Scalable Database Systems - Report

For this assessment, I started by planning what information is required to achieve the outcomes. First, I planned a table with all the required information inside. By creating the database first in 1NF, it was easier to gather all fields I would need for the database. This was the easiest way to begin the task, then after I split the information up into different tables to achieve 3NF and make sure data is not being repeated. The Database is complete and consists of 5 tables used to hold information on the task based for the Queries and Procedures. Drivers were given a unique 6 digit ID number and Vehicles were given a 8 digit ID number to identify them. Drivers addresses and delivery addresses were given in the form, [NUMBER] [STREET NAME], [POSTCODE]. Alternatively, Vehicle location was given in Longitude and Latitude coordinates.

Design Process

I created a Database called ‘gps’ which would hold the tables for manipulating data. Then I created 5 main tables within the database, called ‘Drivers’, ‘Vehicles’, ‘Shift’, ‘Parcels’ and ‘VehicleLocation’. The Drivers table is there to hold information about every individual driver who works for the delivery company. It holds their unique ID, their name and personal address. The Vehicles table holds information on each vehicle the business owns. It's needed to assign vehicles to drivers and keep track of vehicle information. The Vehicles table consists of a VehicleID which is the unique identifier for this table, (primary key). Also in the Vehicles table is VehicleName which is there to identify the vehicle in a human friendly way. The ID is unique but hard for a human to know exactly which vehicle it is, so giving it a name helps. The Shift table is like a link of the drivers table and vehicles. It contains the Drivers ID, The Vehicle they are using for that shift and the time and date of their shift. The table Parcels is there to hold all the parcels information. This includes the unique parcel ID, the driver who is delivering it, the date of delivery and the address. It also contains DeliveredParcel which is type BIT, if its set to 0 it means the parcel has not been delivered yet, and if it's set to 1, it means it has been delivered to the customer. The VehicleLocation table holds everything needed to track a driver and vehicle's location. It has a Vehicle ID to track which vehicle’s location we are looking for, the driver who is in the vehicle, the date and time, as well as the vehicle's location, stored in longitude and latitude.

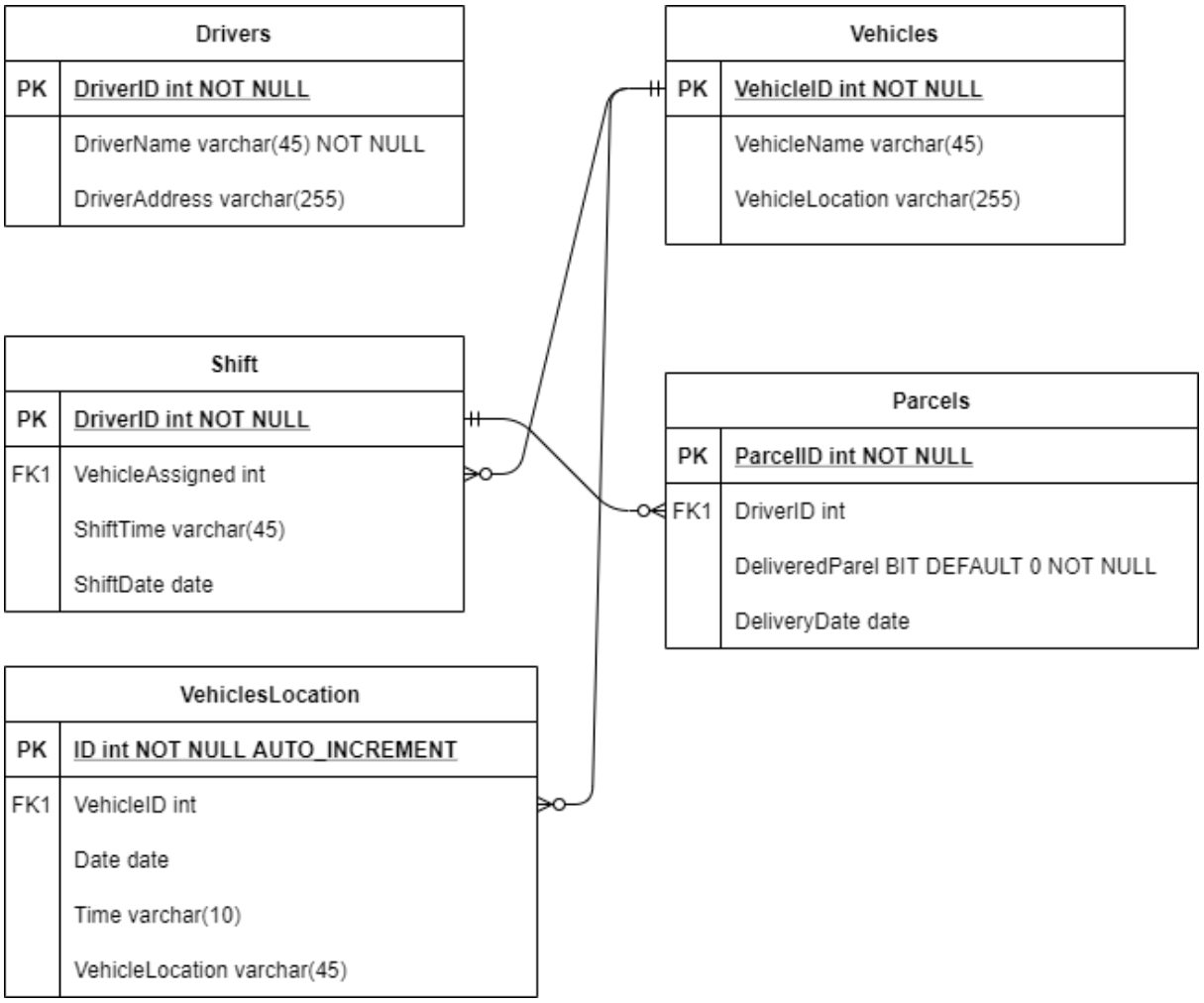
ID’s for Drivers and Vehicles:

Number	DriverID	VehicleID
1	192843	297391189
2	183933	774860030
3	178593	137979860

19701707
Douglas Carrie

4	145940	080341348
5	196532	600880903
6	157294	
7	123934	
8	164739	
9	113859	
10	103859	

Entity Relation Diagram:



Stored procedures:

Search how many parcels have been delivered by a certain driver based on their unique ID:

```
DELIMITER //
CREATE PROCEDURE CheckDriverProgress( IN ID int)

BEGIN
SELECT DriverID, DriverName,ParcelID, DeliveredParcel
FROM gps.Drivers
INNER JOIN gps.Parcels
ON Drivers.DriverID = Parcels.Driver AND Parcels.DeliveredParcel = 1 AND
Drivers.DriverID = ID;
END //

DELIMITER ;

CALL CheckDriverProgress;
```

The procedure could have been created using the driver's name as the input, however employees names are not guaranteed to be unique and there could be user error when entering the employee's name, ie. not including spaces or incorrect spelling.

However, the procedure works as intended. An example:

If we use Joel Gibson and run the Stored Procedure using his DriverID:

Driver:

DriverID	DriverName
192843	Joel Gibson

```
CALL CheckDriverProgress(192843);
```

Parcels:

ParcelID	DriverID	DeliveredParcel
192843	192843	1

Result:

DriverID	DriverName	ParcelID	DeliveredParcel
192843	Joel Gibson	1928435	1

56		
19284357	192843	1
19284358	192843	0

		6	
192843	Joel Gibson	19284357	1

The result table above displays all parcels that have been delivered based on that specific driver. DeliveredParcel is automatically set to 0 and then set to 1 when a parcel has been delivered. The results table shows that two out of three parcels have been delivered. The procedure works as it should and will display all parcels that have been delivered by that driver.

Check which Drivers are working Morning shifts:

```
DELIMITER //

CREATE PROCEDURE CheckMorningShift()

BEGIN
SELECT Drivers.DriverID, DriverName, ShiftTime
FROM gps.Drivers
INNER JOIN gps.Shift
ON Drivers.DriverID = Shift.DriverID AND Shift.ShiftTime = '8:00-11:55';
END //

DELIMITER ;

CALL CheckMorningShift;
```

This procedure checks if ShiftTime is equal to ‘8:00-11:55’, however it could also be changed to ‘12:00-16:00’ which would represent the afternoon shift.
On the other hand, it could easily be adapted to accept an argument, so that the user could then check both shift times. The user would just need to provide an input being the time shift they are searching for.

Output:

DriverID	DriverName	ShiftTime
183933	Emily Yates	8:00-11:55
192843	Joel Gibson	8:00-11:55

Queries:

Show all drivers:

```
SELECT DriverID, DriverName
FROM gps.Drivers
```

This simple query above just gets every entry in the Drivers table and displays their ID and Name.

Output:

DriverID	DriverName
103859	Kaylem Perkins
113859	Iram Gibbons
123934	Alfie-Jay Guthrie
145940	Kayley Bates
157294	Chris Andersen
164739	Mason Ayers
178593	Carolyn Irvine
183933	Emily Yates
192843	Joel Gibson
196532	Rubie Munoz

Check Location of specific driver on specific day:

```
SELECT VehiclesLocation.VehicleID, Drivers.DriverName, VehicleName,
Time, VehicleLocation

FROM gps.VehiclesLocation
LEFT JOIN gps.Vehicles
ON VehiclesLocation.VehicleID = Vehicles.VehicleID
LEFT JOIN gps.Drivers
ON VehiclesLocation.DriverID = Drivers.DriverID
WHERE VehiclesLocation.DriverID = 192843 AND VehiclesLocation.Date =
'2021-01-02';
```

For this query, The DriverID 192843 is used, (Joel Gibson). And the date chosen is 2021-01-02.

Output:

DriverName	VehicleName	Time	VehicleLocation
Joel Gibson	Mercedes-benz Sprinter 314	8:00	53.16935332,-0.45953462
Joel Gibson	Mercedes-benz Sprinter 314	9:00	53.19078782,-0.46382307
Joel Gibson	Mercedes-benz Sprinter 314	10:00	53.28302423,-0.52976645
Joel Gibson	Mercedes-benz Sprinter 314	11:00	53.18805913,-0.67537285
Joel Gibson	Mercedes-benz Sprinter 314	12:00	53.25595858,-0.48887112

If the date was changed to the next day, (2021-01-03). There would be 0 results as Joel Gibson does not have a shift for that day. The DriverID could also be changed here to any other driver to check their location.

This query would be much better suited as a Stored Procedure. Converting it into a procedure would allow the user to choose which Driver and Date to run the query from. I have created this as a stored procedure below:

```
DELIMITER //

CREATE PROCEDURE CheckLocation( IN ID int, IN Date varchar(10))

BEGIN
SELECT VehiclesLocation.VehicleID, Drivers.DriverName, VehicleName,
Time, VehicleLocation
FROM gps.VehiclesLocation
LEFT JOIN gps.Vehicles
ON VehiclesLocation.VehicleID = Vehicles.VehicleID

LEFT JOIN gps.Drivers
ON VehiclesLocation.DriverID = Drivers.DriverID
WHERE VehiclesLocation.DriverID = ID AND VehiclesLocation.Date = date;

END //

DELIMITER ;
```

If we call this procedure and supply it with some valid arguments, it will return results as expected:

```
CALL CheckLocation(113859,'2021-02-02')
```

Above will return 0 results as Driver 113859 (Iram Gibbons) does not have a shift on the 2021-02-03
If we change the date to one where Iram does have a shift like 2021-02-03, we will get results on location, shown below:

```
CALL CheckLocation(113859,'2021-02-03')
```

Result:

VehicleID	DriverName	VehicleName	Time	VehicleLocation
774860030	Iram Gibbons	Mercedes-benz Vito 111	8:00	53.27727126,-0.50767344
774860030	Iram Gibbons	Mercedes-benz Vito 111	9:00	53.22333082,-0.49068089
774860030	Iram Gibbons	Mercedes-benz Vito 111	10:00	52.12851572,-0.61980931

774860030	Iram Gibbons	Mercedes-benz Vito 111	11:00	53.57137048,-0.70034063
774860030	Iram Gibbons	Mercedes-benz Vito 111	12:00	53.14552797,-0.61666297

Further considerations:

This Scalable Database project could now be integrated into a real scenario in real life with a few changes. At the moment, the application feels a bit raw, and would benefit from a proper User interface. Using other languages in conjunction, this database could be converted into a full web application or Desktop App. However, it would need to be made more secure with a login so that employees' private information such as names, locations and personal addresses are not accessible to everybody. Encryption would also be a good measure to implement, so that sensitive information like passwords are not obtained by the wrong people.