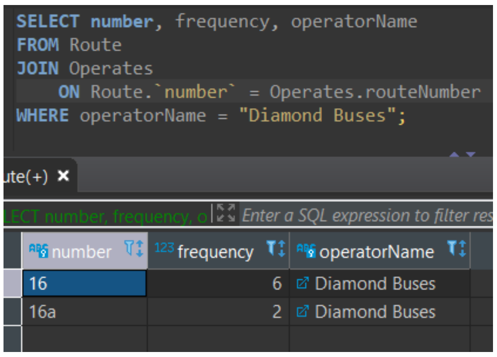
Part 2:

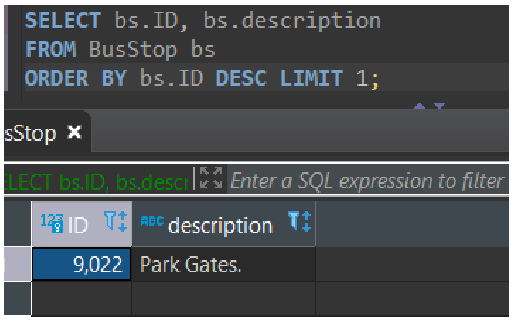
a)



```
SELECT number, frequency, operatorName
FROM Route
JOIN Operates
    ON Route.`number` = Operates.routeNumber
WHERE operatorName = "Diamond Buses";
```

number	frequency	operatorName
16	6	Diamond Buses
16a	2	Diamond Buses

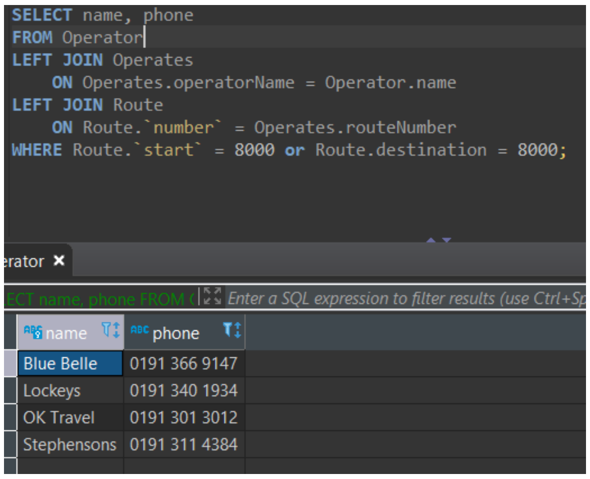
b)



```
SELECT bs.ID, bs.description
FROM BusStop bs
ORDER BY bs.ID DESC LIMIT 1;
```

ID	description
9,022	Park Gates.

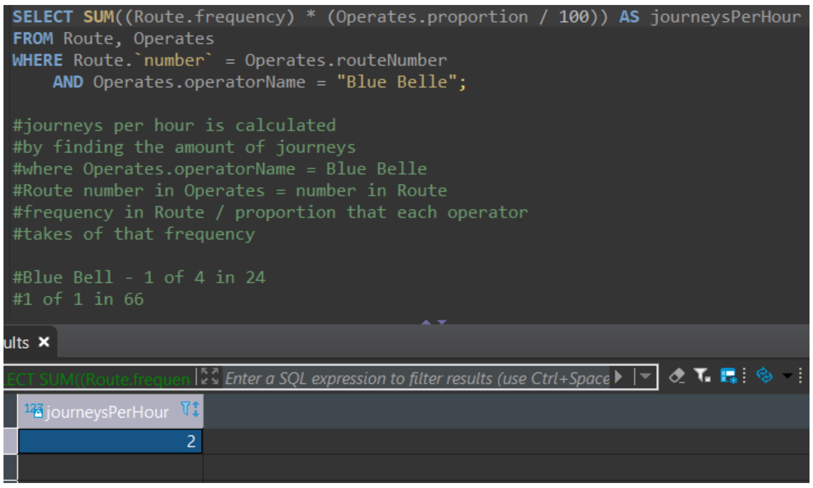
c)



```
SELECT name, phone
FROM Operator
LEFT JOIN Operates
    ON Operates.operatorName = Operator.name
LEFT JOIN Route
    ON Route.`number` = Operates.routeNumber
WHERE Route.`start` = 8000 or Route.destination = 8000;
```

name	phone
Blue Belle	0191 366 9147
Lockeys	0191 340 1934
OK Travel	0191 301 3012
Stephensons	0191 311 4384

d)



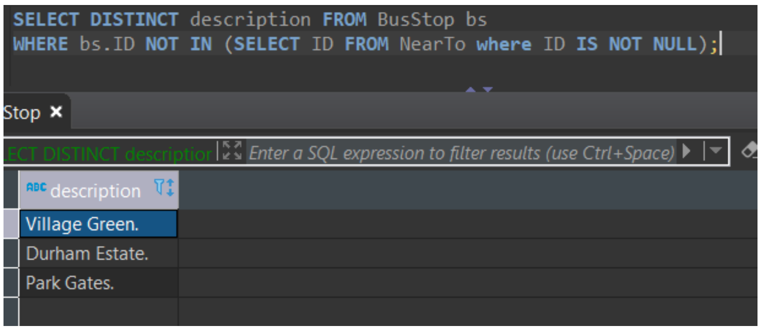
```
SELECT SUM((Route.frequency) * (Operates.proportion / 100)) AS journeysPerHour
FROM Route, Operates
WHERE Route.`number` = Operates.routeNumber
    AND Operates.operatorName = "Blue Belle";

#journeys per hour is calculated
#by finding the amount of journeys
#where Operates.operatorName = Blue Belle
#Route number in Operates = number in Route
#frequency in Route / proportion that each operator
#takes of that frequency

#Blue Bell - 1 of 4 in 24
#1 of 1 in 66
```

journeysPerHour
2

e)



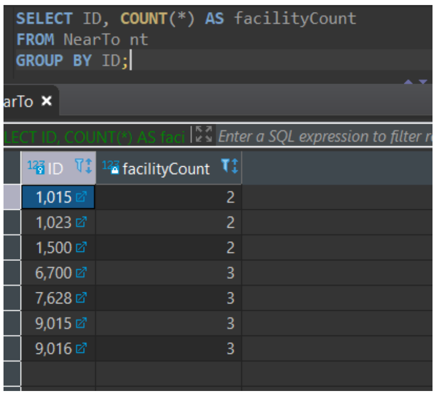
```
SELECT DISTINCT description FROM BusStop bs
WHERE bs.ID NOT IN (SELECT ID FROM NearTo where ID IS NOT NULL);
```

Stop ×

ECT DISTINCT descriptior | Enter a SQL expression to filter results (use Ctrl+Space)

description
Village Green.
Durham Estate.
Park Gates.

f)



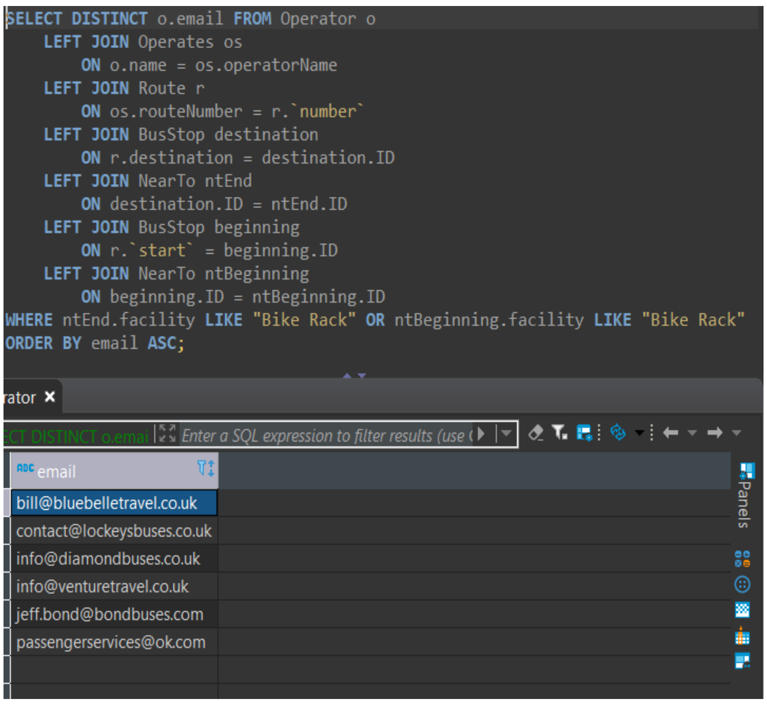
```
SELECT ID, COUNT(*) AS facilityCount
FROM NearTo nt
GROUP BY ID;
```

arTo ×

LECT ID, COUNT(*) AS faci | Enter a SQL expression to filter r

ID	facilityCount
1,015	2
1,023	2
1,500	2
6,700	3
7,628	3
9,015	3
9,016	3

g)



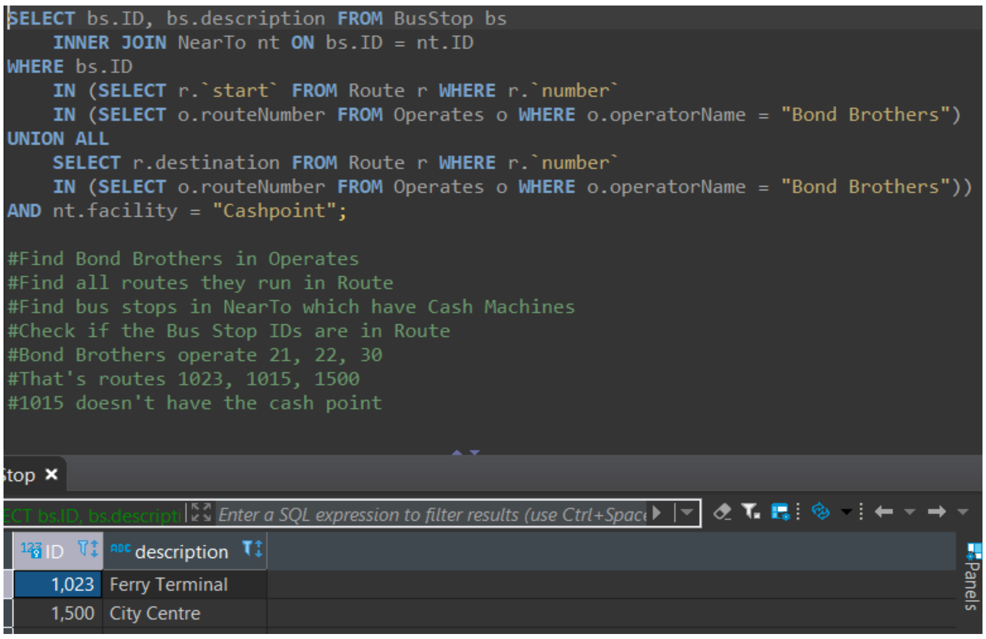
```
SELECT DISTINCT o.email FROM Operator o
    LEFT JOIN Operates os
        ON o.name = os.operatorName
    LEFT JOIN Route r
        ON os.routeNumber = r.`number`
    LEFT JOIN BusStop destination
        ON r.destination = destination.ID
    LEFT JOIN NearTo ntEnd
        ON destination.ID = ntEnd.ID
    LEFT JOIN BusStop beginning
        ON r.`start` = beginning.ID
    LEFT JOIN NearTo ntBeginning
        ON beginning.ID = ntBeginning.ID
WHERE ntEnd.facility LIKE "Bike Rack" OR ntBeginning.facility LIKE "Bike Rack"
ORDER BY email ASC;
```

rator ×

CT DISTINCT o.emai | Enter a SQL expression to filter results (use

email
bill@bluebelletravel.co.uk
contact@lockeysbuses.co.uk
info@diamondbuses.co.uk
info@venturetravel.co.uk
jeff.bond@bondbuses.com
passengerservices@ok.com

h)

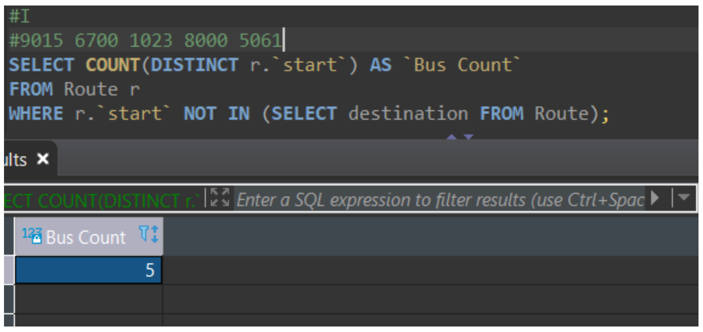


```sql
SELECT bs.ID, bs.description FROM BusStop bs
    INNER JOIN NearTo nt ON bs.ID = nt.ID
WHERE bs.ID
    IN (SELECT r.`start` FROM Route r WHERE r.`number`
    IN (SELECT o.routeNumber FROM Operates o WHERE o.operatorName = "Bond Brothers")
UNION ALL
    SELECT r.destination FROM Route r WHERE r.`number`
    IN (SELECT o.routeNumber FROM Operates o WHERE o.operatorName = "Bond Brothers"))
AND nt.facility = "Cashpoint";

#Find Bond Brothers in Operates
#Find all routes they run in Route
#Find bus stops in NearTo which have Cash Machines
#Check if the Bus Stop IDs are in Route
#Bond Brothers operate 21, 22, 30
#That's routes 1023, 1015, 1500
#1015 doesn't have the cash point
```

ID	description
1,023	Ferry Terminal
1,500	City Centre

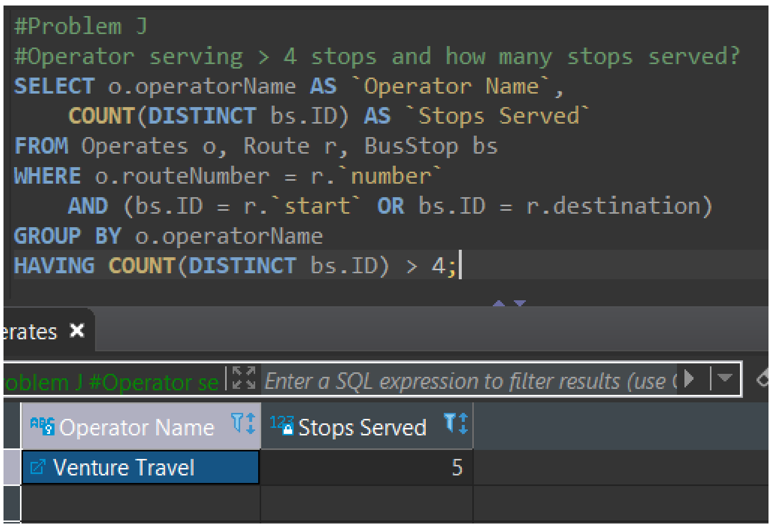
I)



```sql
#I
#9015 6700 1023 8000 5061
SELECT COUNT(DISTINCT r.`start`) AS `Bus Count`
FROM Route r
WHERE r.`start` NOT IN (SELECT destination FROM Route);
```

Bus Count
5

j)



```sql
#Problem J
#Operator serving > 4 stops and how many stops served?
SELECT o.operatorName AS `Operator Name`,
    COUNT(DISTINCT bs.ID) AS `Stops Served`
FROM Operates o, Route r, BusStop bs
WHERE o.routeNumber = r.`number`
    AND (bs.ID = r.`start` OR bs.ID = r.destination)
GROUP BY o.operatorName
HAVING COUNT(DISTINCT bs.ID) > 4;
```

Operator Name	Stops Served
Venture Travel	5

Part 3:

a) Explain how the suggested changes would break First Normal Form.

By implementing the changes described in the question, First Normal Form would be broken because there could potentially be an imbalance of data. The possibility of there being one or more customer service agents that need to be added would mean that in some rows, there might be two or three names inside on column. This would be messy, hard to organise, and most importantly for First Normal Form, the values are not atomic, but multi-valued.

b) What changes would you make to ensure the modifications would conform to First Normal Form?

I would create a new table for the customer service agents, since their details are likely to be different to those of the operator as a whole and there could be multiple agents, yet only one row for the operator. This table would include a foreign key of the operator's name, which are assumed to be unique. Each agent would also have their own ID created as a primary key, which could be as simple as a random number or code, such as 'A01' or 'Agent1' or simply '1'. It is assumed that a customer service agent only works for one operator. If not, they would be treated as separate agents for the sake of the table. The table would still include the suggested attributes of 'Agent Name', 'Agent Phone Number', and 'Agent Email'. Using the operator's name as a foreign key and leaving it out of the primary key avoids creating a composite key, which is important as there may be multiple agents for an operator, therefore needing a numerical value to keep track of this. This also avoids any confusion if two agents shared a name or the same office telephone number or email.

c) Would your changes conform to Second and Third Normal Form? Explain your answer.

Since the idea in part b has avoided using a composite key, the changes would not break Second Normal Form, since one piece of data does not necessarily have to rely on another to still be true. If an agent left an operator only their data would need to be deleted and it would not affect the other data in the table, nor the operator's data, whereas it might had a new table for customer service agents not been created.
Additionally, adding a new agent is simple and normalised because you can add a new agent to the customer service agents table only using the operator's name, not affecting any of the other table data.
I don't think there is any transitive data in this table which would break Third Normal Form. The phone numbers and emails of the agents again do not need to rely on other pieces of data, although it is possible that they could be altered by the actions of the company, this is not necessarily true. It would not seem realistic to break down this table any further, as the data included is simply the Agent's contact information and ID number. The ID number as a primary key does not rely on any specific information from the agent, nor the operator's name. Each value can be self-contained.