Advanced SQL

By George ALSHOUFI

Database Overview

I started with analysing, the database schema tables, columns, and records. Allowing me understand what data is available, how it's organised, and the tables relationships. Also used for results validation.

Listing all tables

```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_Bus_Depots |
+----------------------+
| Ability              |
| Bus                  |
| BusDriver            |
| BusType              |
| Cleaner              |
| Depot                |
| Restriction          |
| Route                |
| Training             |
+----------------------+
9 rows in set (0.00 sec)
```

Displaying all records From **Ability** table

```
mysql> SELECT * FROM Ability;
+------+-----+
| bdno | rno |
+------+-----+
| 007  | 10  |
| 008  | 10  |
| 008  | 11  |
| 001  | 6   |
| 007  | 6   |
| 001  | 7   |
| 009  | 7   |
| 001  | 8   |
+------+-----+
8 rows in set (0.00 sec)
```

Ability table structure

```
mysql> DESCRIBE Ability;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| bdno  | varchar(5) | NO   | PRI |         |       |
| rno   | varchar(5) | NO   | PRI |         |       |
+-------+------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Displaying all records from **Bus** table

```
mysql> SELECT * FROM Bus;
+---------+-----------------+------+------+------+
| regno   | model           | tno  | dno  | cno  |
+---------+-----------------+------+------+------+
| A123ABC | Routemaster     | 1    | 101  | 110  |
| D345GGG | Volvo 8500      | 1    | 101  | 112  |
| D678FGH | Volvo 8700      | 2    | 101  | 110  |
| H259IJK | Daf SB220       | 3    | 102  | 114  |
| P200IJK | Mercedes 709D   | 2    | 102  | 113  |
| P300RTY | Mercedes Citaro | 4    | 102  | 113  |
| R678FDS | Daf SB220       | 1    | NULL | 110  |
+---------+-----------------+------+------+------+
7 rows in set (0.00 sec)
```

Bus table structure

```
mysql> DESCRIBE Bus;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| regno | varchar(10) | NO   | PRI |         |       |
| model | varchar(20) | YES  |     | NULL    |       |
| tno   | varchar(5)  | YES  | MUL | NULL    |       |
| dno   | varchar(5)  | YES  | MUL | NULL    |       |
| cno   | varchar(5)  | YES  | MUL | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Database Overview 2

Displaying all records From **BusDriver** table

```
mysql> SELECT * FROM BusDriver;
+------+-------------+----------+------------+------+
| bdno | bdname      | bdsalary | pcvdate    | dno  |
+------+-------------+----------+------------+------+
| 001  | Jane Brown  |  1800.00 | 1985-02-09 | 101  |
| 006  | Sally Smith |  1750.00 | 1996-03-09 | NULL |
| 007  | James Bond  |  1500.00 | 1999-01-09 | 102  |
| 008  | Maggie May  |  2200.00 | 2000-01-09 | 102  |
| 009  | Jack Jones  |  1400.00 | 2001-08-09 | 101  |
| 010  | Peter Piper |  3500.00 | 2004-06-09 | 104  |
| 011  | John Peel   |  2000.00 | 2005-02-09 | 102  |
+------+-------------+----------+------------+------+
7 rows in set (0.00 sec)
```

Displaying all records From **BusType** table

```
mysql> SELECT * FROM BusType;
+-----+---------------+
| tno | tdescript     |
+-----+---------------+
| 1   | double-decker |
| 2   | metrobus      |
| 3   | midibus       |
| 4   | bendy bus     |
| 5   | open top      |
+-----+---------------+
5 rows in set (0.00 sec)
```

Displaying all records From **Cleaner** table

```
mysql> SELECT * FROM Cleaner;
+-----+-------+---------+------+
| cno | cname | csalary | dno  |
+-----+-------+---------+------+
| 110 | John  | 2550.00 | 101  |
| 111 | Jean  | 2500.00 | 101  |
| 112 | Betty | 2400.00 | 102  |
| 113 | Vince | 2800.00 | 102  |
| 114 | Jay   | 3000.00 | 102  |
| 115 | Doug  | 2000.00 | 102  |
| 116 | Geeta | 4000.00 | NULL |
+-----+-------+---------+------+
7 rows in set (0.00 sec)
```

Ability table structure

```
mysql> DESCRIBE BusDriver;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| bdno     | varchar(5)   | NO   | PRI |         |       |
| bdname   | varchar(20)  | YES  |     | NULL    |       |
| bdsalary | decimal(6,2) | YES  |     | NULL    |       |
| pcvdate  | date         | YES  |     | NULL    |       |
| dno      | varchar(5)   | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

BusType table structure

```
mysql> DESCRIBE BusType;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| tno       | varchar(5)  | NO   | PRI |         |       |
| tdescript | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Cleaner table structure

```
mysql> DESCRIBE Cleaner;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| cno     | varchar(5)  | NO   | PRI |         |       |
| cname   | varchar(20) | YES  |     | NULL    |       |
| csalary | decimal(6,2)| YES  |     | NULL    |       |
| dno     | varchar(5)  | YES  | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Database Overview 3

Displaying all records From **Depot** table

```
mysql> SELECT * FROM Depot;
+-----+-----------+--------------+
| dno | dname     | daddress     |
+-----+-----------+--------------+
| 101 | Holloway  | Camden Road  |
| 102 | Hornsey   | High Road    |
| 104 | Islington | Upper Street |
+-----+-----------+--------------+
3 rows in set (0.00 sec)
```

Displaying all records From **Restriction** table

```
mysql> SELECT * FROM Restriction;
+-----+-----+
| rno | tno |
+-----+-----+
| 10  | 1   |
| 11  | 1   |
| 6   | 1   |
| 7   | 1   |
| 10  | 2   |
| 11  | 2   |
| 6   | 2   |
| 7   | 2   |
| 10  | 3   |
| 11  | 3   |
| 6   | 3   |
| 8   | 3   |
| 10  | 4   |
| 11  | 4   |
| 6   | 4   |
| 8   | 4   |
+-----+-----+
16 rows in set (0.01 sec)
```

Displaying all records From **Route** table

```
mysql> SELECT * FROM Route;
+-----+----------------------+------+
| rno | rdescript            | dno  |
+-----+----------------------+------+
| 10  | Tottenham/Angel      | 102  |
| 11  | Islington/Highgate   | 102  |
| 6   | Camden/Golders Green | 101  |
| 7   | Finchley/Tottenham   | 101  |
| 8   | Hendon/Muswell Hill  | 101  |
+-----+----------------------+------+
5 rows in set (0.00 sec)
```

Displaying all records From **Training** table

```
mysql> SELECT * FROM Training;
+------+-----+--------------+
| bdno | tno | trainingdate |
+------+-----+--------------+
| 001  | 1   | 2006-01-09   |
| 001  | 2   | 2006-01-09   |
| 006  | 2   | 2006-02-09   |
| 007  | 1   | 2006-02-09   |
| 007  | 2   | 2006-02-09   |
| 007  | 3   | 2006-03-09   |
| 008  | 2   | 2006-03-09   |
| 008  | 3   | 2006-03-09   |
| 008  | 4   | 2006-04-09   |
| 009  | 3   | 2006-04-09   |
| 009  | 4   | 2006-05-09   |
| 011  | 1   | 2006-05-09   |
| 011  | 2   | 2006-05-09   |
| 011  | 3   | 2006-06-09   |
| 011  | 4   | 2006-06-09   |
| 011  | 5   | 2006-06-09   |
+------+-----+--------------+
16 rows in set (0.00 sec)
```

Depot table structure

```
mysql> DESCRIBE Depot;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| dno      | varchar(5)  | NO   | PRI |         |       |
| dname    | varchar(20) | YES  |     | NULL    |       |
| daddress | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Restriction table structure

```
mysql> DESCRIBE Restriction;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| rno   | varchar(5) | NO   | PRI |         |       |
| tno   | varchar(5) | NO   | PRI |         |       |
+-------+------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Route table structure

```
mysql> DESCRIBE Route;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| rno       | varchar(5)  | NO   | PRI |         |       |
| rdescript | varchar(30) | YES  |     | NULL    |       |
| dno       | varchar(5)  | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Training table structure

```
mysql> DESCRIBE Training;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| bdno         | varchar(5) | NO   | PRI |         |       |
| tno          | varchar(5) | NO   | PRI |         |       |
| trainingdate | date       | YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Exercise 1

Exercise 1.1

(Project, RESTRICT) List all drivers number and name) who have a salary of less than 1800.

Solution

```
/*! 1- (Project, RESTRICT) */
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name'
FROM BusDriver bd
WHERE bdsalary < 1800
ORDER BY bd.bdno;
```

Result

```
mysql> SELECT bdno AS 'Driver Number', bdname AS 'Driver Name'
    -> FROM BusDriver bd
    -> WHERE bdsalary < 1800
    -> ORDER BY bd.bdno;
+---------------+-------------+
| Driver Number | Driver Name |
+---------------+-------------+
| 006           | Sally Smith |
| 007           | James Bond  |
| 009           | Jack Jones  |
+---------------+-------------+
3 rows in set (0.00 sec)
```

Reflection

Used the **WHERE** clause and the less than < operator to determine which records to select from the BusDriver Table.

Code explanation

Exercise 1.1

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
bdno AS 'Driver Number', bdname AS 'Driver Name'	Select the bdno, and bdname fields from the specified table name, **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM BusDriver bd	Specify the table BusDriver, and **bd** used to assign the BusDriver table alias as **bd**.
WHERE bdsalary < 1800	Used to specify criteria (Salary less than 1800) in the result set returned from the query. Used with the smaller than operator.
ORDER BY bd.bdno	Used to specify the sort order of the result set, sorted by the ascending order of Bus Driver Number.

Exercise 1.2

(Conditional operator LIKE) List all bus drivers (number and name) whose name begins with J.

Solution

```
/*! 2- Conditional operator LIKE */
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name'
FROM BusDriver bd
WHERE bdname LIKE "J%"
ORDER BY bd.bdno;
```

Result

```
mysql> SELECT bdno AS 'Driver Number', bdname AS 'Driver Name'
    -> FROM BusDriver bd
    -> WHERE bdname LIKE "J%"
    -> ORDER BY bd.bdno;
+---------------+-------------+
| Driver Number | Driver Name |
+---------------+-------------+
| 001           | Jane Brown  |
| 007           | James Bond  |
| 009           | Jack Jones  |
| 011           | John Peel   |
+---------------+-------------+
4 rows in set (0.00 sec)
```

Reflection

Used the **LIKE** logical operator with the "J%" pattern. The Like operator can be TRUE if the operand equals a pattern. Note: The default character set and collation are utf8mb4 and utf8mb4_0900_ai_ci, so nonbinary string comparisons are case-insensitive by default. If we are comparing a column and a string that both have the utf8mb4 character set, we can use the COLLATE operator to cause either operand to have the utf8mb4_0900_as_cs or utf8mb4_bin collation:

Code explanation

Exercise 1.2

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name'	Select the bdno, and bdname fields from the specified table name, **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM BusDriver bd	Specify the table BusDriver, and **bd** used to assign the BusDriver table alias as **bd**.
WHERE bdname LIKE "j%"	Used to specify criteria (Name begins with J) in the result set returned from the query. Used with the **Like** operator.
ORDER BY bd.bdno	Used to specify the sort order of the result set, sorted by the ascending order of Bus Driver Number.

Exercise 1.3

(Conditional operator BETWEEN) List all bus drivers details for those drivers who have a salary between 2000 and 4000

Solution

```
/*! 3- Conditional operator BETWEEN */
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name',
bdsalary AS 'Salary', pcvdate AS 'PCV Date', dno AS 'Depot No'
FROM BusDriver bd
WHERE bdsalary BETWEEN 2000 AND 4000
ORDER BY bd.bdno;
```

Result

```
mysql> SELECT bdno AS 'Driver Number', bdname AS 'Driver Name',
    -> bdsalary AS 'Salary', pcvdate AS 'PCV Date', dno AS 'Depot No'
    -> FROM BusDriver bd
    -> WHERE bdsalary BETWEEN 2000 AND 4000
    -> ORDER BY bd.bdno;
+---------------+-------------+---------+------------+-----------+
| Driver Number | Driver Name | Salary  | PCV Date   | Depot No  |
+---------------+-------------+---------+------------+-----------+
| 008           | Maggie May  | 2200.00 | 2000-01-09 | 102       |
| 010           | Peter Piper | 3500.00 | 2004-06-09 | 104       |
| 011           | John Peel   | 2000.00 | 2005-02-09 | 102       |
+---------------+-------------+---------+------------+-----------+
3 rows in set (0.00 sec)
```

Reflection

Used the Between operator to filter the results inside the series of comparisons. In our case the salary between 2000 and 4000. An alternative solution, is using <= and > instead. Logically they're the same, but maybe useful if we are building a solution where the users choose to filter results including the clause variable or not.

Alternative Solution

```
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name',
bdsalary AS 'Salary', pcvdate AS 'PCV Date', dno AS 'Depot No'
FROM BusDriver bd
WHERE bdsalary >= 2000 AND bdsalary < 4000
ORDER BY bd.bdno;
```

Code explanation

Exercise 1.3

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
SELECT bdno AS 'Driver Number', bdname AS 'Driver Name', bdsalary AS 'Salary', pcvdate AS 'PCV Date', dno AS 'Depot No'	Select the bdno, and bdname, bdsalary, pcvdate fields from the specified table name, **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM BusDriver bd	Specify the table BusDriver, and **bd** used to assign the BusDriver table alias as **bd**.
WHERE bdsalary BETWEEN 2000 AND 4000	Used to specify criteria (Salary is between 2000 and 4000) in the result set returned from the query. Used with the **BETWEEN** operator.
ORDER BY bd.bdno	Used to specify the sort order of the result set, sorted by the ascending order of Bus Driver Number.

Exercise 1.4

(AND) List all buses (registration number and model) of type 2 which are not based at depot 101.

Solution

```
/*! 4- AND */
SELECT regno AS 'Registeration Number', model AS 'Model'
FROM Bus b
WHERE b.tno = 2 AND b.dno != 101
ORDER BY b.regno;
```

Result

```
mysql> SELECT regno AS 'Registeration Number', model AS 'Model'
    -> FROM Bus b
    -> WHERE b.tno = 2 AND b.dno != 101
    -> ORDER BY b.regno;
+---------------------+--------------+
| Registeration Number | Model        |
+---------------------+--------------+
| P200IJK             | Mercedes 709D |
+---------------------+--------------+
1 row in set (0.00 sec)
```

Reflection

Used the **AND condition** in the query to filter the data based on 2 columns values (Type, and Depot).

The AND Condition allows to compare multiple conditions in the same SELECT Statement, while returning the output only of both conditions were valid (True) in the same row.

Code explanation

Exercise 1.4

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
regno AS 'Registeration Number', model AS 'Model'	Select only these fields from the specified table name, **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM Bus b	Specify the table Bus, and **b** used to assign the **Bus** table alias as **b**.
WHERE b.tno = 2 AND b.dno != 101	Used to specify criteria (Bus of type 2 which are not based at depot 101) in the result set returned from the query. Used with the equal, **AND, and not equal** operators.
ORDER BY b.regno	Used to specify the sort order of the result set, sorted by the ascending order of Bus Registration Number.

Exercise 1.5

(Controlling duplicates using DISTINCT) List all depot numbers in the bus table. Now eliminate all duplicates.

Solution

```
/*! 5- OR */
SELECT *
FROM Bus b
WHERE model LIKE "Volvo%" OR model LIKE "Mercedes%"
ORDER BY b.model AND b.regno;
```

Result

```
mysql> SELECT *
    -> FROM Bus b
    -> WHERE model LIKE "Volvo%" OR model LIKE "Mercedes%"
    -> ORDER BY b.model AND b.regno;
+---------+----------------+------+------+------+
| regno   | model          | tno  | dno  | cno  |
+---------+----------------+------+------+------+
| D345GGG | Volvo 8500     | 1    | 101  | 112  |
| D678FGH | Volvo 8700     | 2    | 101  | 110  |
| P200IJK | Mercedes 709D  | 2    | 102  | 113  |
| P300RTY | Mercedes Citaro | 4    | 102  | 113  |
+---------+----------------+------+------+------+
4 rows in set, 4 warnings (0.00 sec)
```

Reflection

Used the **OR condition** in the query to filter the data based value of 1 column **Model,** where the model equals one of the assigned conditions.

The AND Condition allows to compare multiple conditions and return them, when either of the conditions is valid (True).

Code explanation

Exercise 1.5

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
*	Used to get all fields from the specified table name. **AS** wasn't used, in case the table can have more or less columns when the script is used.
FROM Bus b	Specify the table Bus, and **b** used to assign the **Bus** table alias as **b**.
WHERE model LIKE "Volvo%" OR model LIKE "Mercedes%";	Used to specify criteria (Models either Volvo or Mercedes) in the result set returned from the query. Used with the **LIKE** logical operator that can be true of it matches either of the model names.
ORDER BY b.model AND b.regno	Used to specify the sort order of the result set, sorted by the ascending order of Bus Model and Registration Number.

Exercise 1.6

(Controlling duplicates using DISTINCT) List all depot numbers in the bus table. Now eliminate all duplicates.

Solution

```
/*! 6- DISTINCT */
SELECT
DISTINCT dno
FROM Bus;
```

Result

```
mysql> SELECT
    -> DISTINCT dno
    -> FROM Bus;
+------+
| dno  |
+------+
| NULL |
| 101  |
| 102  |
+------+
3 rows in set (0.00 sec)
```

Reflection

DISTINCT used to fine tune the results returned from the SQL SELECT statement. Which will help to return only the unique values of the dno (depot number) by eliminating duplicates.

Code explanation

Exercise 1.6

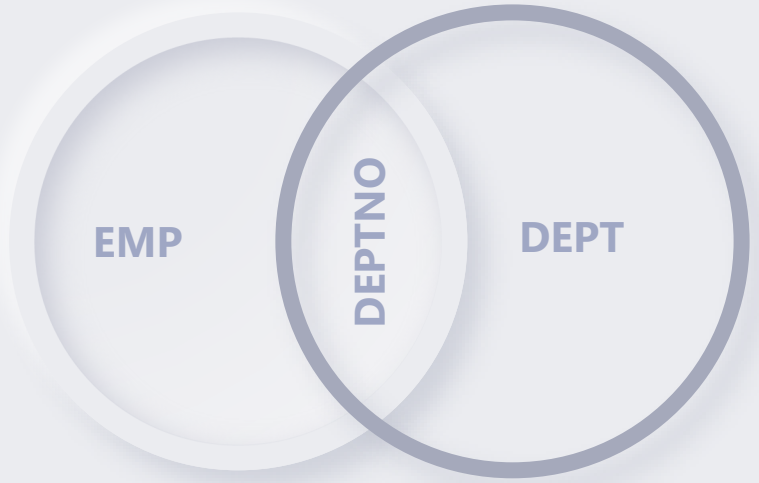
Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
DISTINCT dno	The **DISTINCT** keyword or clause helps us to retrieve the unique or different values of the column in the table.
FROM Bus	Specify the table Bus.

Exercise 1.7

(Two table Join – INNER JOIN) List all cleaners (number and name) with the name and address of their depot, but only for those cleaners located at a depot.

Solution

```
/*! 7- Two table Join - INNER JOIN */
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
d.dname AS 'Depot Name', d.daddress AS 'Depot Address'
FROM Cleaner c
INNER JOIN Depot d
ON c.dno = d.dno
ORDER BY c.cname;
```



Result

```
mysql> SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
    -> d.dname AS 'Depot Name', d.daddress AS 'Depot Address'
    -> FROM Cleaner c
    -> JOIN Depot d
    -> ON c.dno = d.dno
    -> ORDER BY c.cname;
+------------+--------------+------------+---------------+
| Cleaner No | Cleaner Name | Depot Name | Depot Address |
+------------+--------------+------------+---------------+
| 112        | Betty        | Hornsey    | High Road     |
| 115        | Doug         | Hornsey    | High Road     |
| 114        | Jay          | Hornsey    | High Road     |
| 111        | Jean         | Holloway   | Camden Road   |
| 110        | John         | Holloway   | Camden Road   |
| 113        | Vince        | Hornsey    | High Road     |
+------------+--------------+------------+---------------+
6 rows in set (0.00 sec)
```

Reflection

INNER JOIN matches each row in one table with every row in other tables, and allows to query rows that contain columns from both tables, Using **dno** as key.

Code explanation

Exercise 1.7

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot Name', d.daddress AS 'Depot Address'	Select the cno, cname, dname, and daddress fields from both table (Cleaner, and Depot), **AS** keywo optional, but used to give an expression an **alias** to the colur
FROM Cleaner c	Specify the table Cleaner, and **c** used to assign the Cleaner table alias as **c**.
INNER JOIN Depot d	A clause to select data from both tables by matching rows based on the valies in key.
ON c.dno = d.dno	Key identifier used for the JOIN clause.
ORDER BY c.cname	Order the result table by the cleaner name alphabetically.

Alternative Solution

```
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
d.dname AS 'Depot Name', d.daddress AS 'Depot Address'
FROM Cleaner c, Depot d
WHERE c.dno = d.dno
ORDER BY c.cname;
```

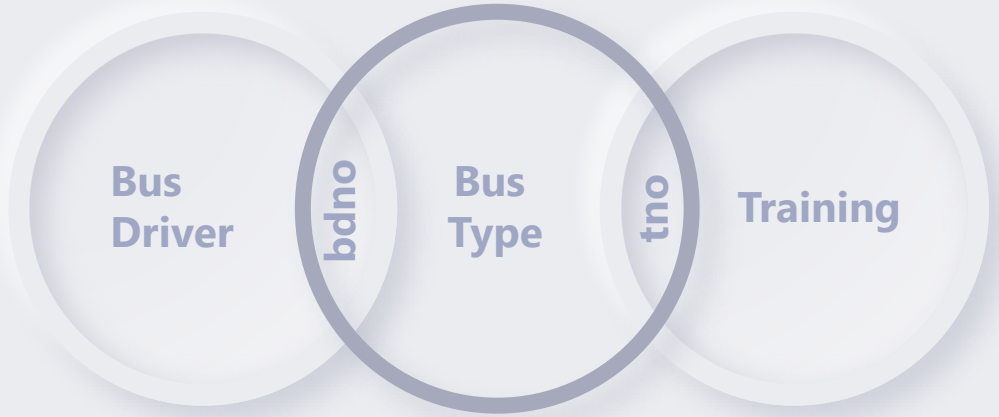
This solution was also able to get identical results from the first solution, the difference might be visible when dealing with a large amount of data while testing which one can operate faster. Also, the other syntax, I was able to define explicitly by telling MySQL which columns to join and how to join them 'Inner Join'.

Exercise 1.8

(Three table JOIN) List bus drivers (number and name) and the bus types (description) for which each bus driver has had training

Solution

```
/*! 8- Three table JOIN */
SELECT bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name',
bt.tdescript AS 'Bus Type Description'
FROM BusDriver bd
JOIN Training t
ON bd.bdno = t.bdno
JOIN BusType bt
ON bt.tno = t.tno
ORDER BY bd.bdno;
```



Result

```
mysql> SELECT bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name',
    -> bt.tdescript AS 'Bus Type Description'
    -> FROM BusDriver bd
    -> JOIN Training t
    -> ON bd.bdno = t.bdno
    -> JOIN BusType bt
    -> ON bt.tno = t.tno
    -> ORDER BY bd.bdno;
+-----------+-------------+---------------------+
| Driver No | Driver Name | Bus Type Description |
+-----------+-------------+---------------------+
| 001       | Jane Brown  | metrobus            |
| 001       | Jane Brown  | double-decker       |
| 006       | Sally Smith | metrobus            |
| 007       | James Bond  | midibus             |
| 007       | James Bond  | metrobus            |
| 007       | James Bond  | double-decker       |
| 008       | Maggie May  | bendy bus           |
| 008       | Maggie May  | midibus             |
| 008       | Maggie May  | metrobus            |
| 009       | Jack Jones  | bendy bus           |
| 009       | Jack Jones  | midibus             |
| 011       | John Peel   | midibus             |
| 011       | John Peel   | double-decker       |
| 011       | John Peel   | metrobus            |
| 011       | John Peel   | bendy bus           |
| 011       | John Peel   | open top            |
+-----------+-------------+---------------------+
16 rows in set (0.00 sec)
```

Reflection

JOIN matches each row in one table with every row in other tables, and allows to query rows that contain columns from both tables, Using **tno** and **bdno** key to link the 3 tables.

Code explanation

Exercise 1.8

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name', bt.tdescript AS 'Bus Type Description'	Select the bdno, bdname, and fields from(BusDriver, Training and BusType) tables, **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM BusDriver bd	Specify the table **BusDriver**, and **bd** used to assign the **BusDriver** table alias as **bd**.
JOIN Training t JOIN BusType bt	A clause to select data from the tables by matching rows based on the value in **key**.
ON c.dno = d.dno ON bt.tno = t.tno	Key identifier used for the **JOIN** clause to join the three tables.
ORDER BY bd.bdno	Order the result table by the cleaner name alphabetically.

Alternative Solution

```
SELECT bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name',
bt.tdescript AS 'Bus Type Description'
FROM BusDriver bd, Training t, BusType bt
WHERE bd.bdno = t.bdno AND  bt.tno = t.tno
ORDER BY bd.bdno;
```

This solution was also able to get identical results from the first solution, the difference might be visible when dealing with a large amount of data while testing which one can operate faster. Also, the other syntax, I was able to define explicitly by telling MySQL which columns to join and how to join them 'Join'.

This alternative solution is also more difficult to read and might be more challenging for other people to maintain as they have to analyse the code how we're joining the tables.

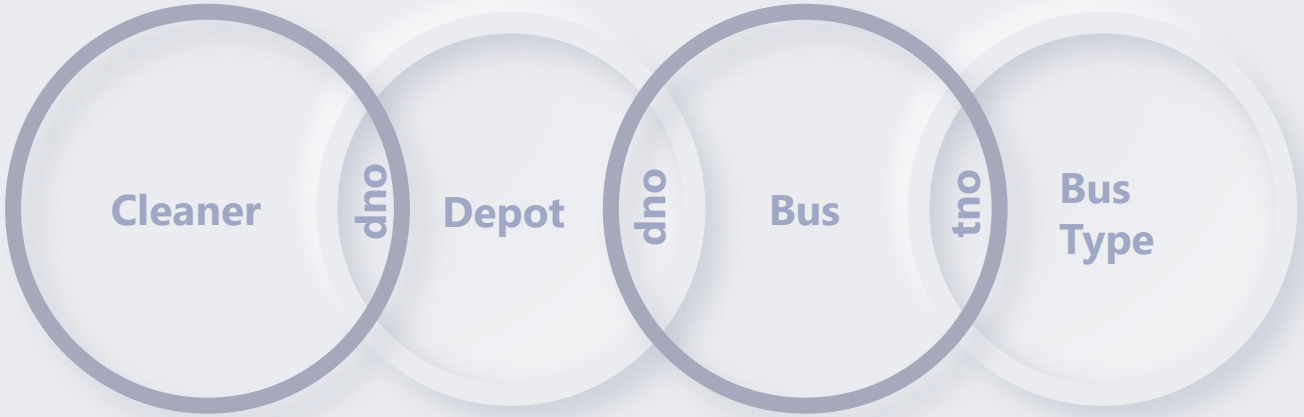
Exercise 1.9

(Three table JOIN) List bus drivers (number and name) and the bus types (description) for which each bus driver has had training

Solution

```
/*! 9- Four table JOIN */
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot No',
b.regno AS 'Bus Registeration No', bt.tno AS 'Bus Type'

FROM Cleaner c
Join Depot d
ON c.dno = d.dno
JOIN Bus b
ON d.dno = b.dno
JOIN BusType bt
ON bt.tno = b.tno
ORDER BY c.cname;
```



Result

```
mysql> SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot No',
    -> b.regno AS 'Bus Registeration No', bt.tno AS 'Bus Type'
    ->
    -> FROM Cleaner c
    -> Join Depot d
    -> ON c.dno = d.dno
    -> JOIN Bus b
    -> ON d.dno = b.dno
    -> JOIN BusType bt
    -> ON bt.tno = b.tno
    -> ORDER BY c.cname;
+------------+--------------+----------+---------------------+----------+
| Cleaner No | Cleaner Name | Depot No | Bus Registeration No | Bus Type |
+------------+--------------+----------+---------------------+----------+
| 112        | Betty        | Hornsey  | H259IJK             | 3        |
| 112        | Betty        | Hornsey  | P200IJK             | 2        |
| 112        | Betty        | Hornsey  | P300RTY             | 4        |
| 115        | Doug         | Hornsey  | P200IJK             | 2        |
| 115        | Doug         | Hornsey  | P300RTY             | 4        |
| 115        | Doug         | Hornsey  | H259IJK             | 3        |
| 114        | Jay          | Hornsey  | H259IJK             | 3        |
| 114        | Jay          | Hornsey  | P200IJK             | 2        |
| 114        | Jay          | Hornsey  | P300RTY             | 4        |
| 111        | Jean         | Holloway | D678FGH             | 2        |
| 111        | Jean         | Holloway | A123ABC             | 1        |
| 111        | Jean         | Holloway | D345GGG             | 1        |
| 110        | John         | Holloway | D678FGH             | 2        |
| 110        | John         | Holloway | A123ABC             | 1        |
| 110        | John         | Holloway | D345GGG             | 1        |
| 113        | Vince        | Hornsey  | H259IJK             | 3        |
| 113        | Vince        | Hornsey  | P200IJK             | 2        |
| 113        | Vince        | Hornsey  | P300RTY             | 4        |
+------------+--------------+----------+---------------------+----------+
18 rows in set (0.00 sec)
```

Reflection

JOIN matches each row in one table with every row in other tables, and allows to query rows that contain columns from both tables, Using **tno** and **dno** key to link the 4 tables.

Code explanation

Exercise 1.9

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot No', b.regno AS 'Bus Registeration No', bt.tno AS 'Bus Type'	Select the cno, cname, dname, regno, and tno fields from the different tables. **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM Cleaner c	Specify the table Cleaner c, and **c** used to assign the **Cleaner** table alias as **c**.
Join Depot d JOIN Bus b JOIN BusType bt	A clause to select data from the tables by matching rows based on the value in key.
ON c.dno = d.dno ON d.dno = b.dno ON bt.tno = b.tno	Key identifier used for the JOIN clause to join the three tables.
ORDER BY c.cname	Order the result table by the cleaner name alphabetically.

Alternative Solution

```sql
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot No',
b.regno AS 'Bus Registeration No', bt.tno AS 'Bus Type'
FROM Cleaner c, Depot d, BusType bt, Bus b
WHERE c.dno = d.dno AND d.dno = b.dno AND bt.tno = b.tno
ORDER BY c.cname;
```

This solution was also able to get identical results from the first solution, the difference might be visible when dealing with a large amount of data while testing which one can operate faster. Also, the other syntax, I was able to define explicitly by telling MySQL which columns to join and how to join them 'Join'.
This solution is also more difficult to read

Exercise 1.10

(OUTER JOIN) Rewrite question 7 as an OUTER JOIN. Describe the query in English. Now list all cleaners (number and name), the name of their depot and the bus registration numbers with the type of bus that they are responsible for, including those cleaners who are not assigned to a bus or a depot.

Solution

```
/*! 10-OUTER JOIN */
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
d.dname AS 'Depot Name',
b.regno AS 'Bus Registeration No', bt.tdescript AS 'Bus Type'
FROM Cleaner c
LEFT OUTER JOIN Depot d
ON c.dno=d.dno
LEFT OUTER JOIN Bus b
ON d.dno = b.dno
LEFT OUTER JOIN BusType bt
ON bt.tno = b.tno
ORDER BY c.cname;
```

Result

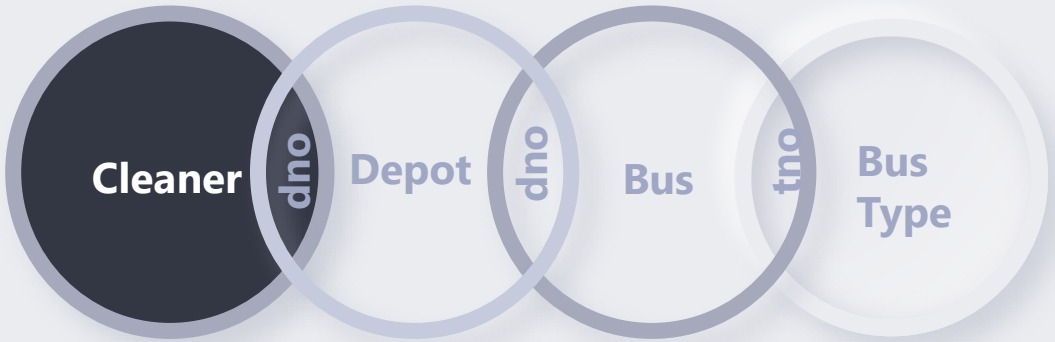
```
mysql> SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
    -> d.dname AS 'Depot Name',
    -> b.regno AS 'Bus Registeration No', bt.tdescript AS 'Bus Type'
    -> FROM Cleaner c
    -> LEFT OUTER JOIN Depot d
    -> ON c.dno=d.dno
    -> LEFT OUTER JOIN Bus b
    -> ON d.dno = b.dno
    -> LEFT OUTER JOIN BusType bt
    -> ON bt.tno = b.tno
    -> ORDER BY c.cname;
+------------+--------------+------------+---------------------+---------------+
| Cleaner No | Cleaner Name | Depot Name | Bus Registeration No | Bus Type      |
+------------+--------------+------------+---------------------+---------------+
| 112        | Betty        | Hornsey    | H259IJK             | midibus       |
| 112        | Betty        | Hornsey    | P200IJK             | metrobus      |
| 112        | Betty        | Hornsey    | P300RTY             | bendy bus     |
| 115        | Doug         | Hornsey    | H259IJK             | midibus       |
| 115        | Doug         | Hornsey    | P200IJK             | metrobus      |
| 115        | Doug         | Hornsey    | P300RTY             | bendy bus     |
| 116        | Geeta        | NULL       | NULL                | NULL          |
| 114        | Jay          | Hornsey    | H259IJK             | midibus       |
| 114        | Jay          | Hornsey    | P200IJK             | metrobus      |
| 114        | Jay          | Hornsey    | P300RTY             | bendy bus     |
| 111        | Jean         | Holloway   | A123ABC             | double-decker |
| 111        | Jean         | Holloway   | D345GGG             | double-decker |
| 111        | Jean         | Holloway   | D678FGH             | metrobus      |
| 110        | John         | Holloway   | A123ABC             | double-decker |
| 110        | John         | Holloway   | D345GGG             | double-decker |
| 110        | John         | Holloway   | D678FGH             | metrobus      |
| 113        | Vince        | Hornsey    | H259IJK             | midibus       |
| 113        | Vince        | Hornsey    | P200IJK             | metrobus      |
| 113        | Vince        | Hornsey    | P300RTY             | bendy bus     |
+------------+--------------+------------+---------------------+---------------+
19 rows in set (0.00 sec)
```


```
+------------+--------------+------------+---------------------+---------------+
| Cleaner No | Cleaner Name | Depot Name | Bus Registeration No | Bus Type      |
+------------+--------------+------------+---------------------+---------------+
| 112        | Betty        | Hornsey    | H259IJK             | midibus       |
| 112        | Betty        | Hornsey    | P200IJK             | metrobus      |
| 112        | Betty        | Hornsey    | P300RTY             | bendy bus     |
| 115        | Doug         | Hornsey    | H259IJK             | midibus       |
| 115        | Doug         | Hornsey    | P200IJK             | metrobus      |
| 115        | Doug         | Hornsey    | P300RTY             | bendy bus     |
| 116        | Geeta        | NULL       | NULL                | NULL          |
| 114        | Jay          | Hornsey    | H259IJK             | midibus       |
| 114        | Jay          | Hornsey    | P200IJK             | metrobus      |
| 114        | Jay          | Hornsey    | P300RTY             | bendy bus     |
| 111        | Jean         | Holloway   | A123ABC             | double-decker |
| 111        | Jean         | Holloway   | D345GGG             | double-decker |
| 111        | Jean         | Holloway   | D678FGH             | metrobus      |
| 110        | John         | Holloway   | A123ABC             | double-decker |
| 110        | John         | Holloway   | D345GGG             | double-decker |
| 110        | John         | Holloway   | D678FGH             | metrobus      |
| 113        | Vince        | Hornsey    | H259IJK             | midibus       |
| 113        | Vince        | Hornsey    | P200IJK             | metrobus      |
| 113        | Vince        | Hornsey    | P300RTY             | bendy bus     |
+------------+--------------+------------+---------------------+---------------+
19 rows in set (0.00 sec)
```

Code explanation

Exercise 1.10

Command	Description
SELECT	is the SQL keyword that lets the database know that you want to retrieve data.
c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name', d.dname AS 'Depot Name', b.regno AS 'Bus Registeration No', bt.tdescript AS 'Bus Type'	Select the cno, cname, dname, regno, and tdescript fields from the different tables. **AS** keyword is optional, but used to give an expression an **alias** to the column.
FROM Cleaner c	Specify the table **BusDriver**, and **bd** used to assign the **BusDriver** table alias as **bd**.
LEFT OUTER JOIN Depot d LEFT OUTER JOIN Bus b LEFT OUTER JOIN BusType bt	A clause to select data from the tables by matching rows based on the value in key. But not only, if the Cleaner row doesn't have any items related in the other tables it still shows the items from the table on the left (cleaner)
ON c.dno = d.dno ON d.dno = b.dno ON bt.tno = b.tno	Key identifier used for the JOIN clause to join the three tables.
ORDER BY c.cname	Order the result table by the cleaner name alphabetically.



Reflection

Using the LEFT **OUTER JOIN** allows to show all **Cleaners** with assigned depot, and bus. At the same time including those cleaners who are not assigned to a bus or a depot.

Whenever we use LEFT, it means to include all the items from the table on the left no matter if they have related items in the joined table or not.
If we were to change the first clause to RIGHT OUTER JOIN **Depot d** then we'll get all the depot items including those that aren't assigned with a cleaner or bus.

```
FROM Cleaner c
RIGHT OUTER JOIN Depot d
ON c.dno=d.dno
```

Cleaner No	Cleaner Name	Depot Name	Bus Registeration No	Bus Type
NULL	NULL	Islington	NULL	NULL
112	Betty	Hornsey	P200IJK	metrobus
112	Betty	Hornsey	P300RTY	bendy bus
112	Betty	Hornsey	H259IJK	midibus
115	Doug	Hornsey	P200IJK	metrobus
115	Doug	Hornsey	P300RTY	bendy bus
115	Doug	Hornsey	H259IJK	midibus
114	Jay	Hornsey	P200IJK	metrobus
114	Jay	Hornsey	P300RTY	bendy bus
114	Jay	Hornsey	H259IJK	midibus
111	Jean	Holloway	D345GGG	double-decker
111	Jean	Holloway	D678FGH	metrobus
111	Jean	Holloway	A123ABC	double-decker
110	John	Holloway	D678FGH	metrobus
110	John	Holloway	A123ABC	double-decker
110	John	Holloway	D345GGG	double-decker
113	Vince	Hornsey	P200IJK	metrobus
113	Vince	Hornsey	P300RTY	bendy bus
113	Vince	Hornsey	H259IJK	midibus

Exercise 2

Exercise 2.1

(Built-in functions) Find the maximum, minimum and average driver's salary.

Solution

```
/*! 1- Built-in functions maximum, minimum and average */
SELECT
  MIN(bdsalary) AS 'Minimum Salary',
  MAX(bdsalary) AS 'Maximum Salary',
  AVG(bdsalary) AS 'Average Salary'
FROM
  BusDriver;
```

Result

```
mysql> SELECT
    ->   MIN(bdsalary) AS 'Minimum Salary',
    ->   MAX(bdsalary) AS 'Maximum Salary',
    ->   AVG(bdsalary) AS 'Average Salary'
    -> FROM
    ->   BusDriver;
+---------------+---------------+---------------+
| Minimum Salary | Maximum Salary | Average Salary |
+---------------+---------------+---------------+
|        1400.00 |        3500.00 |    2021.428571 |
+---------------+---------------+---------------+
1 row in set (0.00 sec)
```

Reflection

The MIN() function returns the minimum value in a set of values.

The MySQL MAX() function returns the maximum value in a set of values.

The AVG function to calculate the average value of the distinct values.

These functions are very handy, as they require only the column name.

Exercise 2.2

(Built-in functions) Count the number of drivers who are working for Middlesex Transport at the moment. Change the column heading in the result to make it 'friendly'.

Solution

```
/*! 2-Built-in functions Count */
SELECT
  COUNT(bdname) AS 'Number of Drivers'
FROM
    BusDriver;
```

Result

```
mysql> SELECT
    ->   COUNT(bdname) AS 'Number of Drivers'
    -> FROM
    ->     BusDriver;
+-------------------+
| Number of Drivers |
+-------------------+
|                 7 |
+-------------------+
1 row in set (0.00 sec)
```

Reflection

The COUNT() function is an aggregate function that returns the number of rows in a table.

Can also helpful when validating the results during data analysis.

Exercise 2.3

(Use a subquery to answer this question) Find route information (route number and description) for all routes which connect to the Holloway Depot.

Solution

```
/*! 3- Subquery */
SELECT rno AS 'Route No', rdescript AS 'Route Description'
FROM Route r
WHERE r.dno IN (SELECT dno
                FROM Depot d
                 WHERE d.dname = 'Holloway'
                )
ORDER BY r.rno;
```

Result

```
mysql> SELECT rno AS 'Route No', rdescript AS 'Route Description'
    -> FROM Route r
    -> WHERE r.dno IN (SELECT dno
    ->                 FROM Depot d
    ->                  WHERE d.dname = 'Holloway'
    ->                 )
    -> ORDER BY r.rno;
+----------+---------------------+
| Route No | Route Description   |
+----------+---------------------+
| 6        | Camden/Golders Green |
| 7        | Finchley/Tottenham  |
| 8        | Hendon/Muswell Hill |
+----------+---------------------+
3 rows in set (0.00 sec)
```

Reflection

Subqueries are queries nested in another query, allowing us to build complex queries to help us retrieve the data we need in a dynamic way. For this example, we were able to filter the data using the id of the depot if we know them. But instead, we used the subquery to find these ids, and can also change the condition easily.

Exercise 2.4

Now try question 3 with a JOIN.

Solution

```sql
/*! 4- Q3 with JOIN */
SELECT rno AS 'Route No', rdescript AS 'Route Description'
FROM Route r
JOIN Depot d
ON r.dno = d.dno
WHERE dname = 'Holloway'
ORDER BY r.rno;
```

Result

```
mysql> SELECT rno AS 'Route No', rdescript AS 'Route Description'
    -> FROM Route r
    -> JOIN Depot d
    -> ON r.dno = d.dno
    -> WHERE dname = 'Holloway'
    -> ORDER BY r.rno;
+----------+----------------------+
| Route No | Route Description    |
+----------+----------------------+
| 6        | Camden/Golders Green |
| 7        | Finchley/Tottenham   |
| 8        | Hendon/Muswell Hill  |
+----------+----------------------+
3 rows in set (0.00 sec)
```

Reflection

I think it's an efficient way to join tables when dealing with data from multiple tables, as we can easily change the query conditions, columns and even tables.

Exercise 2.5

(NULL) List bus details for any bus which has not been assigned to a depot.

Solution

```
/*! 5- NULL */
SELECT regno AS 'Registration No', model AS 'Model'
FROM Bus
WHERE dno IS NULL;
```

Result

```
mysql> SELECT regno AS 'Registration No', model AS 'Model'
    -> FROM Bus
    -> WHERE dno IS NULL;
+-----------------+-----------+
| Registration No | Model     |
+-----------------+-----------+
| R678FDS         | Daf SB220 |
+-----------------+-----------+
1 row in set (0.00 sec)
```

Reflection

A bus without an assigned depot has the value NULL in the dno column.

Was able to find the result with the select query and filter out the condition using the WHERE clause.

Exercise 2.6

(NOT IN) List all drivers (name and number) who are on the system but are not yet responsible for a route.

Solution

```
/*! 6- NOT IN */
SELECT bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name'
FROM BusDriver bd
WHERE bd.bdno
NOT IN (SELECT bdno FROM Ability)
ORDER BY bdno;
```

Result

```
mysql> SELECT bd.bdno AS 'Driver No', bd.bdname AS 'Driver Name'
    -> FROM BusDriver bd
    -> WHERE bd.bdno
    -> NOT IN (SELECT bdno FROM Ability)
    -> ORDER BY bdno;
+-----------+-------------+
| Driver No | Driver Name |
+-----------+-------------+
| 006       | Sally Smith |
| 010       | Peter Piper |
| 011       | John Peel   |
+-----------+-------------+
3 rows in set (0.00 sec)
```

Reflection

The drivers assigned responsibility are registered in the Ability table, and uses the bdno column as a key.

The WHERE clause was able to select the drivers that didn't have a record in the Ability column.

Can also be done using JOIN tables.

Exercise 2.7

(GROUP BY) List each depot name and the average salary for drivers working at the depot.

Solution

```sql
/*! 7- GROUP BY */
SELECT d.dname AS 'Depot Name', AVG(bd.bdsalary) AS 'Average Salary'
FROM Depot d
JOIN BusDriver bd
ON d.dno = bd.dno
GROUP BY d.dno;
```

Result

```
mysql> SELECT d.dname AS 'Depot Name', AVG(bd.bdsalary) AS 'Average Salary'
    -> FROM Depot d
    -> JOIN BusDriver bd
    -> ON d.dno = bd.dno
    -> GROUP BY d.dno;
+------------+----------------+
| Depot Name | Average Salary |
+------------+----------------+
| Holloway   |    1600.000000 |
| Hornsey    |    1900.000000 |
| Islington  |    3500.000000 |
+------------+----------------+
3 rows in set (0.01 sec)
```

Reflection

Used **GROUP BY** to group rows into subgroups based on values of columns or expressions.

This has returned one row for each item in **dno** if wasn't used might have 1 or many items from the query.

Exercise 2.8

(GROUP BY HAVING) List each depot by name and count the number of bus drivers who are assigned to each, for depots with more than one driver.

Solution

```
/*! 8- GROUP BY HAVING */
SELECT dname AS 'Depot Name', COUNT(bdname) AS 'Number of Bus Drivers'
FROM Depot d
JOIN BusDriver bd
ON d.dno = bd.dno
GROUP BY d.dno
HAVING COUNT(bdname) > 1;
```

Result

```
mysql> SELECT dname AS 'Depot Name', COUNT(bdname) AS 'Number of Bus Drivers'
    -> FROM Depot d
    -> JOIN BusDriver bd
    -> ON d.dno = bd.dno
    -> GROUP BY d.dno
    -> HAVING COUNT(bdname) > 1;
+------------+----------------------+
| Depot Name | Number of Bus Drivers |
+------------+----------------------+
| Holloway   |                    2 |
| Hornsey    |                    3 |
+------------+----------------------+
2 rows in set (0.00 sec)
```

Reflection

Used **GROUP BY HAVING** Used as an alternative to **WHERE** keyword, it helps to filter the items using a condition. In our case the depot count number bigger than 1. So the bus driver is assigned to more than 1 depot.

Exercise 2.9

(GROUP BY plus JOIN) For each cleaner responsible for buses of bus type doubledecker or minibus, list his/her name and number and find the total number for which each cleaner is responsible.

Solution

```
/*! 9- GROUP BY plus JOIN */
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
COUNT(b.regno) AS 'Responsible of doubledecker or minibus'

FROM Cleaner c
Join Depot d
ON c.dno = d.dno
JOIN Bus b
ON d.dno = b.dno
JOIN BusType bt
ON bt.tno = b.tno
WHERE bt.tno = 1 OR b.tno = 3
GROUP By c.cno;
```

Result

```
mysql> SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
    -> COUNT(b.regno) AS 'Responsible of doubledecker or minibus'
    ->
    -> FROM Cleaner c
    -> Join Depot d
    -> ON c.dno = d.dno
    -> JOIN Bus b
    -> ON d.dno = b.dno
    -> JOIN BusType bt
    -> ON bt.tno = b.tno
    -> WHERE bt.tno = 1 OR b.tno = 3
    -> GROUP By c.cno;
+------------+--------------+----------------------------------------+
| Cleaner No | Cleaner Name | Responsible of doubledecker or minibus |
+------------+--------------+----------------------------------------+
| 110        | John         |                                      2 |
| 111        | Jean         |                                      2 |
| 112        | Betty        |                                      1 |
| 113        | Vince        |                                      1 |
| 114        | Jay          |                                      1 |
| 115        | Doug         |                                      1 |
+------------+--------------+----------------------------------------+
```

Reflection

Used **GROUP BY** and **JOIN** To retrieve the data from multiple tables and list all of the items from a table, in our case show **each cleaner.**

Exercise 2.10

(GROUP BY plus JOIN) For each cleaner responsible for buses of bus type doubledecker or minibus, list his/her name and number and find the total number for which each cleaner is responsible.

Solution

```
/*! 9- GROUP BY plus JOIN */
SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
COUNT(b.regno) AS 'Responsible of doubledecker or minibus'

FROM Cleaner c
Join Depot d
ON c.dno = d.dno
JOIN Bus b
ON d.dno = b.dno
JOIN BusType bt
ON bt.tno = b.tno
WHERE bt.tno = 1 OR b.tno = 3
GROUP By c.cno;
```

Result

```
mysql> SELECT c.cno AS 'Cleaner No', c.cname AS 'Cleaner Name',
    -> COUNT(b.regno) AS 'Responsible of doubledecker or minibus'
    ->
    -> FROM Cleaner c
    -> Join Depot d
    -> ON c.dno = d.dno
    -> JOIN Bus b
    -> ON d.dno = b.dno
    -> JOIN BusType bt
    -> ON bt.tno = b.tno
    -> WHERE bt.tno = 1 OR b.tno = 3
    -> GROUP By c.cno;
+------------+--------------+----------------------------------------+
| Cleaner No | Cleaner Name | Responsible of doubledecker or minibus |
+------------+--------------+----------------------------------------+
| 110        | John         |                                      2 |
| 111        | Jean         |                                      2 |
| 112        | Betty        |                                      1 |
| 113        | Vince        |                                      1 |
| 114        | Jay          |                                      1 |
| 115        | Doug         |                                      1 |
+------------+--------------+----------------------------------------+
```

Reflection

Used **GROUP BY** and **JOIN** To retrieve the data from multiple tables and list all of the items from a table, in our case show **each cleaner** and the related items from different tables**.**

Also used the bus type number **tno** to filter out only the doubledecker or minibus bus type.

Thank you!