Package Description	Time delivered	Route	Route Description	Miles Covered	Assigned Fuel Card
John C. Klein	Joseph P	Van A	20lb. 14 x 10 x 6	14:40	Route A	West I-65	44	001FC
Edwin D. Joel	Daniel B Brown	Van B	12lb. 14 x 10 x 7	9:30	Route B	North I-65	16	002FC
Gladys F Joyce	Paul L Joel	Van A	3lb. 14 x 10 x 8	12:10	Route C	South I-65	27	003FC
Ellen D Belinda	Lary V Jackson	Van C	90lb. 14 x 10 x 9	11:35	Route D	East I-65	17	004FC

Relationships / Business Rules

- An Employee can either be a Manager or a Driver, but not both at the same time.
- A Manager manages *one or more* drivers, but a driver *can only have one* manager *at a time*.
- A driver *can only be assigned to one* van at a time and a van *can only have one* driver assigned to it at a time.
- A driver *may be assigned one or several routes* and any route may be assigned to *one or more drivers*.

- A van *may carry none or several* packages and a package can only be in one van when dispatched
- A fuel card *can only be assigned to one van* at a time.

Cardinality

- Zero or more drivers to one and only one manager (1: M).
- One and only one driver to one and only one van (1:1).
- One or more routes to one or more drivers (M: M).
- Zero or more packages to one and only one van (1:M).
- One and only one fuel card to one and only one van (1:1)

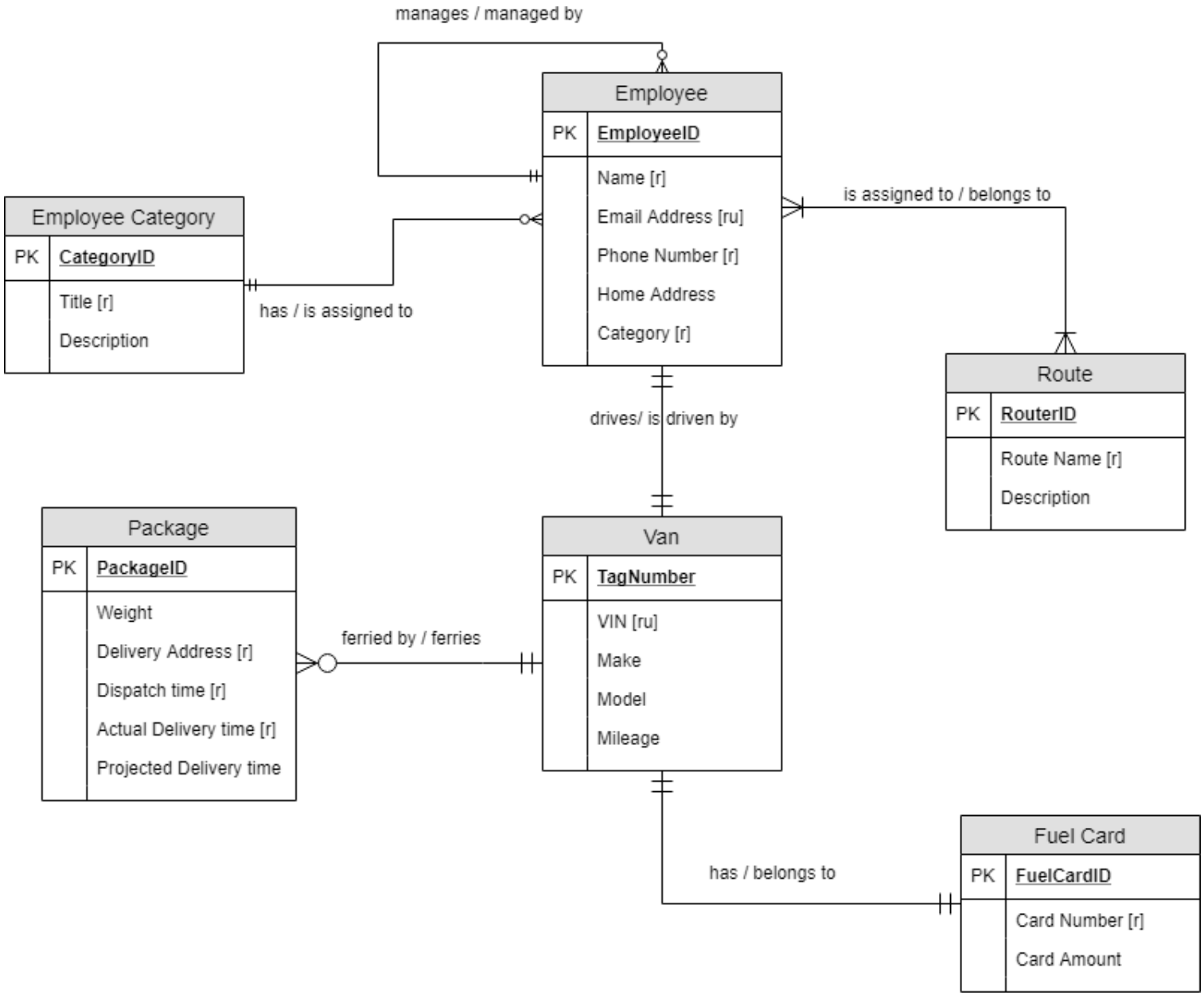
Entities, Attributes and their Descriptions

Attribute Key:

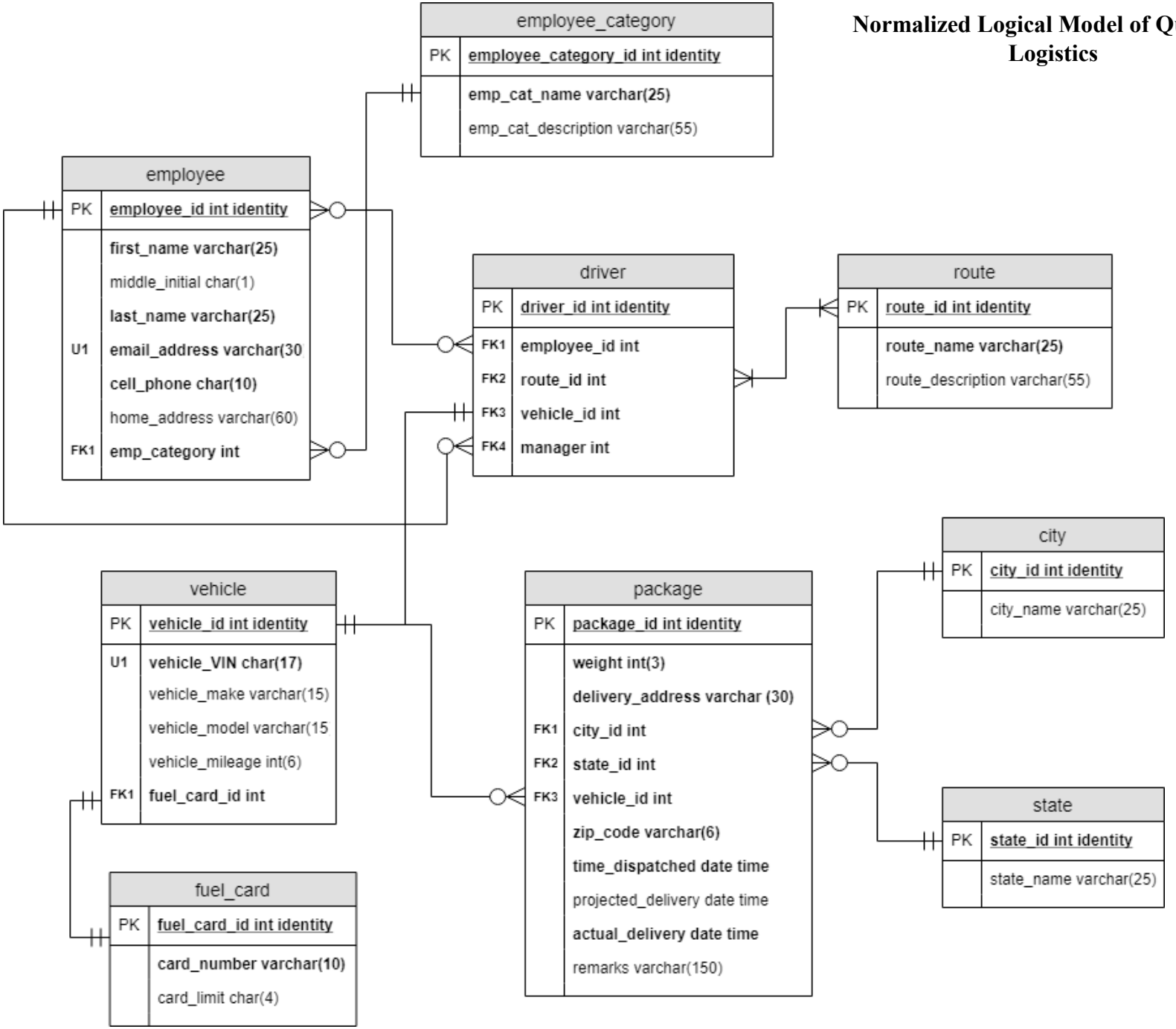
[r] = required [u] = unique [ru] = required, unique

Entity	Attributes	Description
Employees	Name [r] Email Address[ru] Phone Number[r]	Stores employee personal data.
Employee Category	Title [r] Description	Stores employee categories. An employee can either be a *Driver* or a *Manager*.
Van	VIN # Tag #[r] Mileage Make Model	Stores vehicle specifics.
Package	Weight in LBS Delivery address [r] Dispatch time [r] Projected delivery time [r] Actual delivery time [t]	Stores package information
Route	Route Name [r] Description	Stores route data.
Fuel Card	Card # [r] Amount in card	Stores fuel card data.

Conceptual Model of Quantum Logistics



Normalized Logical Model of Quantum Logistics



Glossary of tables, columns and attributes

Table: *employee_category*			
Attribute	**Data type**	**Size**	**Description**
employee_category_id	int identity	system	surrogate primary key
emp_cat_name	varchar	25	required
emp_cat_description	varchar	55	optional

Table: *employee*			
Attribute	**Data type**	**Size**	**Description**
employee_id	int identity	system	surrogate primary key
first_name	varchar	25	required
middle_initial	char	1	optional
last_name	varchar	25	required
email_address	varchar	30	unique, required
cell_phone	char	10	required
home_address	varchar	60	optional
emp_category	int	system	foreign key

Table: *driver*			
Attribute	**Data type**	**Size**	**Description**
driver_id int identity	int identity	system	surrogate primary key
employee_id	int	system	required
route_id	int	system	required foreign key
vehicle_id	int	system	required foreign key
manager	int	system	required foreign key

Table: *route*			
Attribute	**Data type**	**Size**	**Description**
route_id int identity	int identity	system	surrogate primary key
route_name	varchar	25	required
route_description	varchar	55	foreign key

Table: *vehicle*			
Attribute	**Data type**	**Size**	**Description**
vehicle_id int identity	int identity	system	surrogate primary key
vehicle_VIN	char	17	required
vehicle_make	varchar	15	optional
vehicle_model	varchar	15	optional
vehicle_mileage	int	6	required
fuel_card_id	int	system	foreign key

Table: *fuel_card*			
Attribute	**Data type**	**Size**	**Description**
fuel_card_id	int identity	system	surrogate primary key
card_number varchar	varchar	10	required
card_limit char	char	4	optional

Table: *state*			
Attribute	**Data type**	**Size**	**Description**
state_id	int identity	system	surrogate primary key
state_name	varchar	25	required

Table: *city*			
Attribute	**Data type**	**Size**	**Description**
city_id	int identity	system	surrogate primary key
city_name	varchar	25	required

Table: *package*			
Attribute	**Data type**	**Size**	**Description**
package_id	int identity	system	surrogate primary key
weight	3	int	required
delivery_address	varchar	30	required
city_id	int	system	foreign key
state_id	int	system	foreign key
vehicle_id	int	system	foreign key
zip_code	varchar	6	required
time_dispatched	date time	system	required
projected_delivery	date time	system	optional
actual_delivery	date time	system	required
remarks	varchar	150	optional

PHYSICAL DATABASE DESIGN

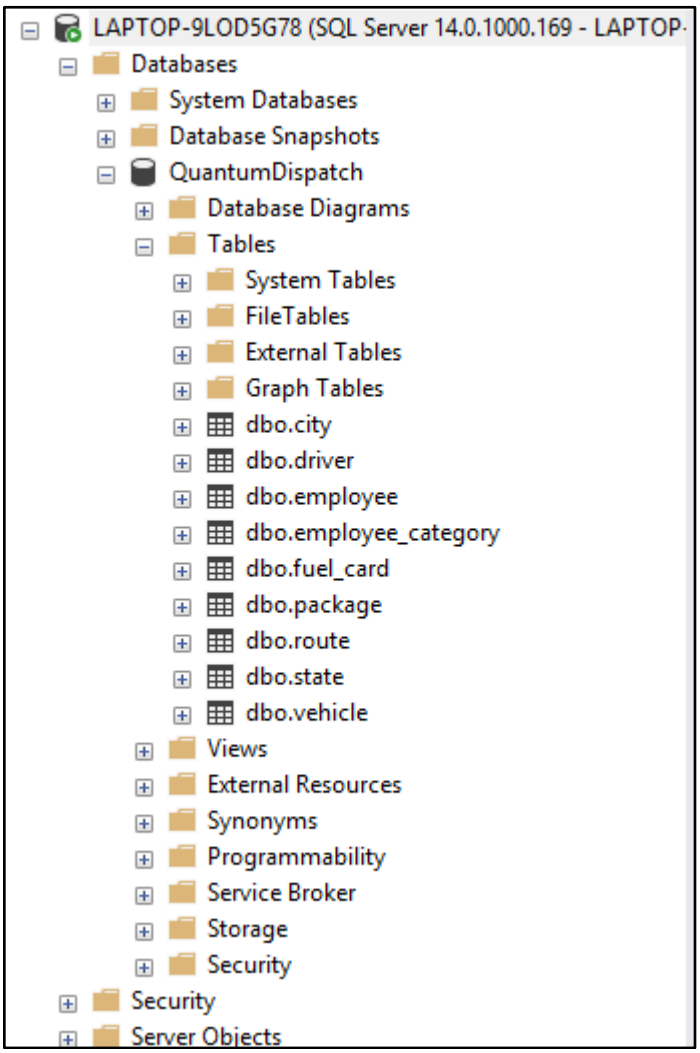
DDL – Create Tables

```
/*
    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
```

```sql
--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
```

```sql
--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
       correspoinding state
*/

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

--end update
```

Figure showing all tables created from DDL above.



DATA CREATION

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

STATE Table

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

SELECT * FROM state
```

Results | Messages

	state_id	state_name
1	1	Alabama
2	2	Alaska
3	3	Arizona
4	4	Arkansas
5	5	California

CITY Table

```
--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

SELECT * FROM city
```

100 %

Results | Messages

	city_id	city_name	state_id
1	1	Mobile	1
2	2	Anchorage	2
3	3	Phoenix	3
4	4	Little Rock	4
5	5	San Francisco	5

EMPLOYEE CATEGORY Table

```
/*
	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

SELECT * FROM employee_category
```

100 %

Results | Messages

	employee_category_id	emp_cat_name	emp_cat_description
1	1	Manager	Manages drivers
2	2	Driver	Reports to a manager

EMPLOYEE Table

```
/*
	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
```

```sql
SELECT * FROM employee
```

100 %

Results | Messages

	employee_id	first_name	middle_initial	last_name	email_address	cell_phone	home_address	emp_category
1	1	John	C	Klein	johnklein@quantum.com	251456690	1630 S. Niazuma Ave, Mobile, AL	1
2	2	Godfrey	O	Obongo	godfrey@quantum.com	251456690	6173 W. Kingdom Ave, Mobile, AL	1
3	3	Edwin	D	Joel	edwinjoel@quantum.com	251456690	1645 S. Bewy Ave, Mobile, AL	2
4	4	Gladys	F	Joyce	gladysjoyce@quantum.com	251456690	1630 S. Hunter Ave, Mobile, AL	2
5	5	Ellen	D	Belinda	ellenbelinda@quantum.com	251456690	1630 S. Victor Rd, Mobile, AL	2
6	6	Diana	A	Omondi	dianaomondi@quantum.com	251456690	2356 S. Gabriel Ave, Mobile, AL	1
7	7	Erick	O	Omollo	erickomollo@quantum.com	251456690	8523 S. Shell Rd, Mobile, AL	1

VEHICLE Table

```sql
/*
	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

SELECT * FROM vehicle
```

100 %

Results | Messages

	vehicle_id	vehicle_VIN	vehicle_make	vehicle_model	vehicle_mileage	fuel_card
1	1	5FNRL5H69CB130272	Ford	Transit 350	5600	1
2	2	19UUA56663AR4CM67	Ford	Transit 350	51000	2
3	3	JH4DB1640LS003578	Ford	Transit 350	69000	3
4	4	WB1053202R6496071	Ford	Transit 350	6370	4

DRIVER Table

```
/*
        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

SELECT * FROM driver
```

100 %

Results | Messages

	driver_id	employee_id	route_id	vehicle_id	manager
1	2	2	1	1	1
2	3	3	2	2	1
3	4	4	3	3	7
4	5	5	4	4	7

PACKAGE Table

```
/*
        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
```

```sql
select * from package
```

Results | Messages

	package_id	weight	delivery_address	city_id	state_id	vehicle_id	zip_code	time_dispatched	projected_delivery	actual_delivery	remarks
1	3	15	6173 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
2	4	10	673 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
3	5	5	1212 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
4	6	7	1256 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
5	7	14	1256 Foley Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
6	8	2	1256 Sherman Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
7	9	19	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
8	10	29	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL

ROUTE Table

```sql
--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

SELECT * FROM route
```

100 %

Results | Messages

	route_id	route_name	route_description
1	1	Eastern Block	East side
2	2	Western Block	West Side
3	3	Southern Block	South Side
4	4	Northern Block	North Side

FUEL_CARD Table

```sql
--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
SELECT * FROM fuel_card
```

100 %

Results | Messages

	fuel_card_id	card_number	card_limit
1	1	A0001	1000
2	2	B002	1000
3	3	C003	1000
4	4	D004	1000
5	5	E005	1000

DATA MANIPULATION

Packages are updated once delivered. A system generated date stamp is inserted into the records at the time it is marked as delivered. This updates the *actual_delivery* column of the *package* table as shown below. By default, *actual_delivery* column accepts a null value when the package has not been delivered. The manager can then evaluate efficiency by looking at the difference between the actual delivery and the projected delivery. A package is considered successful when it is delivered within one hour from dispatch.

```
/*
        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.
```

100 %

Results | Messages

	package_id	weight	delivery_address	city_id	state_id	vehicle_id	zip_code	time_dispatched	projected_delivery	actual_delivery	remarks
1	3	15	6173 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	2018-12-19 12:53:19.040	NULL
2	4	10	673 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
3	5	5	1212 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
4	6	7	1256 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
5	7	14	1256 Foley Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
6	8	2	1256 Sherman Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
7	9	19	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
8	10	29	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL

Query executed successfully. | ist-s-students.syr.edu (12... | AD\soomondi (75) | IST659_M404_soomondi | 00:00:00 | 8 rows

If the delivery did not happen and this record needs to be changed, then it can be reversed back to NULL.

```
--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.
```

100 %

Results | Messages

	package_id	weight	delivery_address	city_id	state_id	vehicle_id	zip_code	time_dispatched	projected_delivery	actual_delivery	remarks
1	3	15	6173 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
2	4	10	673 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
3	5	5	1212 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
4	6	7	1256 Oak Drive	1	1	2	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
5	7	14	1256 Foley Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
6	8	2	1256 Sherman Drive	1	1	3	36535	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
7	9	19	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL
8	10	29	1256 Oak Drive	1	1	4	36695	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL	NULL

DDL – CREATE VIEWS

View of Drivers and Assigned Vehicles and Routes

```
/*
       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
SELECT * FROM Drivers_Assigned
--end view
```

100 %

Results | Messages

	THE_DRIVER	THE_ROUTE	THE_DETAIL	THE_VEHICLE	EMAIL	TEL#
1	Godfrey Obongo	Eastern Block	East side	5FNRL5H69[illegible]B130272	godfrey@quantum.com	251456690
2	Edwin Joel	Western Block	West Side	19UUA56663AR4CM67	edwinjoel@quantum.com	251456690
3	Gladys Joyce	Southern Block	South Side	JH4DB1640LS003578	gladysjoyce@quantum.com	251456690
4	Ellen Belinda	Northern Block	North Side	WB1053202R6496071	ellenbelinda@quantum.com	251456690

View of Top 10 Employees by Category

```
/*
       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
```

```
SELECT * FROM TOP10_Employees
--end
```

Results | Messages

	EMPLOYEE_NAME	EMAIL	TELL#	ADDRESS	CATEGORY
1	Edwin D Joel	edwinjoel@quantum.com	251456690	1645 S. Bewy Ave, Mobile, AL	Driver
2	Gladys F Joyce	gladysjoyce@quantum.com	251456690	1630 S. Hunter Ave, Mobile, AL	Driver
3	Ellen D Belinda	ellenbelinda@quantum.com	251456690	1630 S. Victor Rd, Mobile, AL	Driver
4	John C Klein	johnklein@quantum.com	251456690	1630 S. Niazuma Ave, Mobile, AL	Manager
5	Godfrey O Obongo	godfrey@quantum.com	251456690	6173 W. Kingdom Ave, Mobile, AL	Manager
6	Diana A Omondi	dianaomondi@quantum.com	251456690	2356 S. Gabriel Ave, Mobile, AL	Manager
7	Erick O Omollo	erickomollo@quantum.com	251456690	8523 S. Shell Rd, Mobile, AL	Manager

View of Pending Packages

```
/*
       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
```

100 % | Results | Messages

	THE_DRIVER	THE_ROUTE	THE_DETAIL	THE_VEHICLE	EMAIL	TEL#	DISPATCHED	PROJECTED	DELIVERED
1	Godfrey Obongo	Eastern Block	East side	5FNRL5H69CB130272	godfrey@quantum.com	251456690	NULL	NULL	NULL
2	Edwin Joel	Western Block	West Side	19UUA56663AR4CM67	edwinjoel@quantum.com	251456690	NULL	NULL	NULL
3	Gladys Joyce	Southern Block	South Side	JH4DB1640LS003578	gladysjoyce@quantum.com	251456690	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL
4	Ellen Belinda	Northern Block	North Side	WB1053202R6496071	ellenbelinda@quantum.com	251456690	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	NULL

Click to select the whole column

View of Delivered Packages

```
/*
       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
```

100 %

Results | Messages

	THE_DRIVER	THE_ROUTE	THE_DETAIL	THE_VEHICLE	EMAIL	TEL#	DISPATCHED	PROJECTED	DELIVERED
1	Gladys Joyce	Southern Block	South Side	JH4DB1640LS003578	gladysjoyce@quantum.com	251456690	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	2018-12-19 15:01:49.653

DDL – CREATE FUNCTIONS

```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
```

100 %

Results | Messages

	vehicle_id	vehicle_VIN	vehicle_make	vehicle_model	vehicle_mileage	fuel_card	PackageCountByVehicle
1	1	5FNRL5H69CB130272	Ford	Transit 350	5600	1	2
2	2	19UUA56663AR4CM67	Ford	Transit 350	51000	2	2
3	3	JH4DB1640LS003578	Ford	Transit 350	69000	3	2
4	4	WB1053202R6496071	Ford	Transit 350	6370	4	2

DDL – CREATE STORED PROCEDURES

```
/*
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
```

100 %

Results | Messages

	package_id	weight	delivery_address	city_id	state_id	vehicle_id	zip_code	time_dispatched	projected_delivery	actual_delivery	remarks
1	3	15	6173 Shell Drive	1	1	1	36693	2018-12-19 12:35:01.143	2018-12-19 13:35:01.143	2018-12-19 15:01:49.653	This was delivered on time

ANSWERING DATA QUESTIONS

What is the total package delivered by each vehicle?

```
/*
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
```

100 %

Results | Messages

	vehicle_id	vehicle_VIN	vehicle_make	vehicle_model	vehicle_mileage	fuel_card	PackageCountByVehicle
1	1	5FNRL5H69CB130272	Ford	Transit 350	5600	1	2
2	2	19UUA56663AR4CM67	Ford	Transit 350	51000	2	2
3	3	JH4DB1640LS003578	Ford	Transit 350	69000	3	2
4	4	WB1053202R6496071	Ford	Transit 350	6370	4	2

All vehicles have two deliveries so far.

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

```
/*
	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
```

100 %

Results | Messages

	driver_id	employee_id	route_id	vehicle_id	manager	PackageCountByDriver
1	2	2	1	1	1	8
2	3	3	2	2	1	8
3	4	4	3	3	7	8
4	5	5	4	4	7	8

Therefore, it appears all drivers have the same deliveries currently.

Which city or zip code has the highest number of packages delivered? Which one has the least?

```
SELECT package.city_id, package.zip_code, city_name FROM PACKAGE
INNER JOIN city ON city.city_id = package.city_id
ORDER BY package.zip_code ASC
```

100 %

Results | Messages

	city_id	zip_code	city_name
1	1	36535	Mobile
2	1	36535	Mobile
3	1	36693	Mobile
4	1	36693	Mobile
5	1	36695	Mobile
6	1	36695	Mobile
7	1	36695	Mobile
8	1	36695	Mobile

MOBILE city has and zip code 36695 have the highest number of packages delivered

IMPLEMENTATION AND REFLECTION

I have learned a lot about databases than I ever before. I particularly loved the Stored Procedures, Views and Functions as they were the more novel concepts to me. I also loved working with SQL Server Studio Manager – being a first for me. The introductory parts dealing with conceptual and logical designs were very important in the larger scope of things but this only became apparent as I steeped deeper into the development work.

However, I started out with the hope of implementing a web based interface but this did not pass because of time constraints. I have implemented a basic Microsoft Access Forms and Report view but I know I could have done better – given more time.

In the end, I leave this class a lot more confident in handling SQL Server and working with Microsoft Azure. I now embark on dedicated practice to horn these skills further. The one specific area that I need revision on are on creating users and related permissions.