FDT Worksheet Week 4
Splitting up queries

Subqueries

1. Make a simple SELECT statement, then wrap it in a subquery and select everything from it.

```
SELECT * FROM film
```

Then:

```
SELECT * FROM
(
SELECT * FROM film
) AS t
```

2. Make a simple SELECT statement, then wrap it in a subquery and select just a few columns.

```
SELECT * FROM film
```

Then:

```
SELECT film_id, title
FROM
( SELECT * FROM film) AS t
```

3. Join the film and inventory tables together, wrap the result in a subquery, and select all the inventory columns.

```
SELECT * FROM
film INNER JOIN inventory
ON film.film_id = inventory.film_id
```

Then:

```
SELECT * FROM(
SELECT inventory.* FROM
film INNER JOIN inventory
ON film.film_id = inventory.film_id
        ) AS t
```

4. Show the first half of the products table

```
SELECT * FROM
(SELECT row_number() OVER (ORDER BY ProductID) as n, *
FROM products)
AS t
WHERE n < (SELECT COUNT(*)
FROM products)/2
```

5. Why is this a bad solution to the previous question:

```
SELECT * FROM Products
WHERE ProductID < (SELECT COUNT(*) FROM products)/2
```

Answer: because there may be missing or non-consecutive IDs. If n = 200 and the IDs start at 190, WHERE ProductID > 100 will not give the right result. *The ID is not the same as the row number.*

CTEs (WITH)

6. Set up a simple query, then put it into a CTE and select everything from it.

```
WITH t AS (SELECT * FROM film)
SELECT * FROM t
```

7. Set up a simple query, then put it into a CTE and select some columns from it.

```
WITH t AS (SELECT * FROM film)
SELECT film_id, title FROM t
```

8. In a CTE, select all the films with rating G. Then CROSS JOIN this CTE to itself. Note the number of rows.

```
WITH g_films AS (SELECT * FROM film WHERE rating = 'G')
SELECT * FROM
g_films CROSS JOIN g_films AS g_films_2
```

We need to give one table an alias, as Postgres won't let us join two tables with the same name.
The number of rows is the square of the number of films with rating G.

Views

9. Take a complex query and save it as a view. Then select from the view.

```
CREATE VIEW customers_by_rental_count AS(
	SELECT first_name, last_name, COUNT(rental.rental_id) as rental_count
	FROM
	customer INNER JOIN rental
	ON customer.customer_id = rental.customer_id GROUP BY first_name, last_name
	ORDER BY rental_count DESC
)

SELECT * FROM customers_by_rental_count
```

10. Modify the previous view to run a slightly different query. Then select from the view again.

```
CREATE OR REPLACE VIEW customers_by_rental_count AS(
	SELECT first_name, last_name, COUNT(rental.rental_id) as rental_count
	FROM
	customer INNER JOIN rental
	ON customer.customer_id = rental.customer_id GROUP BY first_name, last_name
	ORDER BY rental_count ASC
)

SELECT * FROM customers_by_rental_count
```

11. Delete the previous view.

```
DROP VIEW customers_by_rental_count
```