3.9

Step One:

1.

```
WITH total_amount_paid_cte (customer_id, first_name, last_name, country, city, total_paid) AS
(SELECT A.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city,
        SUM(A.amount) AS total_amount_paid
FROM payment A
    INNER JOIN customer B ON A.customer_id = B.customer_id
    INNER JOIN address C ON B.address_id = C.address_id
    INNER JOIN city D ON C.city_id = D.city_id
    INNER JOIN country E ON D.country_id = E.country_id
WHERE country IN ('India', 'China', 'United States', 'Japan', 'Mexica', 'Brazil',
             'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
AND city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulla)', 'Kurashiki',
            'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY A.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city
ORDER BY total_amount_paid DESC LIMIT 5)
SELECT AVG(total_paid)
From total_amount_paid_cte
```

Data Output | Explain | Messages | Notifications

	avg numeric
1	97.772

2.

```sql
WITH top_5_cte (customer_id, first_name, last_name, country, city, total_paid) AS
(SELECT A.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city,
        SUM(A.amount) AS total_amount_paid
FROM payment A
    INNER JOIN customer B ON A.customer_id = B.customer_id
    INNER JOIN address C ON B.address_id = C.address_id
    INNER JOIN city D ON C.city_id = D.city_id
    INNER JOIN country E ON D.country_id = E.country_id
WHERE country IN ('India', 'China', 'United States', 'Japan', 'Mexica', 'Brazil',
             'Russian Federation', 'Philippines', 'Turkey', 'Indonesia')
AND city IN ('Aurora', 'Atlixco', 'Xintai', 'Adoni', 'Dhule (Dhulla)', 'Kurashiki',
            'Pingxiang', 'Sivas', 'Celaya', 'So Leopoldo')
GROUP BY A.customer_id,
        B.first_name,
        B.last_name,
        E.country,
        D.city
ORDER BY total_amount_paid DESC LIMIT 5)
SELECT country.country,
    COUNT (DISTINCT customer.customer_id) AS all_customer_count,
    COUNT (DISTINCT country.country) AS top_customer_count
From top_5_cte
LEFT JOIN customer ON customer.customer_id = customer.customer_id
LEFT JOIN address ON customer.address_id = address.address_id
LEFT JOIN city ON address.city_id = city.city_id
LEFT JOIN country ON city.country_id = country.country_id
GROUP BY country.country
ORDER BY all_customer_count DESC LIMIT 5
```

Data Output | Explain | Messages | Notifications

	country character varying (50)	all_customer_count bigint	top_customer_count bigint
1	India	60	1
2	China	53	1
3	United States	36	1
4	Japan	31	1
5	Mexico	30	1

3. I started the CTE with the ‘WITH’ clause and named it . Then listed the columns that will be in the CTE definition using the ‘AS’ clause. I copied the query from the previous exercise starting with the ‘SELECT’ clause to get the answer.

Step two:

4.
CTE step one

Query Editor | Query History | Scratch Pad

```
1  EXPLAIN WITH total_amount_paid_cte (customer_id, first_name, last_name, coun
2  (SELECT A.customer_id,
3          B.first_name,
4          B.last_name,
5          E.country,
6          D.city,
7          SUM(A.amount) AS total_amount_paid
```

```
CREATE TABL
(
 category_id ir
 name text CO
 last_update ti
 CONSTRAINT
);
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=29.22..29.23 rows=1 width=32)
2	[...] -> Limit (cost=29.15..29.16 rows=5 width=65)
3	[...] -> Sort (cost=29.15..29.20 rows=22 width=65)
4	[...] Sort Key: (sum(a.amount)) DESC
5	[...] -> HashAggregate (cost=28.51..28.78 rows=22 width=65)
6	[...] Group Key: a.customer_id, b.first_name, b.last_name, e.country, d.city
7	[...] -> Nested Loop (cost=3.62..28.18 rows=22 width=39)
8	[...] -> Nested Loop (cost=3.33..26.40 rows=1 width=35)
9	[...] -> Nested Loop (cost=3.06..26.01 rows=1 width=22)
10	[...] -> Hash Join (cost=2.79..21.31 rows=1 width=22)
11	[...] Hash Cond: (d.country_id = e.country_id)
12	[...] -> Seq Scan on city d (cost=0.00..18.50 rows=10 width=15)
13	[...] Filter: ((city)::text = ANY ('{Aurora,Atlixco,Xintai,Adoni,"Dhule (Dhulla)",Kurashiki,Pingxiang,Sivas,Celaya,"So Leopoldo"}'::text[]))

Read-only column

CTE step 2

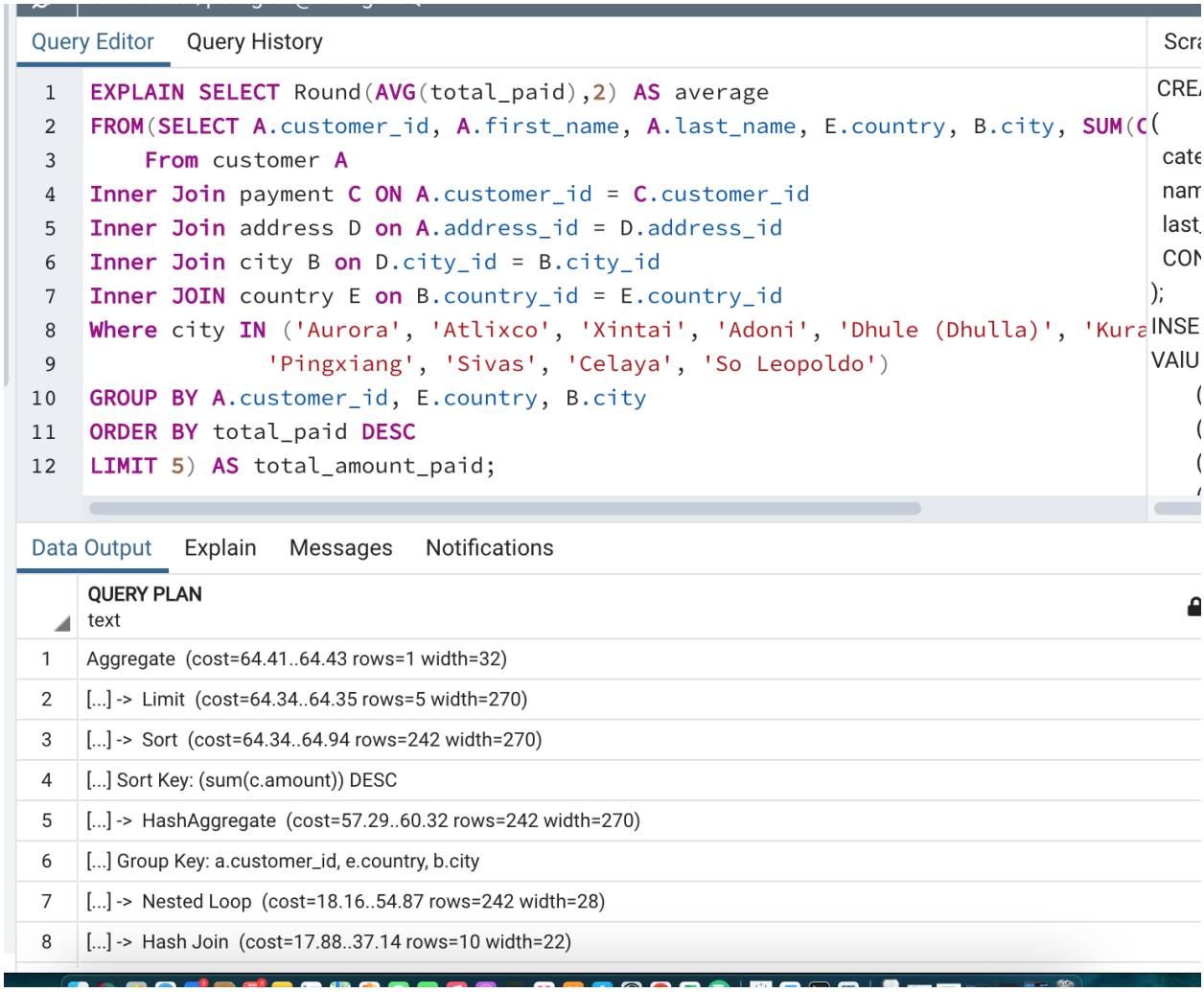
Query Editor | Query History

```
1  EXPLAIN WITH top_5_cte (customer_id, first_name, last_name, country, city, total_paid) AS
2  (SELECT A.customer_id,
3          B.first_name,
4          B.last_name,
5          E.country,
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Limit (cost=337.24..337.25 rows=5 width=25)
2	[...] -> Sort (cost=337.24..337.51 rows=109 width=25)
3	[...] Sort Key: (count(DISTINCT customer.customer_id)) DESC
4	[...] -> GroupAggregate (cost=304.38..335.42 rows=109 width=25)
5	[...] Group Key: country.country
6	[...] -> Sort (cost=304.38..311.87 rows=2995 width=13)
7	[...] Sort Key: country.country
8	[...] -> Nested Loop Left Join (cost=72.67..131.45 rows=2995 width=13)
9	[...] -> Limit (cost=29.15..29.16 rows=5 width=65)
10	[...] -> Sort (cost=29.15..29.20 rows=22 width=65)
11	[...] Sort Key: (sum(a.amount)) DESC
12	[...] -> HashAggregate (cost=28.51..28.78 rows=22 width=65)
13	[...] Group Key: a.customer_id, b.first_name, b.last_name, e.country, d.city
14	[...] -> Nested Loop (cost=3.62..28.18 rows=22 width=39)
15	[...] -> Nested Loop (cost=3.33..26.40 rows=1 width=35)

Subquery step one



Subquery step two

Query Editor | Query History

```
1  EXPLAIN SELECT DISTINCT(A.country),
2      COUNT(DISTINCT D.customer_id) AS all_customer_count,
3      COUNT(DISTINCT A.country) AS top_customer_count
4  FROM country A
5  Inner JOIN city B
6      ON A.country_id = B.country_id
7  INNER JOIN address c
8
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Limit (cost=154.14..154.16 rows=5 width=84)
2	[...] -> Sort (cost=154.14..155.51 rows=545 width=84)
3	[...] Sort Key: (count(DISTINCT d.customer_id)) DESC
4	[...] -> HashAggregate (cost=139.64..145.09 rows=545 width=84)
5	[...] Group Key: count(DISTINCT d.customer_id), a.country, count(DISTINCT a.country)
6	[...] -> GroupAggregate (cost=122.62..135.55 rows=545 width=84)
7	[...] Group Key: a.country, top_5_customers.*
8	[...] -> Sort (cost=122.62..124.11 rows=599 width=72)
9	[...] Sort Key: a.country, top_5_customers.*
10	[...] -> Hash Left Join (cost=72.68..94.98 rows=599 width=72)
11	[...] Hash Cond: ((a.country)::text = (top_5_customers.country)::text)
12	[...] -> Hash Join (cost=43.52..63.30 rows=599 width=13)

I think using the CTE is best due to the fact that if you need to reuse the query, you can just state the name you named it. There's less coding with using a CTE as well. I rewrote my subqueries so the cost results I got may not reflect which one costs more or less. I believe they should be the same cost though.

Step three:

I kept getting errors in syntax so I had to write some of the subquery to make it work. Defining the INNER JOIN and INNER LEFT are a little confusing so that took some time to figure out. Also, figuring out which columns were needed and where to list them was hard.