CITS1402 Week 7 Lab

Gordon Royle

2022 Semester Two

Learning Aims

This is a "consolidation lab" intended to give more practice with the fundamental aspects of SQL, consolidating your knowledge of how SQL (and SQLite) works. Most SQL queries involve both aggregate functions and subqueries so this lab will focus on such queries.

We will use the databases `AustOpen` and `classicmodels`. The SQLite `.db` files for these databases can be downloaded from the Weekly Lab Sheets area.

You should try to ensure that your queries *do not use unnecessary tables* as this could be a major source of inefficiency in a larger database. In the forthcoming test and final examination, the markers will first be checking to ensure that each query uses only the tables that it *must use*, and then consider whether the query does the right thing.

Questions

1. The "Big Four" male tennis players are Roger Federer, Rafael Nadal, Novak Djokovic and Andy Murray. Write a single SQL query, using the keyword `IN`, that lists the *year* and the names of both the *winner* and the *loser* for every match where one of the Big Four *lost*.

```
SELECT yr, winnerName, loserName
FROM ATPResult
WHERE losername IN ('Andy Murray', 'Roger Federer', 'Novak Djokovic', 'Rafael Nadal');
```

   Self Check Hint: In both 2004 and 2005, Lleyton Hewitt beat Rafael Nadal.

2. Write an SQL query that lists that *name* of every player that has beaten one of the Big Four, along with the *number of times* that this has happened. One player stands out as having accomplished this more feat far more often than any other player - who is it? (You can answer this by looking at the output for the query, you do not need to write any more SQL.)

```
SELECT winnerName, COUNT(*)
FROM ATPResult
WHERE losername IN ('Andy Murray', 'Roger Federer', 'Novak Djokovic', 'Rafael Nadal')
GROUP BY winnerName;
```

   Self Check Hint: Lleyton Hewitt has twice beaten one of the Big Four.

3. Which tennis player has been the *victim* of one of the Big Four most often? Write an SQL query that lists, for each player who has lost at least once against the Big Four, the player's *name* and

the total *number of times* that they have *lost* to the one of Big Four, sorted in decreasing order of number of losses

```
SELECT loserName, COUNT(*)
FROM ATPResult
WHERE winnerName IN ('Andy Murray', 'Roger Federer', 'Novak Djokovic', 'Rafael Nadal')
GROUP BY loserName
ORDER BY COUNT(*) DESC;
```

Self Check Hint: Roger Federer is second on this list, having lost 8 times to (other) players in the Big Four.

4. In the Australian Open, each player is viewed as representing their *home country*. The fields `winnerCountry` and `loserCountry` contain the 3-letter country codes of the winners and the losers respectively. Write a single SQL query that lists, for each country, the *country code* and the *total number* of matches won by players representing that country.

   Use the results for the Women's Singles that are in the table `WTAResult`.

   Self Check Hint: French women (with code `FRA`) have won 167 matches in total, taking France to 4th on the list of "most-winning countries". So the fourth row of the output table should be: `FRA 167`

```
SELECT winnerCountry, COUNT(*)
FROM WTAResult
GROUP BY winnerCountry
ORDER BY COUNT(*) DESC;
```

5. Write an SQL query that lists the *customer numbers* of all the customers who have *made a payment* to Classic Models. Each customer number should be listed only once, regardless of how many payments have been made. The table `payments` contains information about payments.

```
SELECT DISTINCT customerNumber
FROM payments;
```

6. Write a single SQL query that lists the *minimum*, *maximum* and *average* payment amount over all the payments that have been made.

```
SELECT MIN(amount), AVG(amount), MAX(amount) FROM payments;
```

   Self Check Hint The output for this should be a table with one row and three columns.

7. Write an SQL query that lists, for each customer who has made at least one payment, their *customer number*, the *number of payments* and the *total amount* they have paid.

   Self Check Hint Customer 379 has made three payments, totalling $73533.65.

```
SELECT customerNumber, COUNT(*), SUM(amount)
FROM payments
GROUP BY customerNumber;
```

8. Write an SQL query that lists, for each customer who has made at least one payment, their *name*, *number of payments*, and *total amount paid.*

   For example, "Collectables For Less Inc." have made 3 payments totalling $73533.65.

```
SELECT C.customerName, COUNT(*), SUM(amount)
FROM customers C, payments P
WHERE C.customerNumber = P.customerNumber
GROUP BY C.customerNumber;
```

9. Write a single SQL query that lists the names of the customers (once each) who have made any payments for an amount larger than the average payment amount.

```
SELECT DISTINCT C.customerName
FROM customers C JOIN payments P USING (customerNumber)
WHERE P.amount > (SELECT AVG(amount) FROM payments);
```

10. Write an SQL query that lists *the names* of all of the customers that have not made any orders. For example, "Messner Shopping Network" is a customer that has made no orders.

```
SELECT customerName
FROM customers C
WHERE C.customerNumber
  NOT IN (SELECT customerNumber FROM orders);
```

11. Write an SQL query that lists the order numbers (only) for all of the orders with *exactly one* line item.

```
SELECT ordernumber
FROM orderdetails
GROUP BY ordernumber
HAVING COUNT(*) = 1;
```

SELF CHECK HINT There are 17 order numbers that only had one line item.

12. Write an SQL query that lists the *order numbers* of all orders that contain planes.

```
SELECT DISTINCT orderNumber
FROM orderdetails JOIN products USING (productCode)
WHERE productLine = 'Planes';
```

SELF CHECK HINT There are 66 orders that contain planes.

13. Write an SQL query that lists the *order numbers* of all orders that contain both planes and trains.

```
SELECT DISTINCT orderNumber
FROM orderdetails JOIN products USING (productCode)
WHERE productLine = 'Planes'
INTERSECT
SELECT DISTINCT orderNumber
FROM orderdetails JOIN products USING (productCode)
WHERE productLine = 'Trains';
```

```
SELECT DISTINCT D.orderNumber
FROM orderdetails D JOIN products P USING (productCode)
WHERE productLine = 'Planes'
AND EXISTS (SELECT *
    FROM orderDetails D2 JOIN Products P2
    USING (productCode)
    WHERE D.orderNumber = D2.orderNumber
    AND P2.productLine = 'Trains');
```

14. Without using a correlated subquery, write an SQL query that lists the *order numbers* of all orders that *do not* contain any planes.

```
SELECT ordernumber
FROM orders
```

```
WHERE ordernumber NOT IN
  (SELECT DISTINCT orderNumber
   FROM orderdetails, products
   WHERE orderdetails.productCode = products.productCode
   AND productLine = 'Planes');
```

15. Repeat the previous question, but this time make sure that your SQL query *does* use a correlated subquery.

```
SELECT O.ordernumber FROM orders O WHERE
NOT EXISTS
  (SELECT * FROM orderdetails D, products P
   WHERE D.productCode = P.productCode
   AND P.productLine = 'Planes'
   AND O.ordernumber = D.ordernumber);
```

16. Write an SQL query that lists the *order numbers* of all orders that *only contain* planes.

```
SELECT O.ordernumber FROM orders O WHERE
NOT EXISTS
  (SELECT * FROM orderdetails D, products P
   WHERE D.productCode = P.productCode
   AND P.productLine <> 'Planes'
   AND O.ordernumber = D.ordernumber);
```