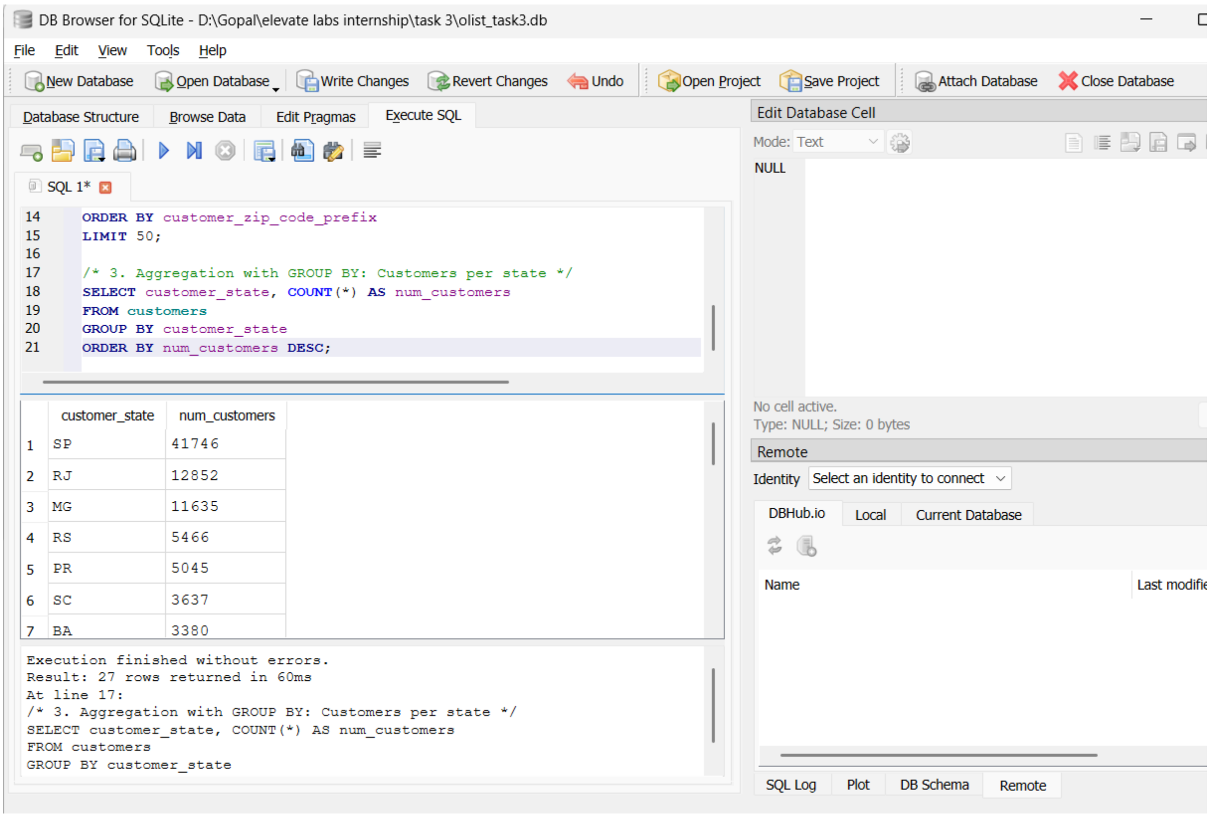
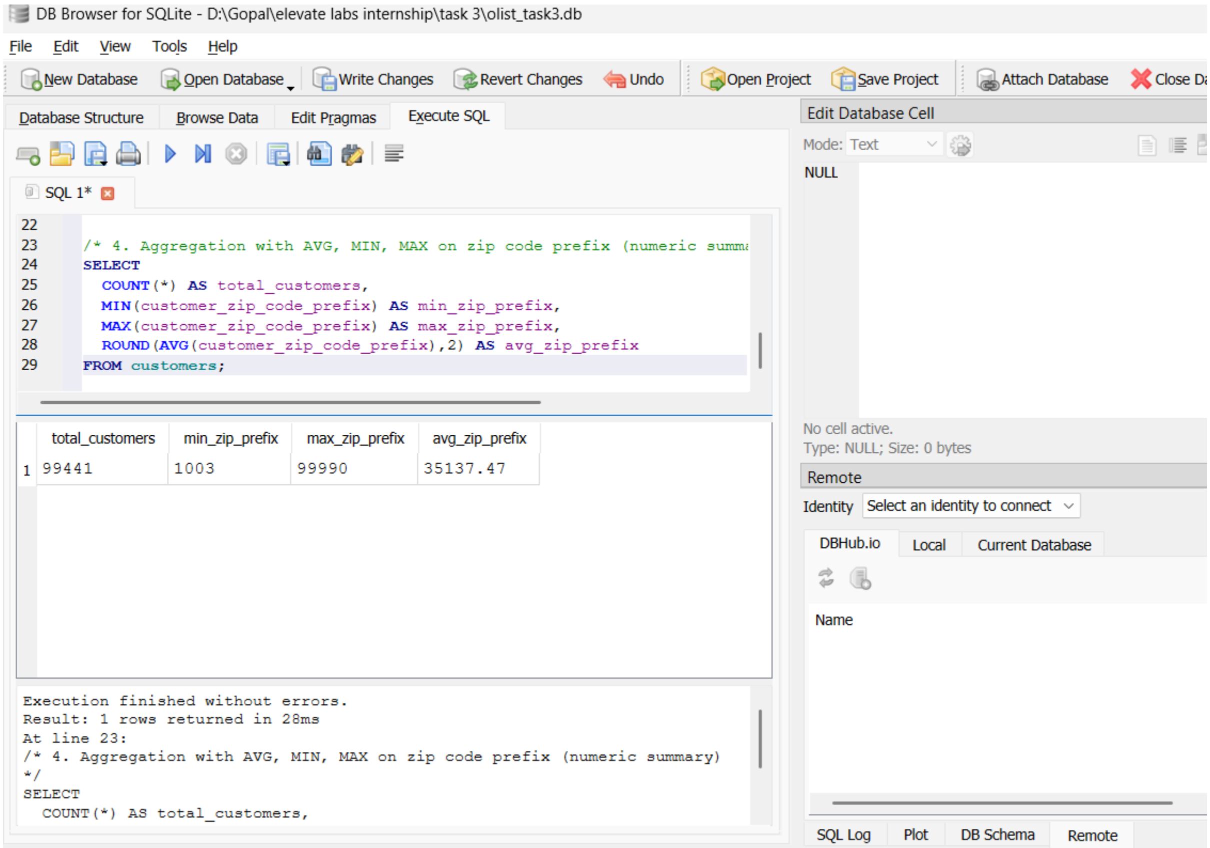


```
ORDER BY customer_zip_code_prefix
LIMIT 50;

/* 3. Aggregation with GROUP BY: Customers per state */
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
ORDER BY num_customers DESC;
```

	customer_state	num_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

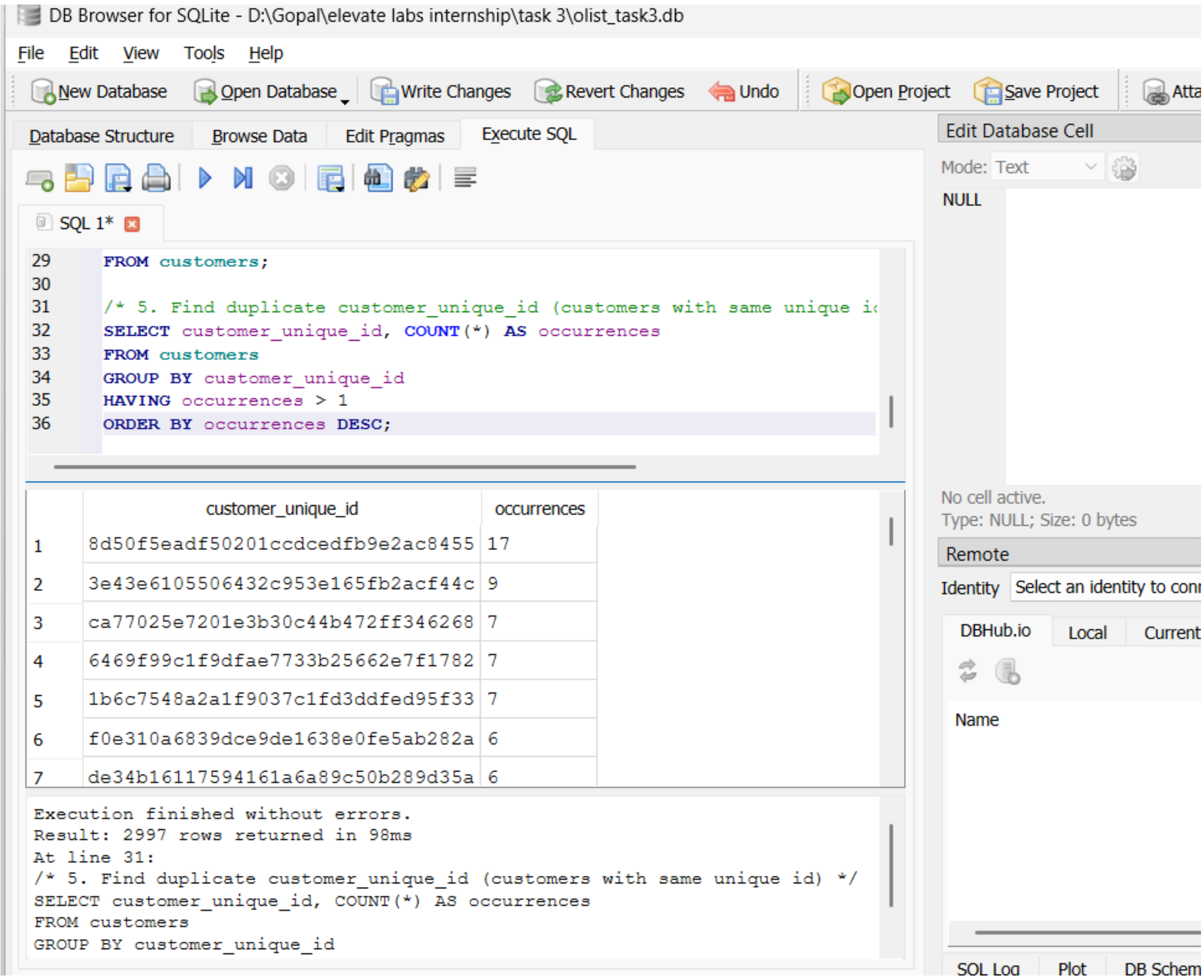
```
Execution finished without errors.
Result: 27 rows returned in 60ms
At line 17:
/* 3. Aggregation with GROUP BY: Customers per state */
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
```



```

/* 4. Aggregation with AVG, MIN, MAX on zip code prefix (numeric summa
SELECT
   COUNT(*) AS total_customers,
   MIN(customer_zip_code_prefix) AS min_zip_prefix,
   MAX(customer_zip_code_prefix) AS max_zip_prefix,
   ROUND(AVG(customer_zip_code_prefix),2) AS avg_zip_prefix
FROM customers;
```

	total_customers	min_zip_prefix	max_zip_prefix	avg_zip_prefix
1	99441	1003	99990	35137.47

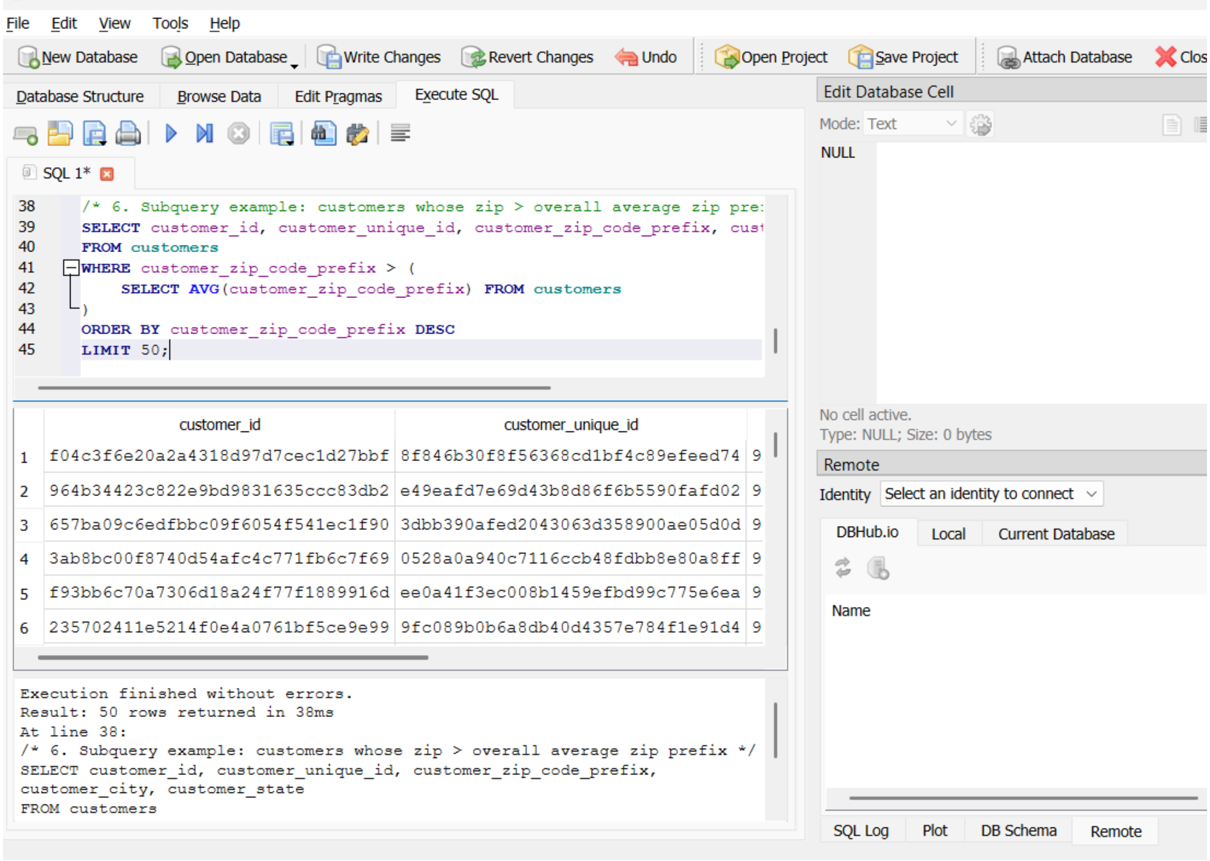
```
Execution finished without errors.
Result: 1 rows returned in 28ms
At line 23:
/* 4. Aggregation with AVG, MIN, MAX on zip code prefix (numeric summary)
*/
SELECT
  COUNT(*) AS total_customers,
```



```
FROM customers;

/* 5. Find duplicate customer_unique_id (customers with same unique i
SELECT customer_unique_id, COUNT(*) AS occurrences
FROM customers
GROUP BY customer_unique_id
HAVING occurrences > 1
ORDER BY occurrences DESC;
```

	customer_unique_id	occurrences
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff346268	7
4	6469f99c1f9dfae7733b25662e7f1782	7
5	1b6c7548a2a1f9037c1fd3ddfed95f33	7
6	f0e310a6839dce9de1638e0fe5ab282a	6
7	de34b16117594161a6a89c50b289d35a	6

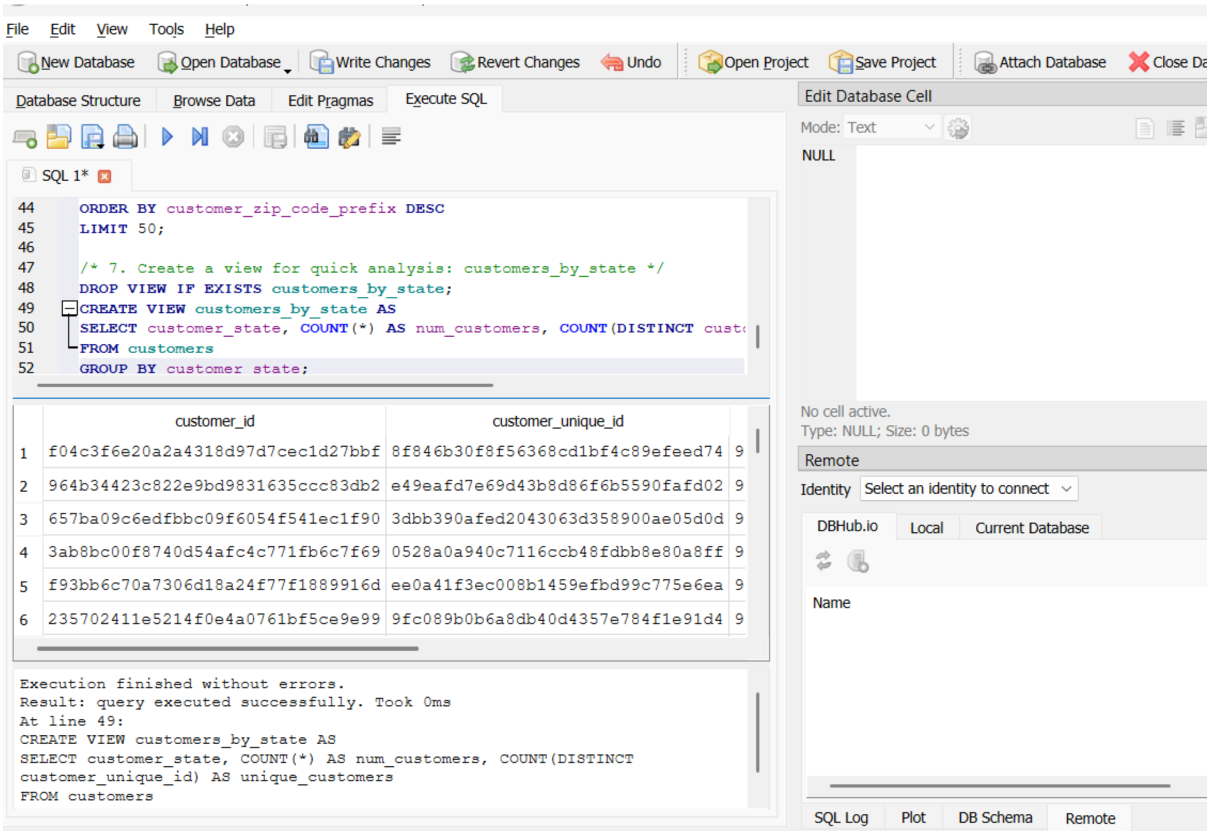
```
Execution finished without errors.
Result: 2997 rows returned in 98ms
At line 31:
/* 5. Find duplicate customer_unique_id (customers with same unique id) */
SELECT customer_unique_id, COUNT(*) AS occurrences
FROM customers
GROUP BY customer_unique_id
```



```
/* 6. Subquery example: customers whose zip > overall average zip pre:
SELECT customer_id, customer_unique_id, customer_zip_code_prefix, cust
FROM customers
WHERE customer_zip_code_prefix > (
    SELECT AVG(customer_zip_code_prefix) FROM customers
)
ORDER BY customer_zip_code_prefix DESC
LIMIT 50;
```

	customer_id	customer_unique_id	
1	f04c3f6e20a2a4318d97d7cec1d27bbf	8f846b30f8f56368cd1bf4c89efeed74	9
2	964b34423c822e9bd9831635ccc83db2	e49eafd7e69d43b8d86f6b5590fafd02	9
3	657ba09c6edfbbc09f6054f541ec1f90	3dbb390afed2043063d358900ae05d0d	9
4	3ab8bc00f8740d54afc4c771fb6c7f69	0528a0a940c7116ccb48fdbb8e80a8ff	9
5	f93bb6c70a7306d18a24f77f1889916d	ee0a41f3ec008b1459efbd99c775e6ea	9
6	235702411e5214f0e4a0761bf5ce9e99	9fc089b0b6a8db40d4357e784f1e91d4	9

```
Execution finished without errors.
Result: 50 rows returned in 38ms
At line 38:
/* 6. Subquery example: customers whose zip > overall average zip prefix */
SELECT customer_id, customer_unique_id, customer_zip_code_prefix,
customer_city, customer_state
FROM customers
```



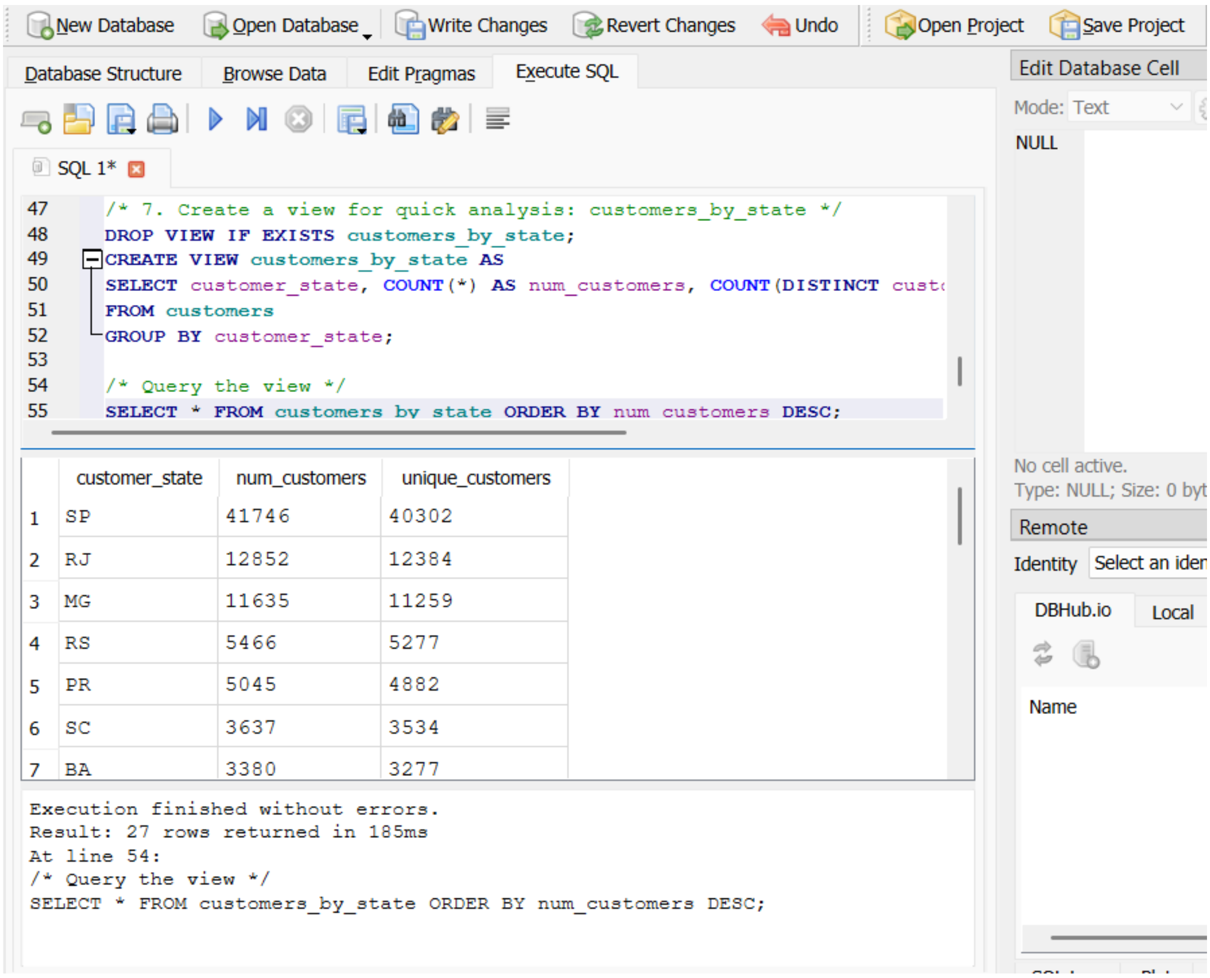
```
44    ORDER BY customer_zip_code_prefix DESC
45    LIMIT 50;
46
47    /* 7. Create a view for quick analysis: customers_by_state */
48    DROP VIEW IF EXISTS customers_by_state;
49    CREATE VIEW customers_by_state AS
50    SELECT customer_state, COUNT(*) AS num_customers, COUNT(DISTINCT custo
51    FROM customers
52    GROUP BY customer_state;
```

	customer_id	customer_unique_id	
1	f04c3f6e20a2a4318d97d7cec1d27bbf	8f846b30f8f56368cd1bf4c89efeed74	9
2	964b34423c822e9bd9831635ccc83db2	e49eafd7e69d43b8d86f6b5590fafd02	9
3	657ba09c6edfbbc09f6054f541ec1f90	3dbb390afed2043063d358900ae05d0d	9
4	3ab8bc00f8740d54afc4c771fb6c7f69	0528a0a940c7116ccb48fdbb8e80a8ff	9
5	f93bb6c70a7306d18a24f77f1889916d	ee0a41f3ec008b1459efbd99c775e6ea	9
6	235702411e5214f0e4a0761bf5ce9e99	9fc089b0b6a8db40d4357e784f1e91d4	9

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 49:
CREATE VIEW customers_by_state AS
SELECT customer_state, COUNT(*) AS num_customers, COUNT(DISTINCT
customer_unique_id) AS unique_customers
FROM customers
```

NULL

No cell active.
Type: NULL; Size: 0 bytes



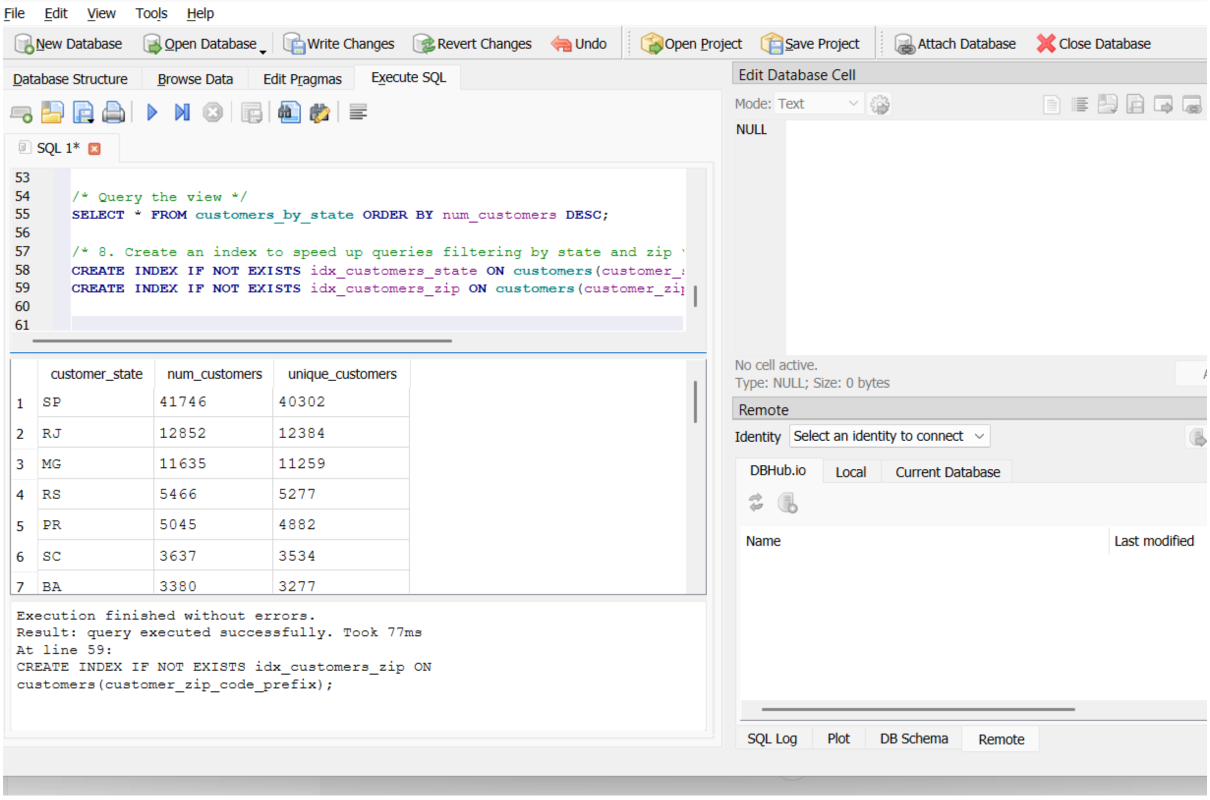
```
47    /* 7. Create a view for quick analysis: customers_by_state */
48    DROP VIEW IF EXISTS customers_by_state;
49    CREATE VIEW customers_by_state AS
50    SELECT customer_state, COUNT(*) AS num_customers, COUNT(DISTINCT custo
51    FROM customers
52    GROUP BY customer_state;
53
54    /* Query the view */
55    SELECT * FROM customers_by_state ORDER BY num_customers DESC;
```

	customer_state	num_customers	unique_customers
1	SP	41746	40302
2	RJ	12852	12384
3	MG	11635	11259
4	RS	5466	5277
5	PR	5045	4882
6	SC	3637	3534
7	BA	3380	3277

```
Execution finished without errors.
Result: 27 rows returned in 185ms
At line 54:
/* Query the view */
SELECT * FROM customers_by_state ORDER BY num_customers DESC;
```

NULL

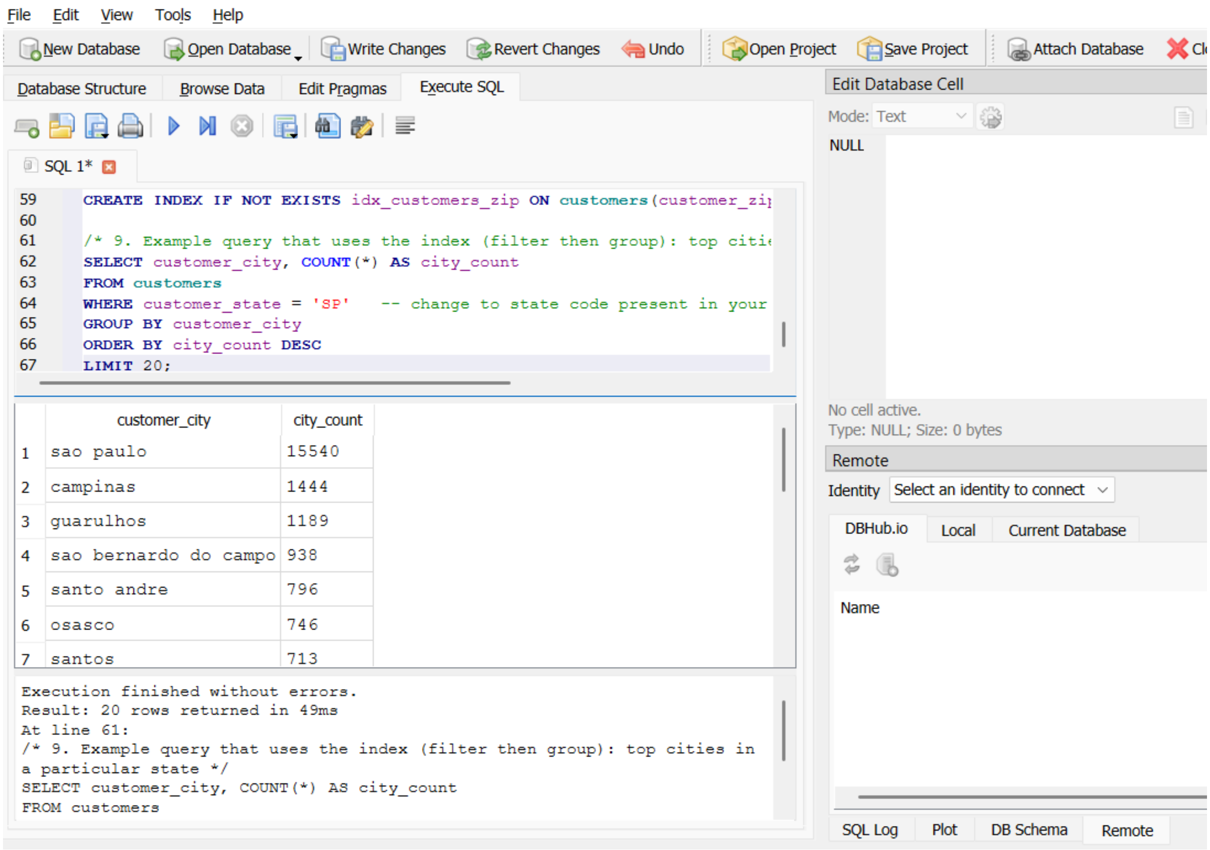
No cell active.
Type: NULL; Size: 0 byt



```

/* Query the view */
SELECT * FROM customers_by_state ORDER BY num_customers DESC;

/* 8. Create an index to speed up queries filtering by state and zip '
CREATE INDEX IF NOT EXISTS idx_customers_state ON customers(customer_s
CREATE INDEX IF NOT EXISTS idx_customers_zip ON customers(customer_zi

```

	customer_state	num_customers	unique_customers
1	SP	41746	40302
2	RJ	12852	12384
3	MG	11635	11259
4	RS	5466	5277
5	PR	5045	4882
6	SC	3637	3534
7	BA	3380	3277

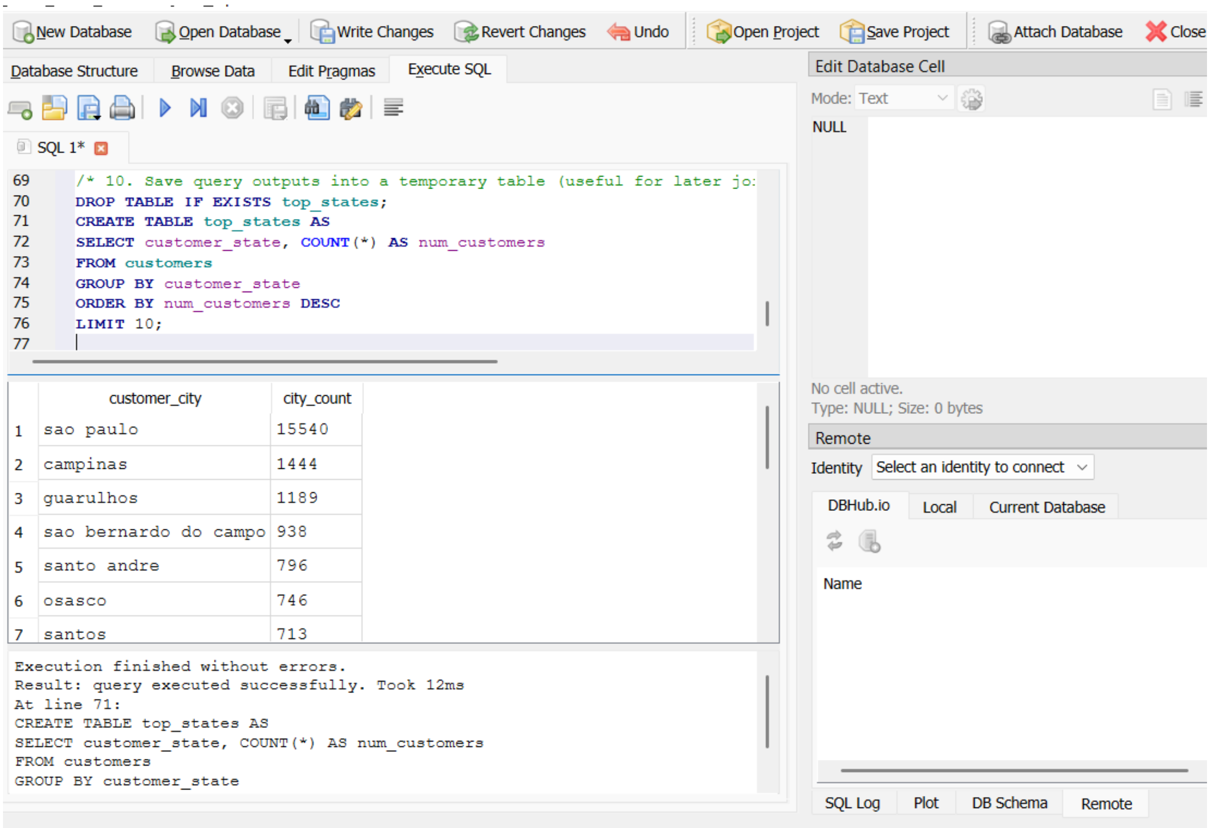
```
Execution finished without errors.
Result: query executed successfully. Took 77ms
At line 59:
CREATE INDEX IF NOT EXISTS idx_customers_zip ON
customers(customer_zip_code_prefix);
```



```
CREATE INDEX IF NOT EXISTS idx_customers_zip ON customers(customer_zi

/* 9. Example query that uses the index (filter then group): top citie
SELECT customer_city, COUNT(*) AS city_count
FROM customers
WHERE customer_state = 'SP'   -- change to state code present in your
GROUP BY customer_city
ORDER BY city_count DESC
LIMIT 20;
```

	customer_city	city_count
1	sao paulo	15540
2	campinas	1444
3	guarulhos	1189
4	sao bernardo do campo	938
5	santo andre	796
6	osasco	746
7	santos	713

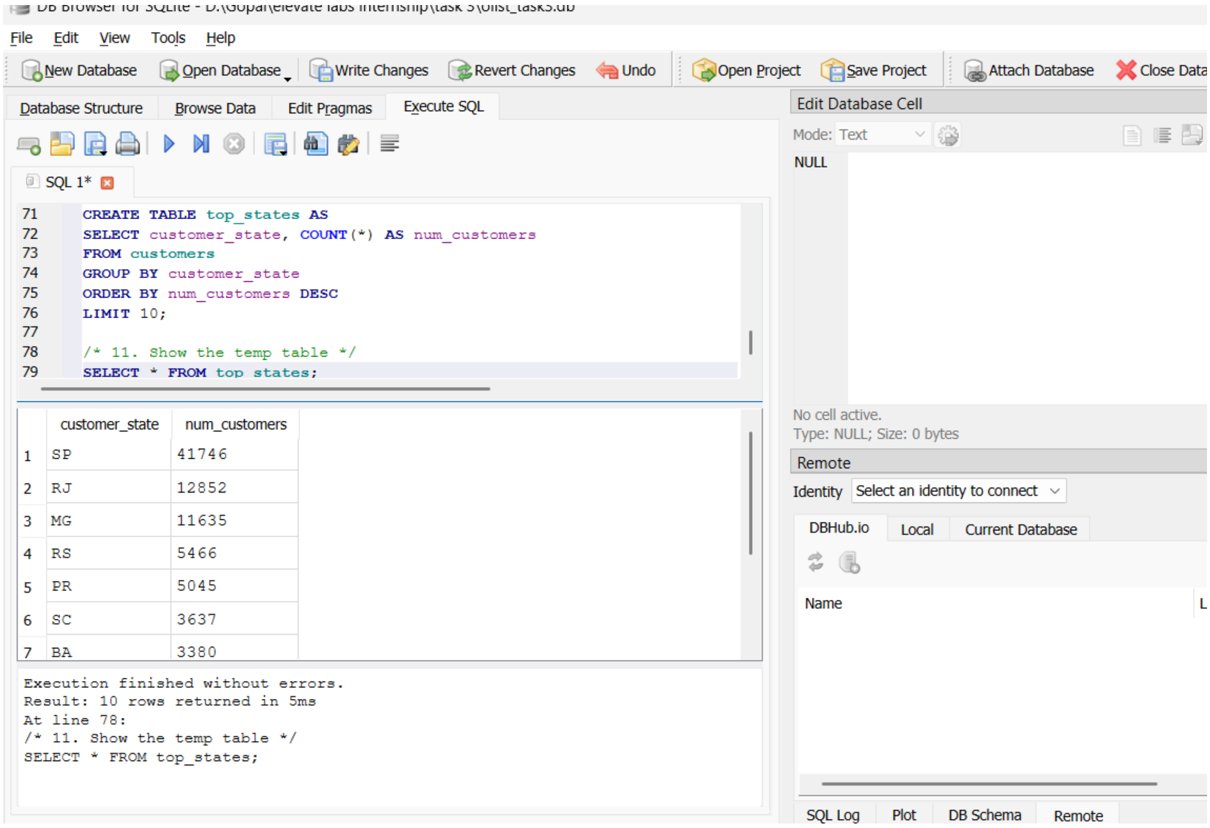
```
Execution finished without errors.
Result: 20 rows returned in 49ms
At line 61:
/* 9. Example query that uses the index (filter then group): top cities in
a particular state */
SELECT customer_city, COUNT(*) AS city_count
FROM customers
```



```sql
/* 10. Save query outputs into a temporary table (useful for later jo:
DROP TABLE IF EXISTS top_states;
CREATE TABLE top_states AS
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
ORDER BY num_customers DESC
LIMIT 10;
```

	customer_city	city_count
1	sao paulo	15540
2	campinas	1444
3	guarulhos	1189
4	sao bernardo do campo	938
5	santo andre	796
6	osasco	746
7	santos	713

```
Execution finished without errors.
Result: query executed successfully. Took 12ms
At line 71:
CREATE TABLE top_states AS
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
```



```sql
CREATE TABLE top_states AS
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
ORDER BY num_customers DESC
LIMIT 10;

/* 11. Show the temp table */
SELECT * FROM top_states;
```

	customer_state	num_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

```
Execution finished without errors.
Result: 10 rows returned in 5ms
At line 78:
/* 11. Show the temp table */
SELECT * FROM top_states;
```