

**School of Computer Science
University of Lincoln**

CMP2806M Scalable Database Systems

Report

Delivery Tracking Database Design



**UNIVERSITY OF
LINCOLN**

Oakleigh Weekes

**Submitted as part fulfilment of the requirement for
BSc/MComp Computer Science**

January 2021

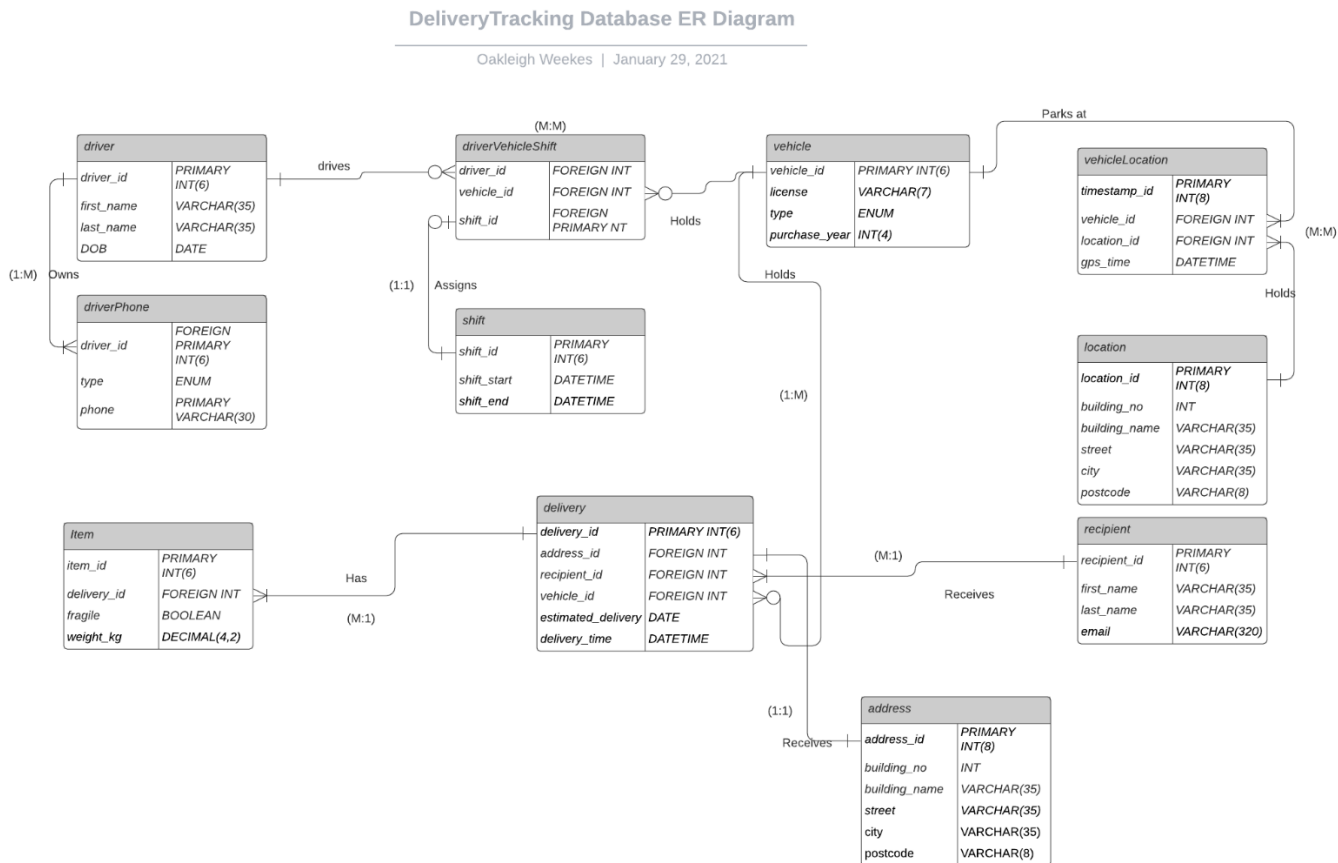
Contents

Introduction.....	3
Entity-Relationship Diagram.....	3
Design Overview.....	3
Assumptions.....	3
Design Process	4
Normalisation	5
Queries.....	5
Location of any vehicle at any time.....	5
Number of parcels delivered by any specific driver during a day's work.....	6
A listing of all drivers.....	6
Drivers who have only worked morning shifts.....	7
Procedures.....	8
Locate_vehicle.....	8
Items_delivered	9
Security & Scalability	9
Conclusion	10
References.....	11
Appendix	12

Introduction

The following is a descriptive report on the design and implementation of a relational database used for parcel tracking within a theoretical parcel delivery company.

Entity-Relationship Diagram



Design Overview

Assumptions

To design the database, several assumptions were concluded about the delivery company and its systems.

The Company works within the Nottingham/Lincolnshire area. They consist of ten fictitious drivers and five vehicles. The vehicles are labelled with a vehicle ID and have a license associated with them in the records for easier identification. They are stored in the 'vehicle' table. The vehicles can include vans, cars or 'other' (which are non-company vehicular assets for ad-hoc deliveries). Driver information is stored in the 'driver' table. Drivers are associated with phone number(s) for contact purposes which are stored in the 'driverPhone' table. These are labelled 'mobile' or 'home' numbers.

Each driver is assigned to a shift sometime before each working day according to their contract. They are also assigned a different vehicle each time they work. Assignment can be

CMP2806M Scalable Database Systems Assessment

seen within the 'driverVehicleShift' table. The current shifts available are from 08:00am to 11:55am ('Morning'), or 12:00pm to 16:00pm ('Afternoon'), and are stored in the 'shift' table. However, the database is configured in such a way that other shift patterns could be possible in the future. The company may introduce a 'clocking in' system to measure the exact time worked versus the associated shift. For query sake, possible morning shifts could start from 06:00am-09:00am and end from 07:00am-12:00pm, whilst afternoon shifts could start from 12:00pm-14:00pm and end from 15:00pm-17:00pm.

The GPS system attached to the vehicles sends its current location to the 'location' table in the database every hour. The system converts X/Y coordinates into a standard address format before sending. A location id is associated with each location sent. Each hour is logged in a 'vehicleLocation' table which links the vehicle to its location, and an associated timestamp ID is recorded for each entry. Drivers signal when a delivery was successful using a device, and the time of delivery is sent to the associated 'delivery' table. Undelivered deliveries would be marked as 'NULL' here. This table also includes an estimated delivery date (calculated on delivery confirmation by the company), the vehicle associated with the delivery, the recipient ID and Address ID.

Individual Item information for a delivery (linked by 'delivery_id') is stored in another 'item' table. The table includes if the item is fragile and how much it weighs (In kg, rounded to 2 decimal points). The company does not deliver parcels over 20kg.

Recipient information is stored in the 'recipient' table. This includes their full name and email if they have signed up the company's tracking application. The address associated with each delivery is stored in the 'address' table.

Design Process

The design process followed was in accordance with Microsoft's (2021) Database design basics guide.

The database purpose was established as per the assessment brief. A database scheme needed to be created to store relevant vehicle GPS, driver, shift, vehicle, and parcel data.

On this occasion the data to be stored was fictional and so was randomly created to be somewhat realistic. In a real scenario information would be extracted and organised from paper records, a real GPS, recipient information from an application, or a legacy database or spreadsheet.

Information collected was sorted into different tables with the help of the Microsoft Excel software (2018). An Entity-Relationship diagram was drafted with 'crows-foot' notation. Each table is associated with an entity. Attributes for an entity were organised into separate columns with an associated data type. [Please see Appendix A for the Excel tables].

Primary keys were identified to uniquely identify each record in the tables. 'driverPhone' included a composite primary key of 'phone' and 'driver_id' such that different workers could have the same phone number, but each individual driver's phone numbers must be unique.

CMP2806M Scalable Database Systems Assessment

Table relationships were established. For instance, 'delivery' has a one-to-many relationship with 'item'. 'address' has a one-to-one relationship to 'delivery'. Foreign keys link respective related tables together. In the case of many-to-many relationships, a link table is established to unify the entities. For example, 'vehicleLocation' links many vehicles with many locations, and 'driverVehicleShift' links many drivers to many vehicles with a shift each.

Tables were finally refined by reducing redundancy using normalisation. This database is normalised to third normal form.

Normalisation

To achieve '3NF', the first and second normal forms need to be satisfied.

First normal form had been met as each table includes a primary key. There is no single column with multiple values and no duplicate values exist among tables.

Second normal form had been met as "each attribute does in fact describe the entity" within a table (Microsoft 2012 p63). Non-candidate key attributes are functionally dependant upon the primary key. No partial dependencies exist. Phone numbers of the drivers were moved to a separate 'phone' table to avoid this, as two drivers could share the same phone number or have multiple. Foreign keys are created. Recipient email could be a partial dependency with 'recipient id', however, to create an account with the company each account must have a unique email. 'address' has its own table as addresses are not linked to accounts, they can be constantly changed, and multiple recipients can receive deliveries to the same address. In this sense the 'address' is more so related to the delivery than the recipient.

Third normal form is reached on removal of transitive dependencies. 3NF is violated when "the attribute depends on the key but also on another non-key attribute" (Microsoft 2012 p64). Derived attributes were removed, such as "items_delivered" in 'delivery' and "years_owned" in the 'vehicle' table. The database had then been normalized to 3NF.

Queries

Location of any vehicle at any time

This nested query outputs a record of the location table based on the vehicle id and time of day.

Inputs: vehicle_id = 2 // gps_time = 2020-02-01 10:00:00

```
SELECT * FROM location WHERE location_id = (SELECT location_id FROM
vehiclelocation where vehicle_id = 2 AND gps_time = '2020-02-01 10:00:00');
```

Output:

	location_id	building_no	building_name	street	city	postcode
	3	5	Oak apts	Tree In	Lincoln	LN7 89U

CMP2806M Scalable Database Systems Assessment

Number of parcels delivered by any specific driver during a day's work.

The specific day's work is associated with the specific driver, and so the input is the unique 'shift_id' rather than the 'driver_id'. The query counts the number of parcels delivered within the timeframe of the 'shift_id' by accessing a joined delivery and item table. MySQL variables are used for storage of nested queries.

Inputs: shift_id = 2

```
SET @shiftStart = (SELECT shift_start FROM shift WHERE shift_id = 2);
```

```
SET @shiftEnd = (SELECT shift_end FROM shift WHERE shift_id = 2);
```

```
SET @driverVehicle = (SELECT vehicle_id FROM drivervehicleshift WHERE shift_id = 2);
```

```
SELECT COUNT(*) 'Items Delivered'
```

```
FROM item i
```

```
INNER JOIN delivery d
```

```
USING(delivery_id)
```

```
WHERE d.vehicle_id = @driverVehicle AND (d.delivery_time BETWEEN @shiftStart AND @shiftEnd);
```

Output:

Items Delivered

6

A listing of all drivers


This query simply outputs the contents of the 'driver' table and orders by drivers' last name.

```
SELECT * FROM driver
```

```
ORDER BY last_name;
```

Output: [See next page]

CMP2806M Scalable Database Systems Assessment

	driver_id	first_name	last_name	1	DOB
<input type="checkbox"/>  Edit  Copy  Delete	7	Kathy	Burke		1946-06-13
<input type="checkbox"/>  Edit  Copy  Delete	4	Rupaul	Charles		1960-11-17
<input type="checkbox"/>  Edit  Copy  Delete	3	Bianca	Del Rio		1975-06-27
<input type="checkbox"/>  Edit  Copy  Delete	2	Veronica	Green		1985-04-24
<input type="checkbox"/>  Edit  Copy  Delete	1	Ginni	Lemon		1989-02-07
<input type="checkbox"/>  Edit  Copy  Delete	8	Sarah	Paulson		1974-12-17
<input type="checkbox"/>  Edit  Copy  Delete	9	John	Smith		1982-10-28
<input type="checkbox"/>  Edit  Copy  Delete	6	John	Smith		1971-04-18
<input type="checkbox"/>  Edit  Copy  Delete	5	Dave	Sylvando		1990-06-02
<input type="checkbox"/>  Edit  Copy  Delete	10	Patricia	Tannis		1985-05-24

Drivers who have only worked morning shifts.

This query uses several temporary tables. One table stores drivers who work morning shifts and the other stores drivers who work evening shifts. Using 'sets' logic, a third temporary table is created with drivers in the morning-working table that are not in the afternoon-working table. A join with this third table and the 'driver' table allows the driver information to then be displayed.

#Drivers Who have worked Morning Shifts

```
CREATE TEMPORARY TABLE morningworkers (driver_id int);

INSERT INTO morningWorkers (

    SELECT DISTINCT driver_id FROM driverVehicleShift

    INNER JOIN shift USING (shift_id)

    WHERE (cast(shift_start as time) BETWEEN '06:00:00' AND '09:00:00')

    AND (cast(shift_end as time) BETWEEN '07:00:00' AND '12:00:00')

);
```

#Drivers Who have worked Afternoon shifts

```
CREATE TEMPORARY TABLE afternoonWorkers(driver_id int);

INSERT INTO afternoonWorkers(

    SELECT DISTINCT driver_id FROM drivervehicleshift

    INNER JOIN shift USING (shift_id)

    WHERE (cast(shift_start as time) BETWEEN '12:00:00' AND '14:00:00')

    AND (cast(shift_end as time) BETWEEN '15:00:00' AND '17:00:00')
```

CMP2806M Scalable Database Systems Assessment

);

#Drivers Who have worked only Morning shifts

```
CREATE TEMPORARY TABLE onlyMorningWorkers(driver_id int);
```

```
INSERT INTO onlyMorningWorkers(
```

```
    SELECT * FROM morningWorkers
```

```
        WHERE driver_id NOT IN
```

```
        (SELECT * FROM afternoonWorkers)
```

```
);
```

#Show Names

```
SELECT * FROM driver
```

```
INNER JOIN onlyMorningWorkers USING (driver_id);
```

Output:

driver_id	first_name	last_name	DOB
3	Bianca	Del Rio	1975-06-27
8	Sarah	Paulson	1974-12-17
9	John	Smith	1982-10-28

Procedures

These procedures are based on the “vehicle location at any time” and the “number of items delivered by a driver” queries.

Locate_vehicle

```
DELIMITER //
```

```
CREATE PROCEDURE locate_vehicle
```

```
(IN my_vehicle int(6), my_time DATETIME)
```

```
BEGIN
```

```
    SELECT * FROM location WHERE location_id = (SELECT location_id FROM
```

```
        vehiclelocation where vehicle_id = my_vehicle AND gps_time = my_time);
```

```
END //
```

```
DELIMITER ;
```


CALL locate_vehicle(2,'2020-02-01 10:00:00');

Items_delivered

DELIMITER //

CREATE PROCEDURE items_delivered

(IN shiftAllocation INT(6))

BEGIN

SET @shiftStart = (SELECT shift_start FROM shift WHERE shift_id = shiftAllocation);

SET @shiftEnd = (SELECT shift_end FROM shift WHERE shift_id = shiftAllocation);

SET @driverVehicle = (SELECT vehicle_id FROM drivervehicleshift WHERE shift_id = shiftAllocation);

SELECT COUNT(*) 'Items Delivered'

FROM item i

INNER JOIN delivery d

USING(delivery_id)

WHERE d.vehicle_id = @driverVehicle AND (d.delivery_time BETWEEN @shiftStart AND @shiftEnd);

END //

DELIMITER ;

CALL items_delivered(2);

Security & Scalability

For the current database, a simple security plan would suffice, as the company consists of only 10 drivers and 5 vehicles.

There would be a need to have a dedicated database administrator permitted to use DDL and DML queries. The head administrator would have MySQL "SUPER" privileges. For other DBMS users, User accounts would be beneficial, with access requiring authentication. These would be created using the "CREATE USER" syntax. If the company were to scale up and require more administration, varying roles could be issued to maintain the database (using "CREATE ROLE"). There could be user-defined roles that manage recipient information and

roles that manage employee information. Users could be given varying permissions using the "GRANT" and "DENY" keywords. If a member of the DBMS administration team left the company, or changed roles, "REVOKE" would be used to remove permissions (MySQL, 2021).

Storing the database in the cloud would be more beneficial in this case as the company would not have to pay for physical hardware storage. The remote location of a cloud database would be highly secure and a good way to outsource physical security concerns. Server-level authorization could be used here prior to accessing the remote database. Cloud technology helps with vertical scaling, with more server resources allocated to the database easily if needed. (Microsoft Learn 2021).

Both the use of transactions and backups of the database would be important in retaining vital data. These allow the database to be 'rolled-back' to the ideal state in the case of a loss of data, or database attack (Microsoft 2012).

SQL injection attacks could occur against this database. Attackers can insert malicious code in areas where the public influence the data, through user inputs. For example, a particular entry in a name, email or even phone field could lead to an injection (Positive Technologies 2021). To combat this, the web application backend could include a form of input validation. This could disallow SQL key words and characters (such as " or ;) from being entered in forms. Character-escaping functions for this input could prevent possible SQL syntax conflicts. It could also ensure phone numbers or emails adhere to a strict format. Stored procedures, such as 'locate_vehicle' or 'items_delivered' allow statements to be automatically parameterized to help combat these attacks (Tiwari, 2015).

Conclusion

A delivery tracking relational database has successfully been created. This required organised design techniques and normalisation to remove redundant data. Specific queries have been defined, and two have been recreated as stored procedures. Security considerations are important after creation of a database and grow in importance as the database scales up.

References

Maior H, Fathulla K (2020/2021). Lecture Content, CMP2806M Scalable Database Systems, University of Lincoln, delivered 15 October 2020- 21 January 2021.[ONLINE] Available at: https://blackboard.lincoln.ac.uk/webapps/blackboard/content/listContent.jsp?course_id= 146760_1&content_id= 2965903_1&mode=reset . [Accessed 29 January 2021].

Microsoft (2021) *Database design basics* - Access. [ONLINE] Available at: <https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5>. [Accessed 28 January 2021].

Microsoft (Official Academic Course) (2012) *Administering a Database in Database Administration Fundamentals*. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ, US p84-106.

Microsoft (Official Academic Course) (2012) *Understanding Data Storage in Database Administration Fundamentals*. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ, US p61-80.

Microsoft Corporation (2018) *Microsoft Excel*. Available at: <https://office.microsoft.com/excel>.

Microsoft Learn (2021). *Benefits of cloud computing - Learn* . [ONLINE] Available at: <https://docs.microsoft.com/en-us/learn/modules/principles-cloud-computing-dynamics-365-deployment/4-benefits-cloud-computing>. [Accessed 29 January 2021].

MySQL (2021) *Reference Manual:: 6.2.2 Privileges Provided by MySQL*. [ONLINE] Available at: https://dev.mysql.com/doc/refman/5.6/en/privileges-provided.html#priv_super. [Accessed 29 January 2021].

Positive Technology (2021) *How to Prevent SQL Injection: Attacks and Defense Techniques - Tutorial and Best Practices*. [ONLINE] Available at: <https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-prevent-sql-injection-attacks/>. [Accessed 29 January 2021].

Tiwari Y, Tiwari M (2015) 'A Study of SQL of Injections Techniques and their Prevention Method', *International Journal of Computer Applications*, vol. 114, no. 17, p. 31-33.

Appendix

APPENDIX A – Excel tables

driver

driver_id	first_name	last_name	DOB
1	Ginni	Lemon	1989-02-07
2	Veronica	Green	1985-04-24
3	Bianca	Del Rio	1975-06-27
4	Rupaul	Charles	1960-11-17
5	Dave	Sylvando	1990-06-02
6	John	Smith	1971-04-18
7	Kathy	Burke	1946-06-13
8	Sarah	Paulson	1974-12-17
9	John	Smith	1982-10-28
10	Patricia	Tannis	1985-05-24

vehicle

vehicle_id	license	type	purchase_year
1	HE11O55	van'	2018
2	QU33N12	van'	2009
3	DEL1V3R	van'	2015
4	FA5TT1M	car'	2017
5	ONT1ME1	other'	2020

driverPhone

driver_id	type	phone
1	mobile	07501654600'
2	mobile	01230156900'
3	home	0116 496 0690'
4	mobile	07700 900909'
5	mobile	07700 900126'
6	home	0118 496 0804'
7	home	0161 496 0904'
8	mobile	07700 900989
9	home	0115 496 0444'
10	mobile	07700 900318'
2	home	0113 496 0810'

CMP2806M Scalable Database Systems Assessment

shift

shift_id	shift_start	shift_end
	2020-02-01	
1	08:00:00'	2020-02-01 11:55:00'
	2020-02-01	
2	08:00:00'	2020-02-01 11:55:00'
	2020-02-01	
3	12:00:00'	2020-02-01 16:00:00'
	2020-02-02	
4	08:00:00'	2020-02-02 11:55:00'
	2020-02-02	
5	08:00:00'	2020-02-02 11:55:00'
	2020-02-02	
6	08:00:00'	2020-02-02 11:55:00'
	2020-02-02	
7	12:00:00'	2020-02-02 11:55:00'
	2020-02-03	
8	08:00:00'	2020-02-03 11:55:00'
	2020-02-03	
9	12:00:00'	2020-02-03 16:00:00'
	2020-02-04	
10	08:00:00'	2020-02-04 11:55:00'
	2020-02-04	
11	12:00:00'	2020-02-04 16:00:00'
	2020-02-05	
12	08:00:00'	2020-02-05 08:00:00'
	2020-02-05	
13	12:00:00'	2020-02-05 16:00:00'
	2020-02-06	
14	08:00:00'	2020-02-06 11:55:00'
	2020-02-06	
15	12:00:00'	2020-02-06 16:00:00'
	2020-02-07	
16	08:00:00'	2020-02-07 11:55:00'
	2020-02-07	
17	12:00:00'	2020-02-07 16:00:00'
	2020-02-08	
18	08:00:00'	2020-02-08 11:55:00'
	2020-02-08	
19	12:00:00'	2020-02-08 16:00:00'

CMP2806M Scalable Database Systems Assessment

driverVehicleShift

shift_id	driver_id	vehicle_id
1	1	2
2	2	1
3	4	4
4	3	3
5	1	1
6	9	2
7	10	5
8	8	4
9	1	5
10	3	1
11	2	3
12	10	2
13	1	4
14	4	4
15	5	1
16	3	5
17	6	3
18	8	1
19	7	2

location

location_id	building_no	building_name	street	city	postcode
1	2	Lincoln Lake Depot	Lake st	Lincoln	LN5 78P
2	3	Wells Court	Darling rd	Lincoln	LN3 65O
3	5	Oak apts	Tree ln	Lincoln	LN7 89U
4	NULL	NULL	Camel rd	Nottingham	NG1 7YU
5	99	NULL	Specs st	Nottingham	NG1 7YU
6	32	NULL	Hello rd	Lincoln	LN5 9P6
7	NULL	NULL	Somewhere st	Lincoln	LN4 TU6
8	34	NULL	Magpie rd	Boston	PE21 3ZX
9	NULL	Billy's café	Billy st	Boston	PE21 6YK
10	4	NULL	Covid rd	Lincoln	LN4 6YJ
11	NULL	NULL	Here ln	Lincoln	LN4 OP2
12	44	NULL	Uni rd	Lincoln	LN5 HG2
13	5	Chatty flats	Zoo ln	Nottingham	NG2 4RR
14	NULL	NULL	Pencil st	Nottingham	NG5 CR3
15	NULL	Applewood	Cute ln	Lincoln	LN5 TR4

CMP2806M Scalable Database Systems Assessment

vehicleLocation

timestamp_id	vehicle_id	location_id	gps_time
1	2	1	2020-02-01 08:00:00'
2	2	2	2020-02-01 09:00:00
3	2	3	2020-02-01 10:00:00
4	1	1	2020-02-01 08:00:00
5	1	4	2020-02-01 09:00:00
6	1	5	2020-02-01 10:00:00
7	4	1	2020-02-01 12:00:00
8	4	6	2020-02-01 13:00:00
9	4	7	2020-02-01 14:00:00
10	4	8	2020-02-01 15:00:00
11	3	1	2020-02-02 08:00:00
12	3	9	2020-02-02 09:00:00
13	3	10	2020-02-02 10:00:00
14	1	1	2020-02-02 08:00:00
15	1	11	2020-02-02 09:00:00

recipient

recipient_id	first_name	last_name	email
1	Karen	Smith	karensmith@outlook.com
2	Prince	Charles	someprince@yahoo.com
3	Jill	Jenkins	NULL
4	Harry	Hill	funnyman@gmail.com
5	Billie	Piper	NULL
6	David	Tennant	thedoctor@universalmail.com
7	Katy	Perry	NULL
8	Drew	Barrymore	NULL
9	Kevin	Perry	NULL
10	John	Smith	commonname@hotmail.com
11	Henry	Caville	NULL
12	Julie	Andrews	NULL
13	Joanna	Lumley	NULL
14	Jennifer	Saunders	NULL
15	Dawn	French	NULL

CMP2806M Scalable Database Systems Assessment

address

address_id	building_no	building_name	street	city	postcode
1	2	Acorn flats	Wood st	Lincoln	LN6 789
2	3	NULL	Early rd	Nottingham	NG7 85K
3	6	NULL	Scent ln	Lincoln	LN5 R2D
4	122	Game Creations	Tech st	Boston	PE21 3EF
5	134	NULL	Notts ln	Nottingham	NG1 FF4
6	56	NULL	Family rd	Lincoln	LN5 5HK
7	97	NULL	Crisis st	Lincoln	LN4 SSD
8	12	NULL	Help rd	Lincoln	LN3 EWJ
9	4	NULL	Something st	Boston	PE23 YUM
10	2	Terry's Chocolates	Luxury rd	Boston	PE21 3FR
11	45	NULL	Cranberry st	Nottingham	NG2 TTY
12	1	NULL	Some rd	Lincoln	LN5 8JG
13	47	NULL	Machine ln	Boston	PE20 XDR
14	33	NULL	David rd	Lincoln	LN4 HHJ
15	1	NULL	Henry ln	Nottingham	NG1 K8Y

delivery

deliver_id	address_id	recipient_id	vehicle_id	estimated_delivery	delivery_time
1	1	1	1	2020-02-01	2020-02-01 08:15:00
2	3	2	1	2020-02-01	2020-02-01 08:30:00
3	2	3	1	2020-02-01	2020-02-01 08:30:00
4	4	4	4	2020-02-01	2020-02-01 13:05:10
5	5	4	4	2020-02-02	2020-02-01 13:10:18
6	6	5	2	2020-02-03	2020-02-01 09:04:25
7	7	6	3	2020-02-03	2020-02-02 10:05:00
8	8	7	2	2020-02-03	2020-02-02 09:00:00
9	10	8	3	2020-02-04	2020-02-02 11:00:00
10	9	9	5	2020-02-04	2020-02-03 14:00:00
11	11	10	5	2020-02-04	2020-02-03 15:00:00
12	13	10	5	2020-02-04	2020-02-03 15:25:00
13	12	11	1	2020-02-04	2020-02-04 09:15:00
14	15	12	2	2020-02-05	2020-02-05 09:10:00

CMP2806M Scalable Database Systems Assessment

15	14	13	2	2020-02-05	NULL
----	----	----	---	------------	------

item

item_id	delivery_id	fragile	weight_kg
1	1	TRUE	2.1
2	1	FALSE	3.4
3	1	TRUE	5.6
4	2	TRUE	13.5
5	2	TRUE	14.2
6	3	TRUE	10.23
7	4	FALSE	2
8	5	FALSE	1
9	5	FALSE	1.4
10	6	FALSE	6.34
11	7	TRUE	1
12	8	FALSE	2.5
13	9	FALSE	3.5
14	10	FALSE	2.1
15	11	FALSE	4.5