1.1


```
WITH top_5_customers (customer_id, first_name, last_name, country, city, total_amount_paid) AS
        (SELECT A.customer_id, A.first_name AS "First name", A.last_name AS "Last name", D.country,
C.city,
        SUM (E.amount) AS Total_Amount_Paid
        FROM customer A
                INNER JOIN address B ON A.address_id = B.address_id
                INNER JOIN city C ON B.city_id = C.city_id
                INNER JOIN country D ON C.country_id = D.country_id
                INNER JOIN payment E ON A.customer_id = E.customer_id
        WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary',
                'Tianji', 'Cianjur', 'So Leopoldo')
        GROUP BY country, city, "First name", "Last name", A.customer_id
        ORDER BY total_amount_paid DESC
        LIMIT 5)
SELECT AVG (total_amount_paid) AS average
FROM top_5_customers
```

1.1 Went back and cleaned up my original query. Then placed the WITH in the front, using the lesson exercise as an example. I chose to keep the renaming “First name” and “Last name” because it adds to readability and comprehension, while attempting to change ‘top_5_customers’ broke it. I chose to leave it alone rather than try to figure it out.

```
WITH top_5_customers (customer_id, first_name, last_name, country, city, total_amount_paid) AS
	(SELECT A.customer_id, A.first_name AS "First name", A.last_name AS "Last name", D.country,
C.city,
	SUM (E.amount) AS "Total paid"
	FROM customer A
		INNER JOIN address B ON A.address_id = B.address_id
		INNER JOIN city C ON B.city_id = C.city_id
		INNER JOIN country D ON C.country_id = D.country_id
		INNER JOIN payment E ON A.customer_id = E.customer_id
	WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary',
		'Tianji', 'Cianjur', 'So Leopoldo')
	GROUP BY country, city, "First name", "Last name", A.customer_id
	ORDER BY "Total paid" DESC
	LIMIT 5)
SELECT D.country AS "Country", COUNT(A.customer_id) AS "Total customers",
COUNT(top_5_customers) AS "Top customers"
	FROM customer A
		INNER JOIN address B on A. address_id=B. address_id
		INNER JOIN city C on B. city_id=C. city_id
		INNER JOIN country D on C. country_id=D. Country_id
	LEFT JOIN top_5_customers
	ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
HAVING COUNT (top_5_customers) > 0
ORDER BY COUNT(top_5_customers), COUNT(A.customer_id) DESC
```

```sql
WITH top_5_customers (customer_id, first_name, last_name, country, city, total_amount_paid) AS
    (SELECT A.customer_id, A.first_name AS "First name", A.last_name AS "Last name", D.country, C.city,
    SUM (E.amount) AS "Total paid"
    FROM customer A
        INNER JOIN address B ON A.address_id = B.address_id
        INNER JOIN city C ON B.city_id = C.city_id
        INNER JOIN country D ON C.country_id = D.country_id
        INNER JOIN payment E ON A.customer_id = E.customer_id
    WHERE city IN ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambattur', 'Shanwei', 'Teboksary',
        'Tianji', 'Cianjur', 'So Leopoldo')
    GROUP BY country, city, "First name", "Last name", A.customer_id
    ORDER BY "Total paid" DESC
    LIMIT 5)
SELECT D.country AS "Country", COUNT(A.customer_id) AS "Total customers",
COUNT(top_5_customers) AS "Top customers"
    FROM customer A
        INNER JOIN address B on A. address_id=B. address_id
        INNER JOIN city C on B. city_id=C. city_id
        INNER JOIN country D on C. country_id=D. Country_id
    LEFT JOIN top_5_customers
    ON A.customer_id = top_5_customers.customer_id
GROUP BY D.country
HAVING COUNT (top_5_customers) > 0
ORDER BY COUNT(top_5_customers), COUNT(A.customer_id) DESC
```

1.2: After looking at my last assignment, I chose to start over and ***only*** rename the items I needed to, so that the query would be cleaner and I could follow along better. “First name” “Last name” “Total Paid” and “Country” which were listed in the results of the query. Again, This is for readability for the end user.

E. Christian Cotterman
Answers Ex 3.9

2: Depending on the end needs of the query, I think the CTE from the first example would perform better. I think this is also discussed in the event of continuously updated databases. The second example seems more verbose, but seeing as it works, I honestly can't say if it would be better than the subquery from the last exercise. Again, it would depend on the size and complexity of the database.

The milliseconds for such a small query and database are the same or so microscopic as to be irrelevant. When DB's become larger and more complex it will make a difference.

3: I was challenged simply by being new to SQL. The Order of Operations that exist are very helpful, but as with all things in this course, it is immersion and using daily that will help me meet the challenges of becoming faster and more confident in writing queries, etc.
I am slightly intimidated by the use of subqueries and CTE's but as I just said, I think this will fade with consistent usage. The main issue I have is with my wanting to rename tables/columns to fit my thought process; I will need to be more disciplined with this to avoid other time-wasting issues that can be avoided.