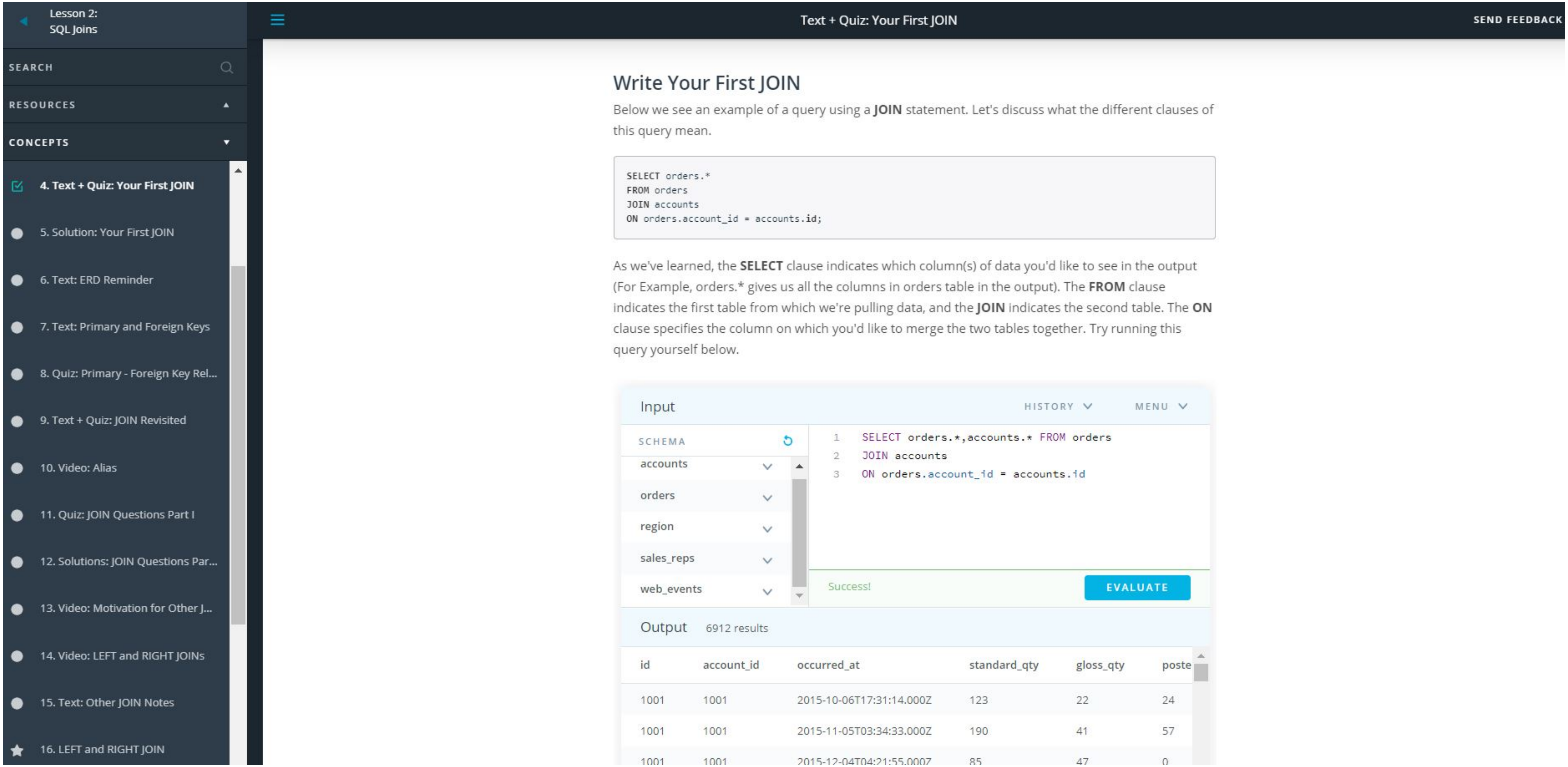

Lesson 2:
SQL Joins
Text + Quiz: Your First JOIN
SEND FEEDBACK
SEARCH
RESOURCES
CONCEPTS
4. Text + Quiz: Your First JOIN
5. Solution: Your First JOIN
6. Text: ERD Reminder
7. Text: Primary and Foreign Keys
8. Quiz: Primary - Foreign Key Rel...
9. Text + Quiz: JOIN Revisited
10. Video: Alias
11. Quiz: JOIN Questions Part I
12. Solutions: JOIN Questions Par...
13. Video: Motivation for Other J...
14. Video: LEFT and RIGHT JOINs
15. Text: Other JOIN Notes
16. LEFT and RIGHT JOIN

Write Your First JOIN

Below we see an example of a query using a **JOIN** statement. Let's discuss what the different clauses of this query mean.

```
SELECT orders.*
FROM orders
JOIN accounts
ON orders.account_id = accounts.id;
```

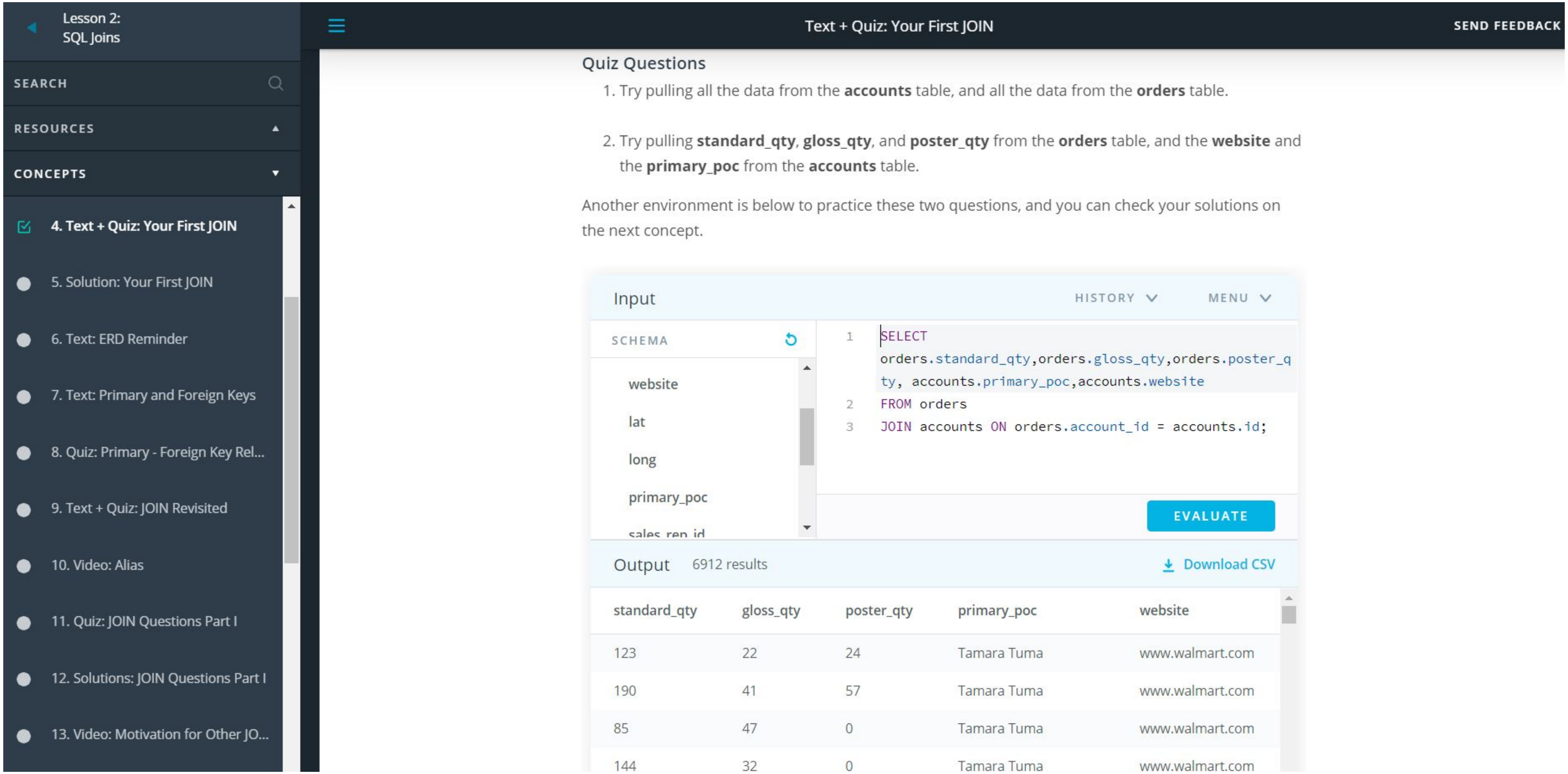
As we've learned, the **SELECT** clause indicates which column(s) of data you'd like to see in the output (For Example, orders.* gives us all the columns in orders table in the output). The **FROM** clause indicates the first table from which we're pulling data, and the **JOIN** indicates the second table. The **ON** clause specifies the column on which you'd like to merge the two tables together. Try running this query yourself below.

```
SELECT orders.*,accounts.* FROM orders
JOIN accounts
ON orders.account_id = accounts.id
```

Success!

Output 6912 results

id	account_id	occurred_at	standard_qty	gloss_qty	poste
1001	1001	2015-10-06T17:31:14.000Z	123	22	24
1001	1001	2015-11-05T03:34:33.000Z	190	41	57
1001	1001	2015-12-04T04:21:55.000Z	85	47	0


Lesson 2:
SQL Joins
SEARCH
RESOURCES ▲
CONCEPTS ▼
☑ 4. Text + Quiz: Your First JOIN
● 5. Solution: Your First JOIN
● 6. Text: ERD Reminder
● 7. Text: Primary and Foreign Keys
● 8. Quiz: Primary - Foreign Key Rel...
● 9. Text + Quiz: JOIN Revisited
● 10. Video: Alias
● 11. Quiz: JOIN Questions Part I
● 12. Solutions: JOIN Questions Part I
● 13. Video: Motivation for Other JO...
Text + Quiz: Your First JOIN
SEND FEEDBACK

Quiz Questions

1. Try pulling all the data from the **accounts** table, and all the data from the **orders** table.

2. Try pulling **standard_qty**, **gloss_qty**, and **poster_qty** from the **orders** table, and the **website** and the **primary_poc** from the **accounts** table.

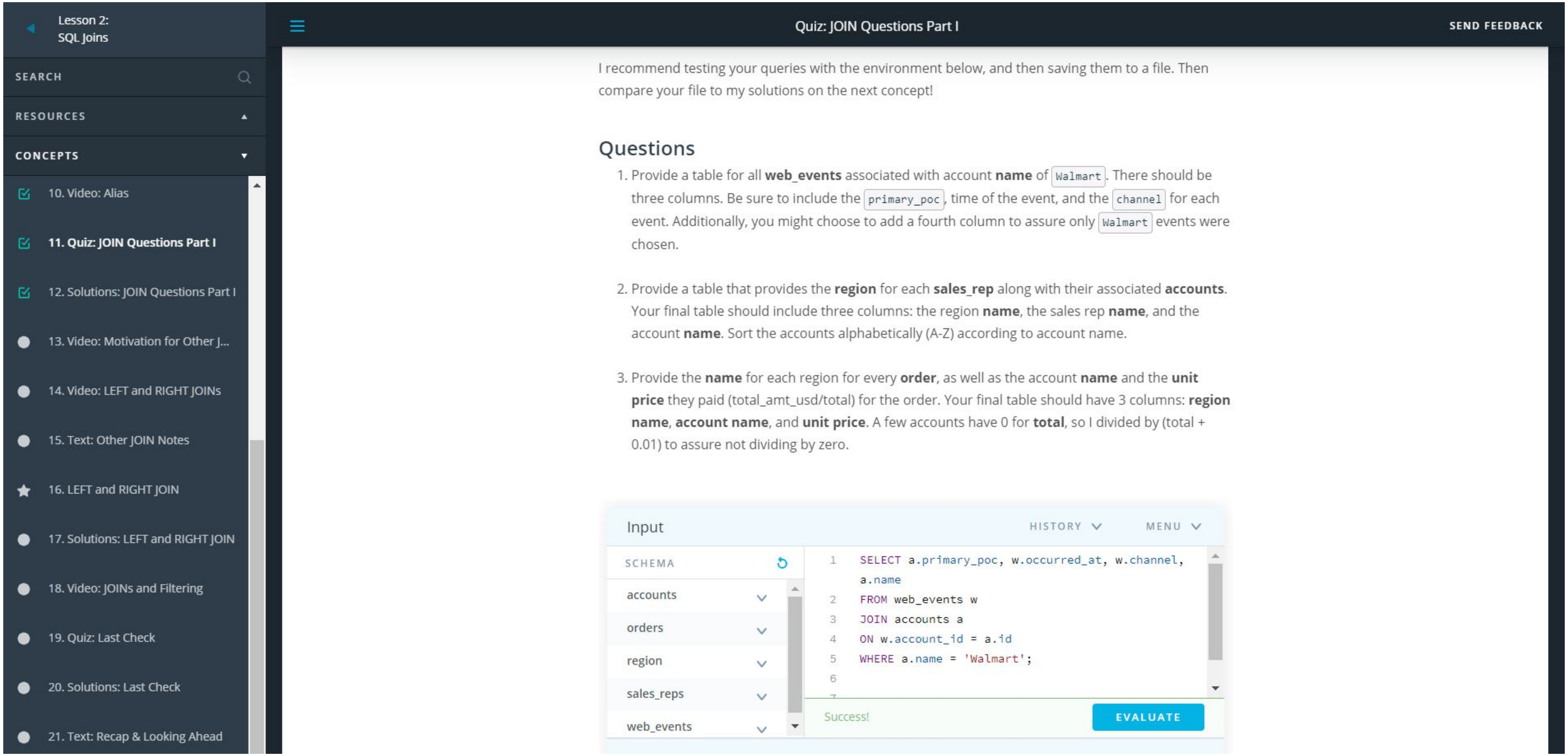
Another environment is below to practice these two questions, and you can check your solutions on the next concept.

Input
HISTORY
MENU
SCHEMA
website
lat
long
primary_poc
sales_rep_id
EVALUATE

```
SELECT
orders.standard_qty,orders.gloss_qty,orders.poster_qty, accounts.primary_poc,accounts.website
FROM orders
JOIN accounts ON orders.account_id = accounts.id;
```

6912 results

standard_qty	gloss_qty	poster_qty	primary_poc	website
123	22	24	Tamara Tuma	www.walmart.com
190	41	57	Tamara Tuma	www.walmart.com
85	47	0	Tamara Tuma	www.walmart.com
144	32	0	Tamara Tuma	www.walmart.com



Lesson 2:
SQL Joins
SEARCH
RESOURCES ▴
CONCEPTS ▾
☑ 10. Video: Alias
☑ 11. Quiz: JOIN Questions Part I
☑ 12. Solutions: JOIN Questions Part I
● 13. Video: Motivation for Other J...
● 14. Video: LEFT and RIGHT JOINs
● 15. Text: Other JOIN Notes
★ 16. LEFT and RIGHT JOIN
● 17. Solutions: LEFT and RIGHT JOIN
● 18. Video: JOINs and Filtering
● 19. Quiz: Last Check
● 20. Solutions: Last Check
● 21. Text: Recap & Looking Ahead
Quiz: JOIN Questions Part I
SEND FEEDBACK

I recommend testing your queries with the environment below, and then saving them to a file. Then compare your file to my solutions on the next concept!

Questions

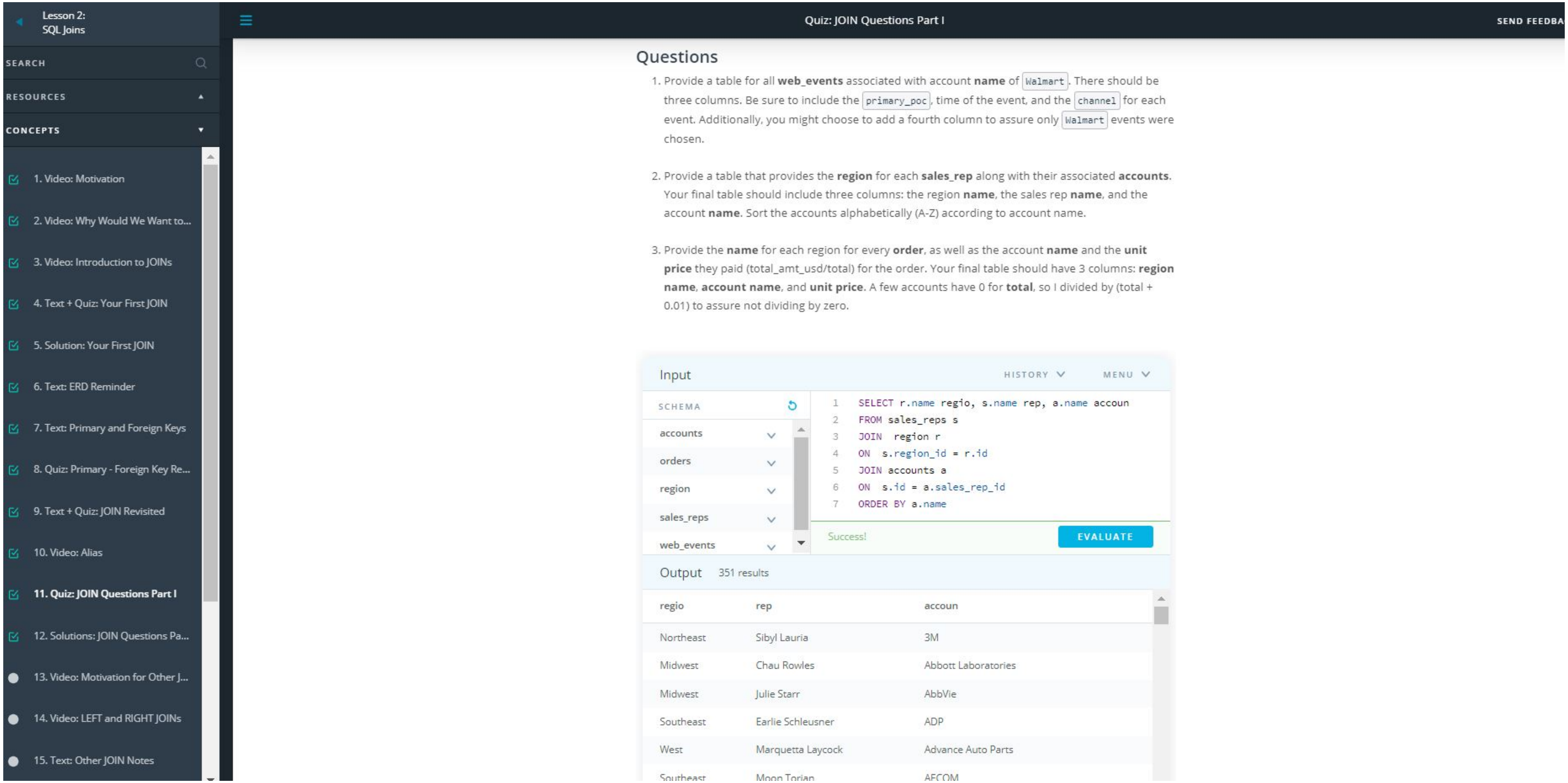
1. Provide a table for all **web_events** associated with account **name** of `Walmart`. There should be three columns. Be sure to include the `primary_poc`, time of the event, and the `channel` for each event. Additionally, you might choose to add a fourth column to assure only `Walmart` events were chosen.

2. Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

3. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. A few accounts have 0 for **total**, so I divided by (total + 0.01) to assure not dividing by zero.

```sql
SELECT a.primary_poc, w.occurred_at, w.channel,
a.name
FROM web_events w
JOIN accounts a
ON w.account_id = a.id
WHERE a.name = 'Walmart';
```

Success!


Lesson 2:
SQL Joins
SEARCH
RESOURCES ▲
CONCEPTS ▼
☑ 1. Video: Motivation
☑ 2. Video: Why Would We Want to...
☑ 3. Video: Introduction to JOINs
☑ 4. Text + Quiz: Your First JOIN
☑ 5. Solution: Your First JOIN
☑ 6. Text: ERD Reminder
☑ 7. Text: Primary and Foreign Keys
☑ 8. Quiz: Primary - Foreign Key Re...
☑ 9. Text + Quiz: JOIN Revisited
☑ 10. Video: Alias
☑ 11. Quiz: JOIN Questions Part I
☑ 12. Solutions: JOIN Questions Pa...
● 13. Video: Motivation for Other J...
● 14. Video: LEFT and RIGHT JOINs
● 15. Text: Other JOIN Notes
Quiz: JOIN Questions Part I
SEND FEEDBA

Questions

1. Provide a table for all **web_events** associated with account **name** of `Walmart`. There should be three columns. Be sure to include the `primary_poc`, time of the event, and the `channel` for each event. Additionally, you might choose to add a fourth column to assure only `Walmart` events were chosen.

2. Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

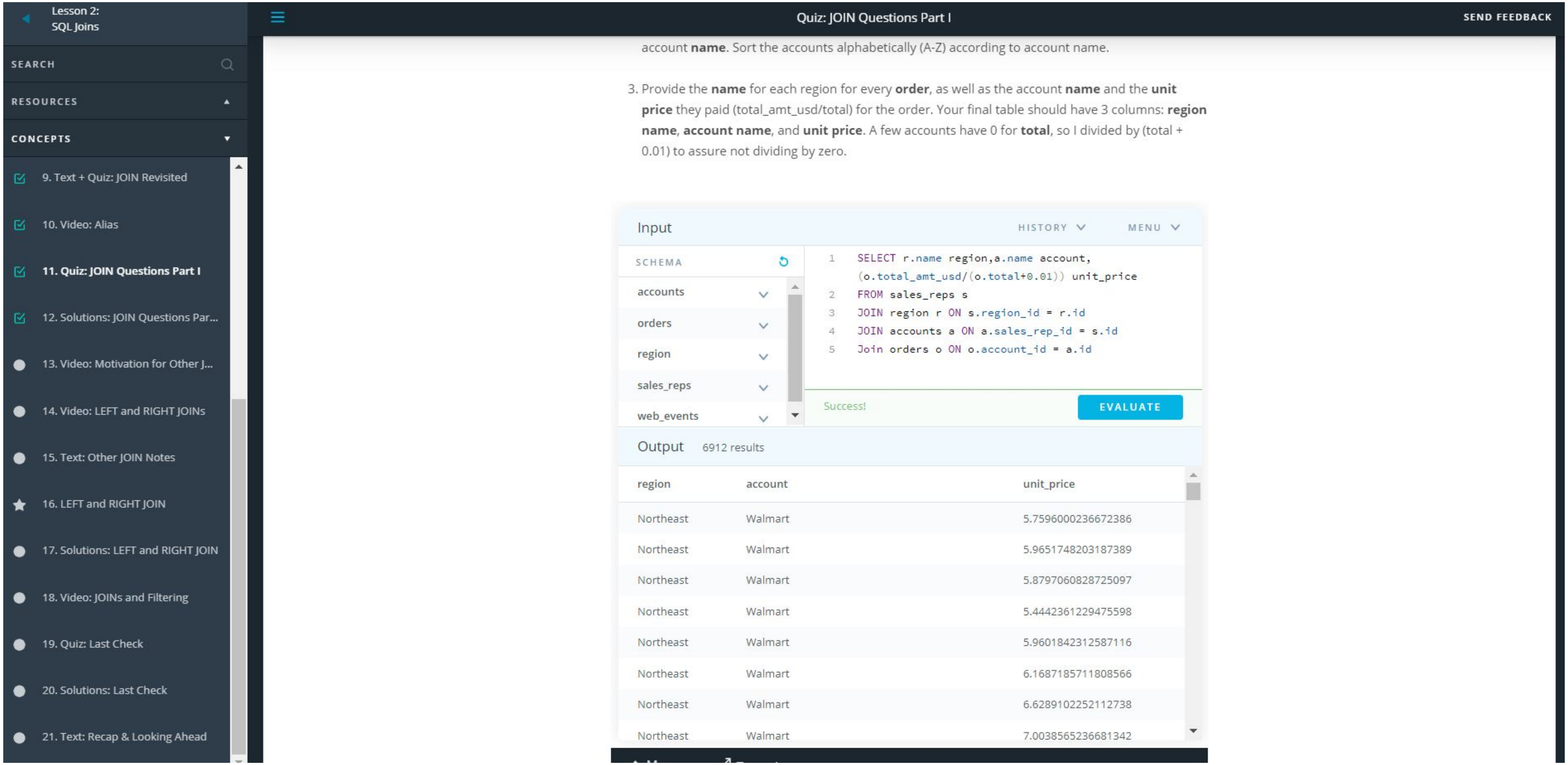
3. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. A few accounts have 0 for **total**, so I divided by (total + 0.01) to assure not dividing by zero.

```
SELECT r.name regio, s.name rep, a.name accoun
FROM sales_reps s
JOIN  region r
ON  s.region_id = r.id
JOIN accounts a
ON  s.id = a.sales_rep_id
ORDER BY a.name
```

Success!

Output 351 results

regio	rep	accoun
Northeast	Sibyl Lauria	3M
Midwest	Chau Rowles	Abbott Laboratories
Midwest	Julie Starr	AbbVie
Southeast	Earlie Schleusner	ADP
West	Marquetta Laycock	Advance Auto Parts
Southeast	Moon Torian	AECOM


Lesson 2:
SQL Joins
SEARCH
RESOURCES ▲
CONCEPTS ▼
☑ 9. Text + Quiz: JOIN Revisited
☑ 10. Video: Alias
☑ 11. Quiz: JOIN Questions Part I
☑ 12. Solutions: JOIN Questions Par...
● 13. Video: Motivation for Other J...
● 14. Video: LEFT and RIGHT JOINs
● 15. Text: Other JOIN Notes
★ 16. LEFT and RIGHT JOIN
● 17. Solutions: LEFT and RIGHT JOIN
● 18. Video: JOINs and Filtering
● 19. Quiz: Last Check
● 20. Solutions: Last Check
● 21. Text: Recap & Looking Ahead
Quiz: JOIN Questions Part I
SEND FEEDBACK

account **name**. Sort the accounts alphabetically (A-Z) according to account name.

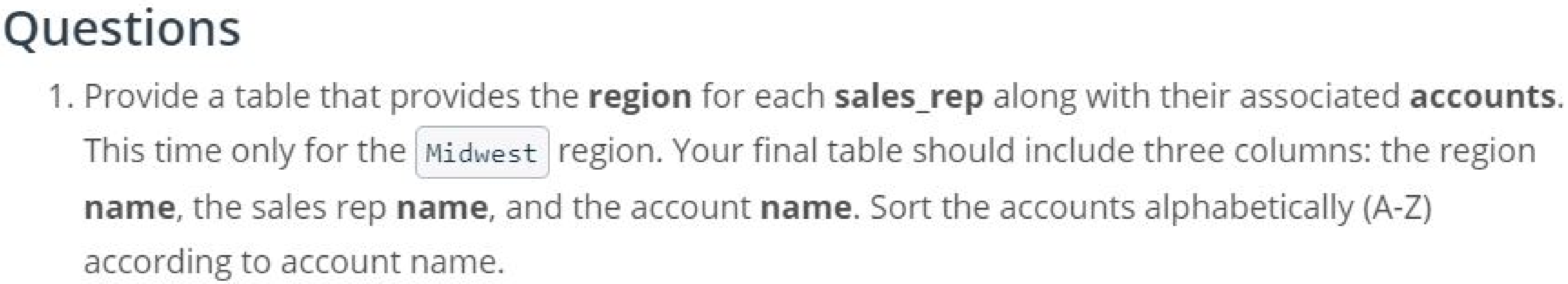
3. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. A few accounts have 0 for **total**, so I divided by (total + 0.01) to assure not dividing by zero.

```
SELECT r.name region,a.name account,
(o.total_amt_usd/(o.total+0.01)) unit_price
FROM sales_reps s
JOIN region r ON s.region_id = r.id
JOIN accounts a ON a.sales_rep_id = s.id
Join orders o ON o.account_id = a.id
```

Success!

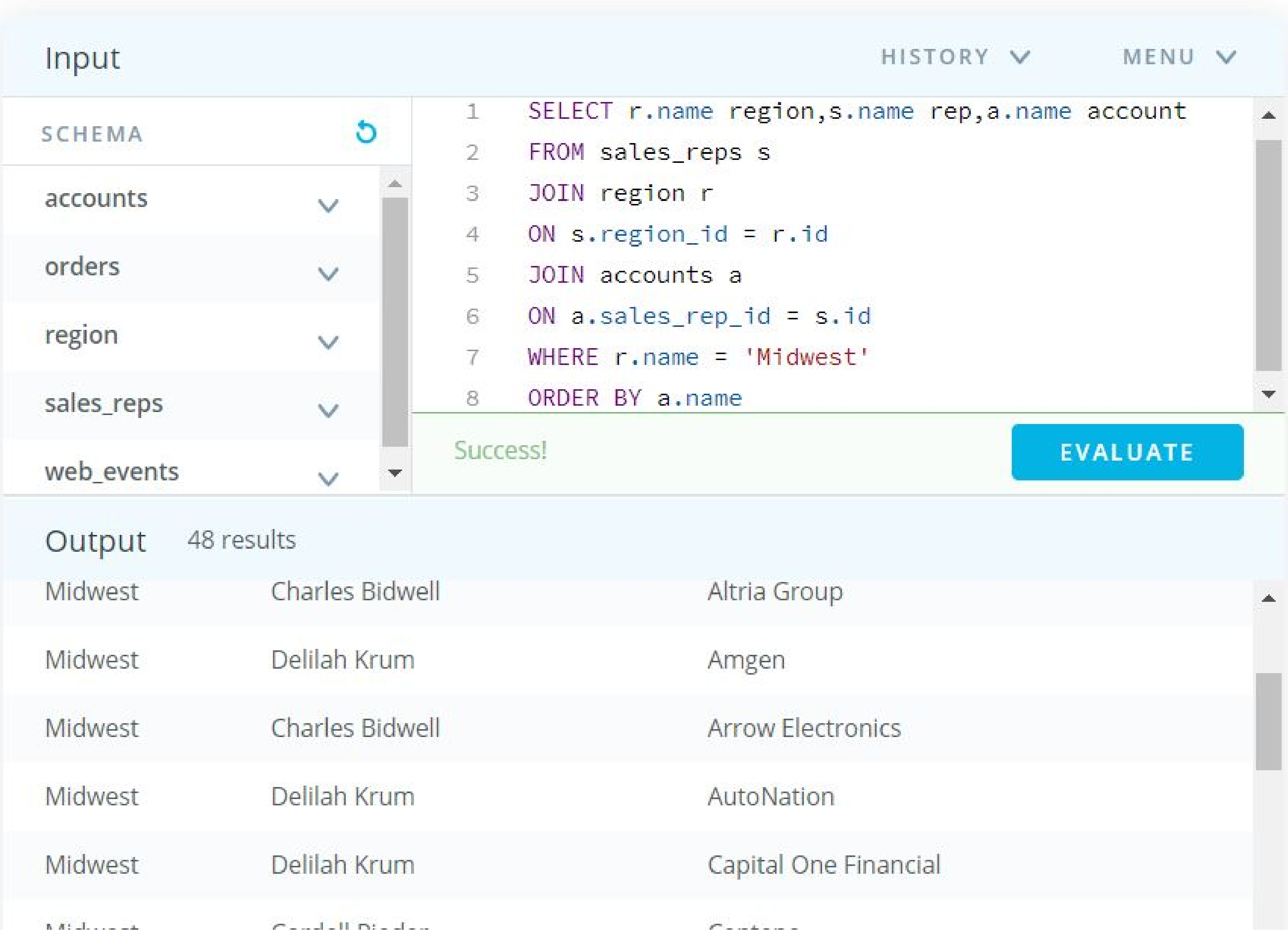
Output 6912 results

region	account	unit_price
Northeast	Walmart	5.7596000236672386
Northeast	Walmart	5.9651748203187389
Northeast	Walmart	5.8797060828725097
Northeast	Walmart	5.4442361229475598
Northeast	Walmart	5.9601842312587116
Northeast	Walmart	6.1687185711808566
Northeast	Walmart	6.6289102252112738
Northeast	Walmart	7.0038565236681342



Questions

1. Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. This time only for the `Midwest` region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

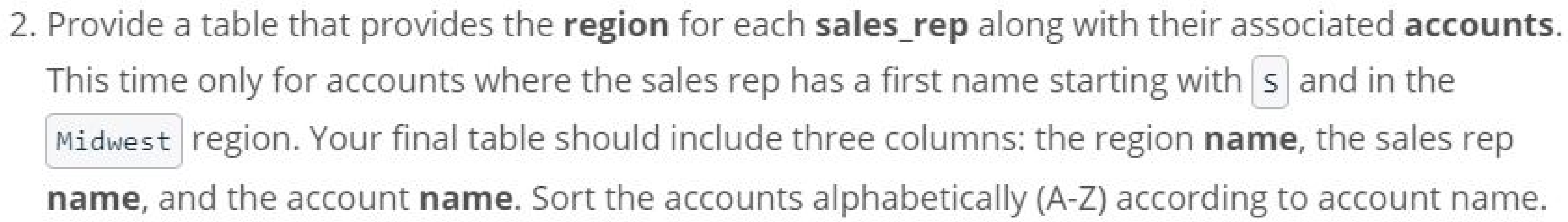


```sql
SELECT r.name region,s.name rep,a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
WHERE r.name = 'Midwest'
ORDER BY a.name
```

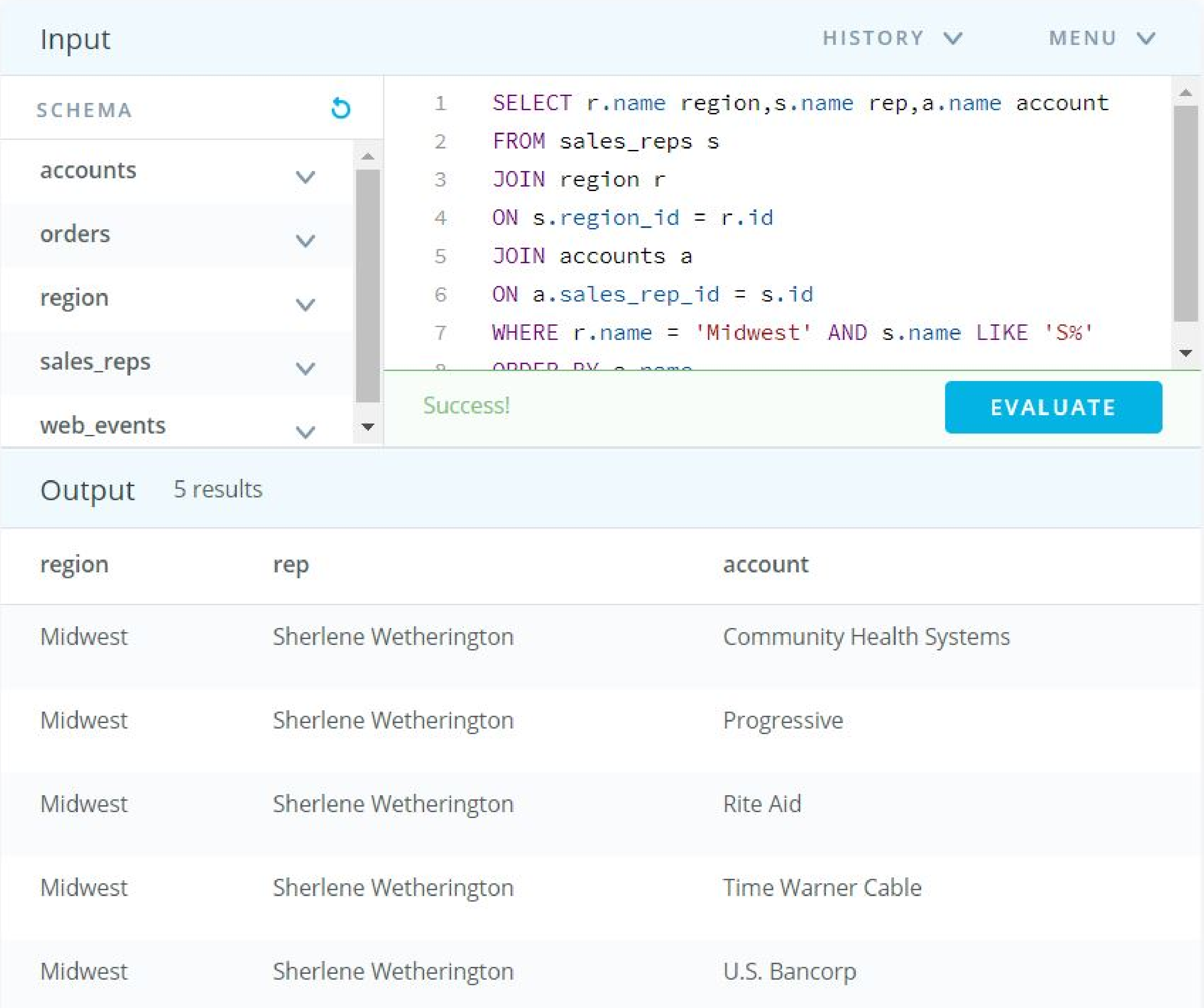
Success!

Output 48 results

```
Midwest     Charles Bidwell     Altria Group
Midwest     Delilah Krum        Amgen
Midwest     Charles Bidwell     Arrow Electronics
Midwest     Delilah Krum        AutoNation
Midwest     Delilah Krum        Capital One Financial
Midwest     Cordell Rieder      Centene
```



2. Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a first name starting with `S` and in the `Midwest` region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

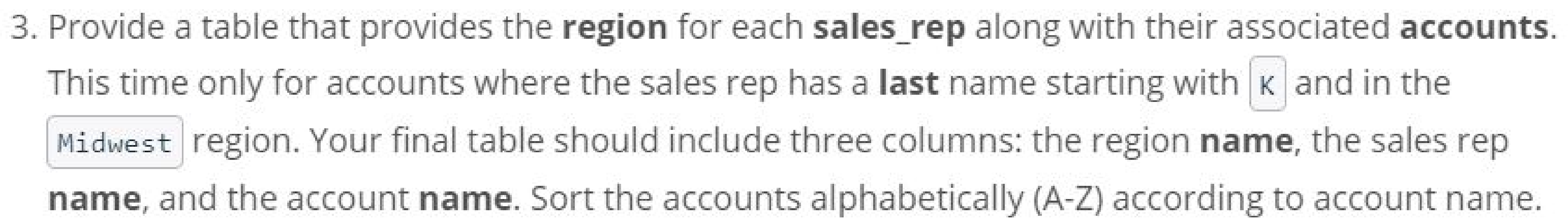


```sql
SELECT r.name region,s.name rep,a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
WHERE r.name = 'Midwest' AND s.name LIKE 'S%'
ORDER BY a.name
```

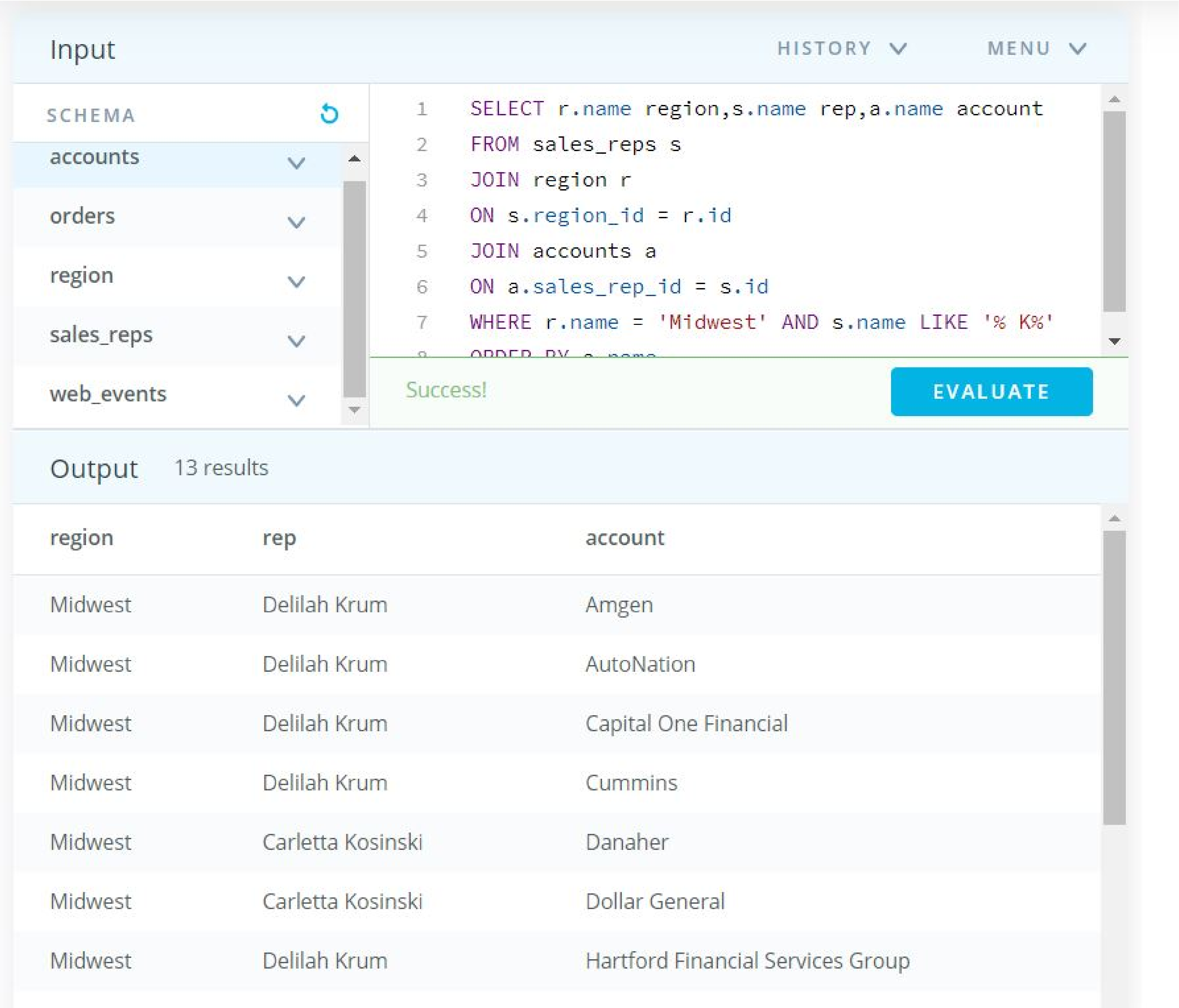
Success!

Output 5 results

region	rep	account
Midwest	Sherlene Wetherington	Community Health Systems
Midwest	Sherlene Wetherington	Progressive
Midwest	Sherlene Wetherington	Rite Aid
Midwest	Sherlene Wetherington	Time Warner Cable
Midwest	Sherlene Wetherington	U.S. Bancorp

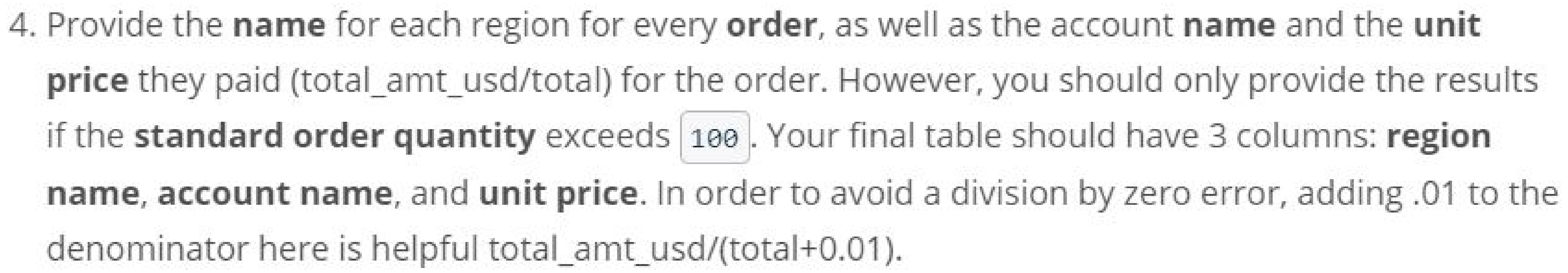


3. Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a **last** name starting with `K` and in the `Midwest` region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

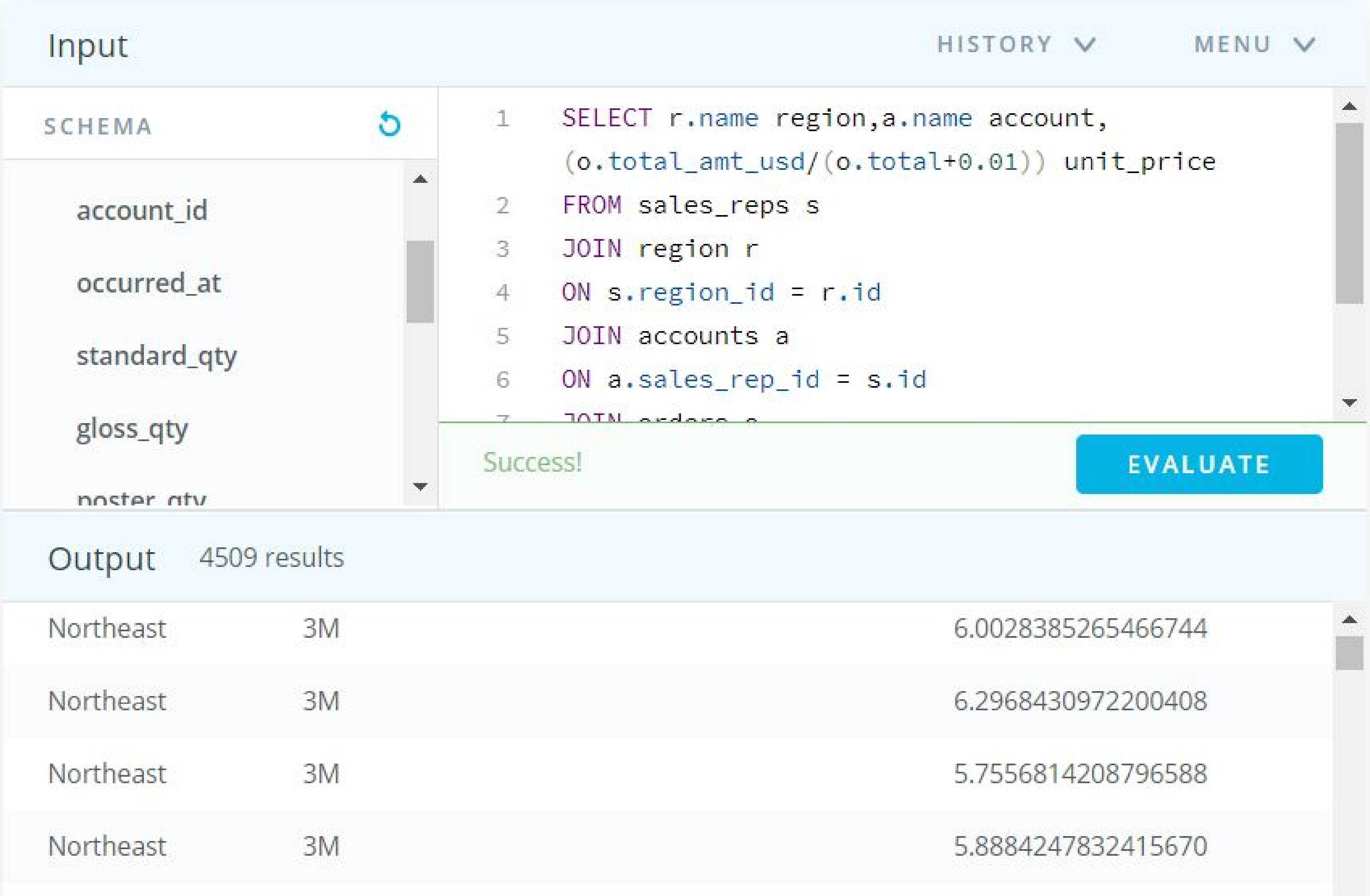


```
SELECT r.name region,s.name rep,a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
WHERE r.name = 'Midwest' AND s.name LIKE '% K%'
ORDER BY a.name
```

region	rep	account
Midwest	Delilah Krum	Amgen
Midwest	Delilah Krum	AutoNation
Midwest	Delilah Krum	Capital One Financial
Midwest	Delilah Krum	Cummins
Midwest	Carletta Kosinski	Danaher
Midwest	Carletta Kosinski	Dollar General
Midwest	Delilah Krum	Hartford Financial Services Group



4. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds `100`. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. In order to avoid a division by zero error, adding .01 to the denominator here is helpful total_amt_usd/(total+0.01).



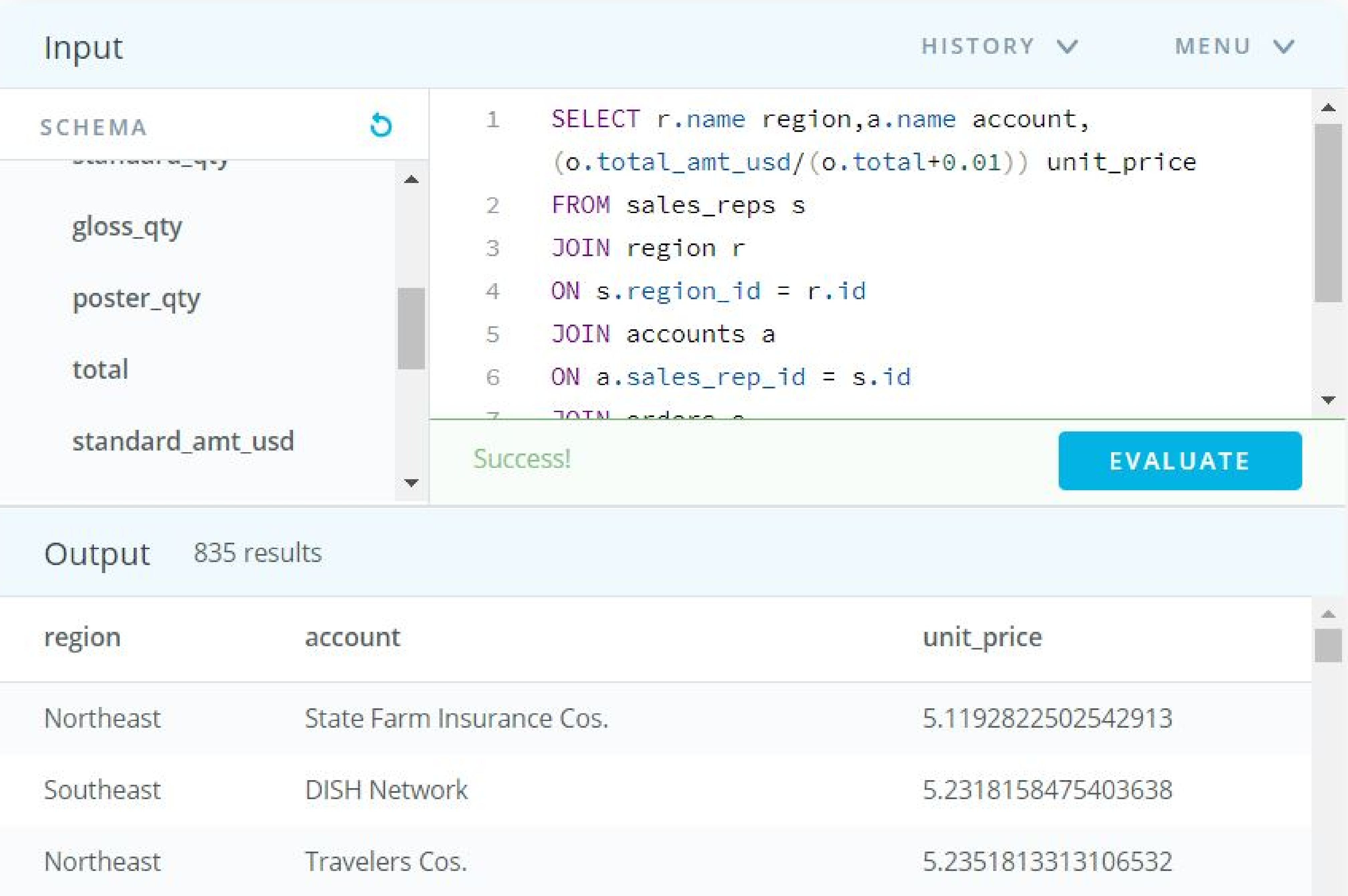
```
SELECT r.name region,a.name account,
(o.total_amt_usd/(o.total+0.01)) unit_price
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
```

Success!

4509 results

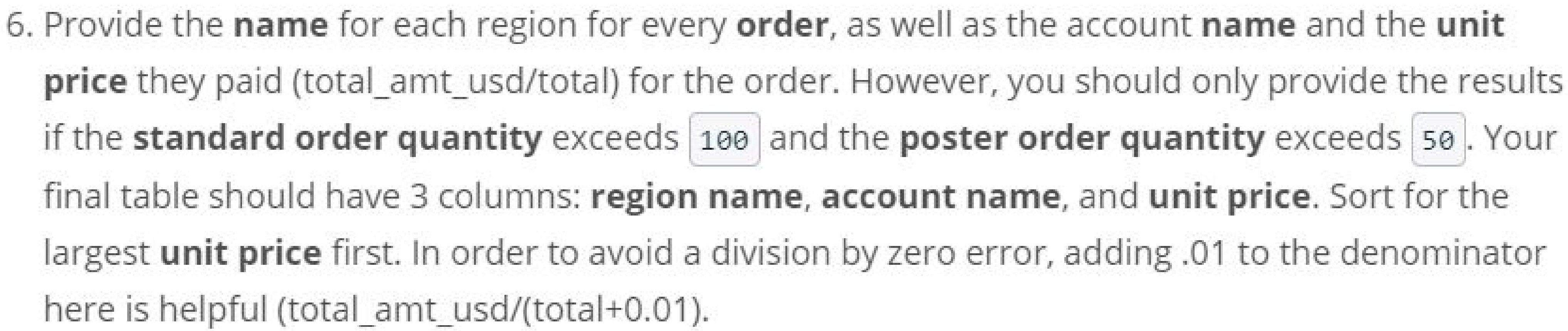
```
Northeast    3M    6.0028385265466744
Northeast    3M    6.2968430972200408
Northeast    3M    5.7556814208796588
Northeast    3M    5.8884247832415670
```

5. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds `100` and the **poster order quantity** exceeds `50`. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. Sort for the smallest **unit price** first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total_amt_usd/(total+0.01).

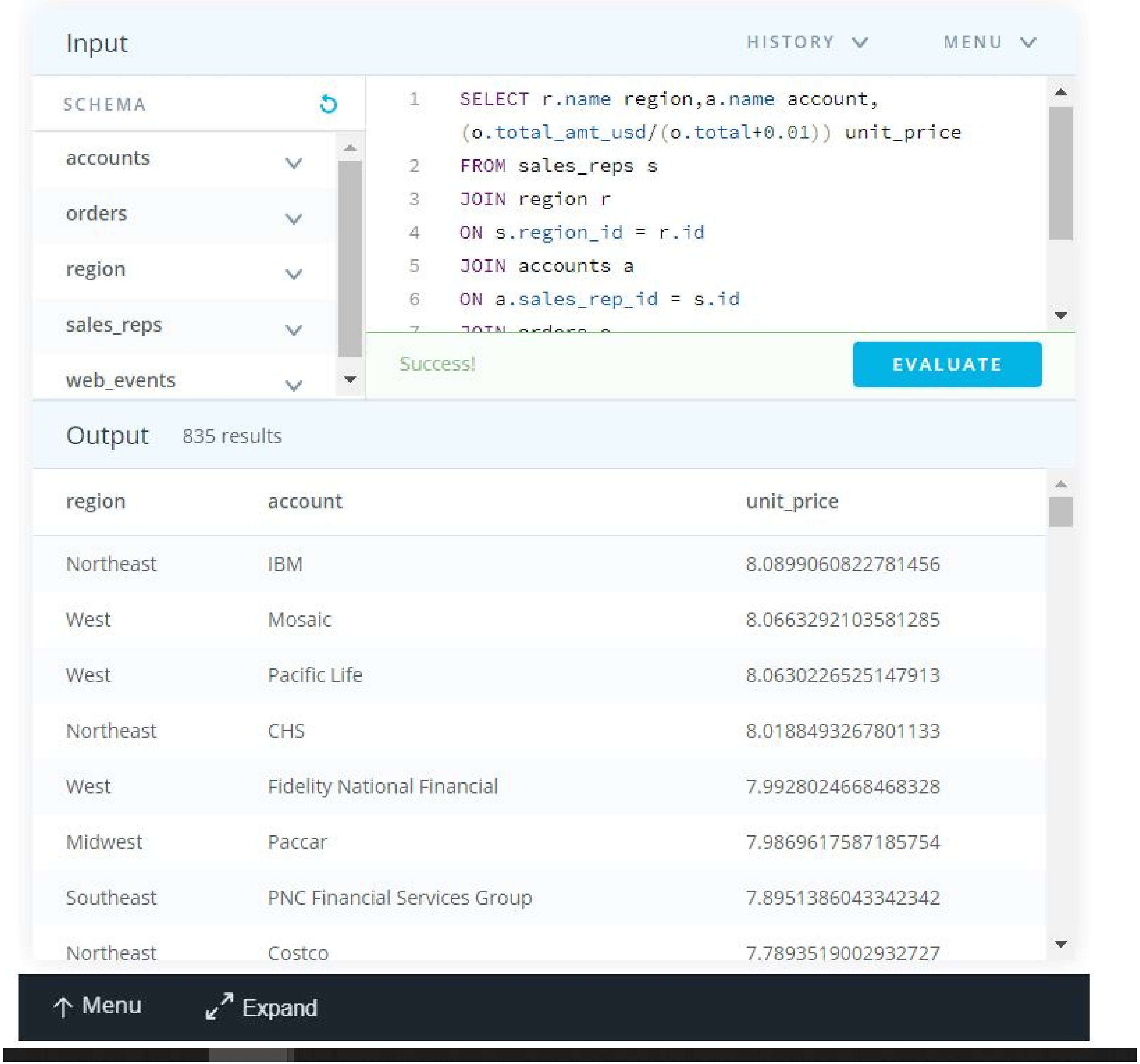


```sql
SELECT r.name region,a.name account,
(o.total_amt_usd/(o.total+0.01)) unit_price
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
```

region	account	unit_price
Northeast	State Farm Insurance Cos.	5.1192822502542913
Southeast	DISH Network	5.2318158475403638
Northeast	Travelers Cos.	5.2351813313106532



6. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds `100` and the **poster order quantity** exceeds `50`. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. Sort for the largest **unit price** first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total_amt_usd/(total+0.01).

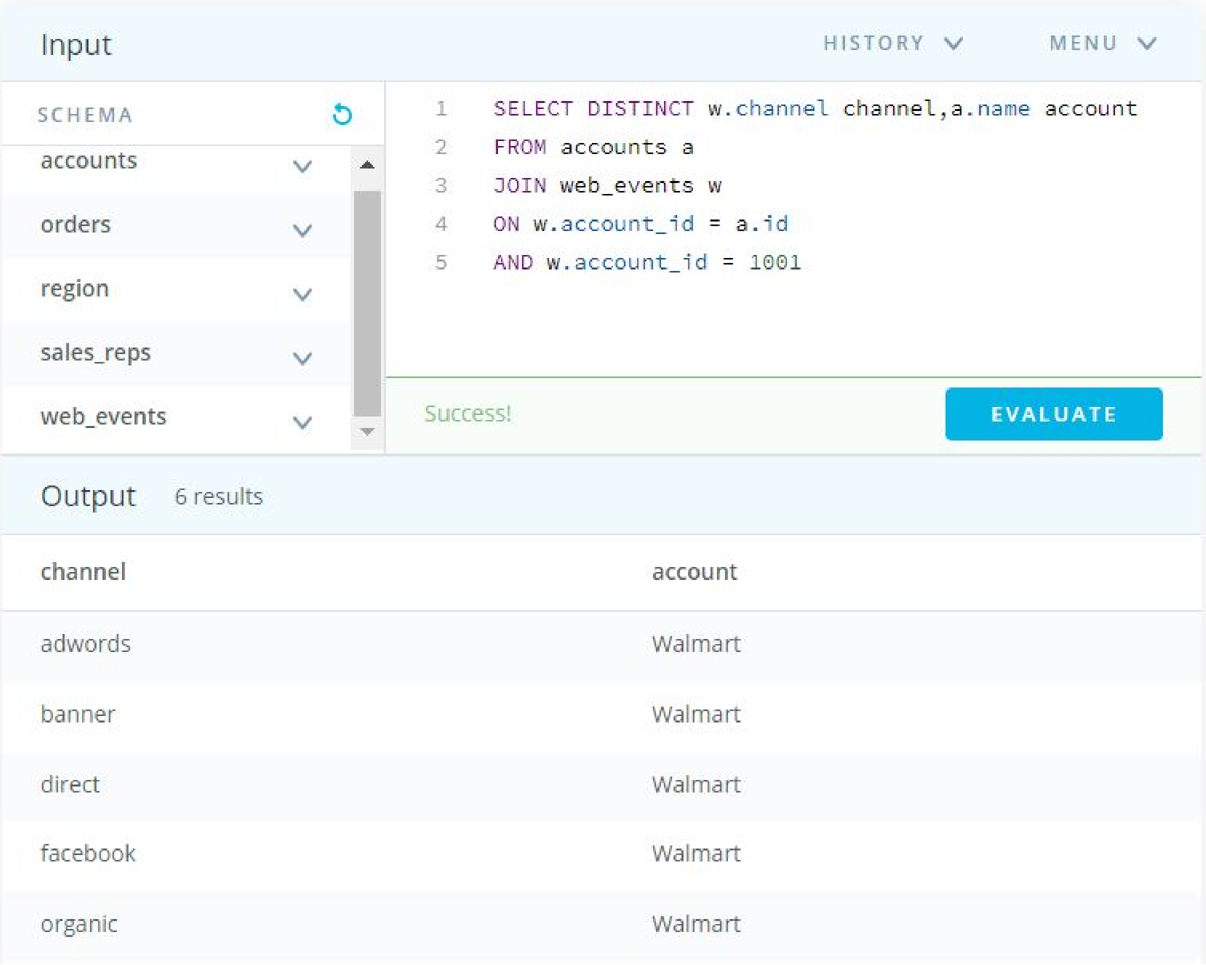


```
SELECT r.name region,a.name account,
(o.total_amt_usd/(o.total+0.01)) unit_price
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
```

region	account	unit_price
Northeast	IBM	8.0899060822781456
West	Mosaic	8.0663292103581285
West	Pacific Life	8.0630226525147913
Northeast	CHS	8.0188493267801133
West	Fidelity National Financial	7.9928024668468328
Midwest	Paccar	7.9869617587185754
Southeast	PNC Financial Services Group	7.8951386043342342
Northeast	Costco	7.7893519002932727

7. What are the different **channel**s used by **account id** `1001`? Your final table should have only 2 columns: **account name** and the different **channel**s. You can try **SELECT DISTINCT** to narrow down the results to only the unique values.

8. Find all the orders that occurred in `2015`. Your final table should have 4 columns: **occurred_at**, **account name**, **order total**, and **order total_amt_usd**.



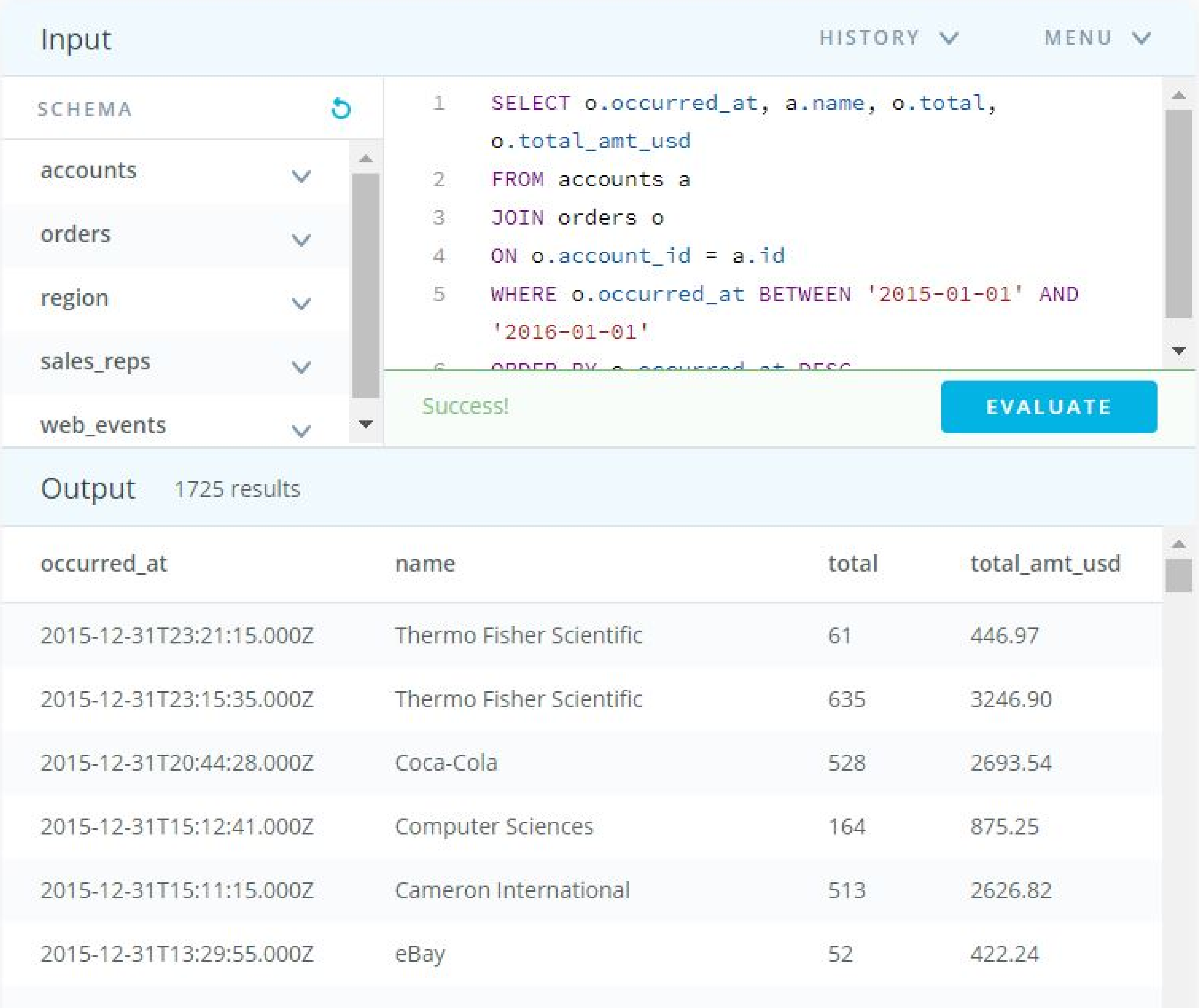
```sql
SELECT DISTINCT w.channel channel,a.name account
FROM accounts a
JOIN web_events w
ON w.account_id = a.id
AND w.account_id = 1001
```

Success!

6 results

channel	account
adwords	Walmart
banner	Walmart
direct	Walmart
facebook	Walmart
organic	Walmart

8. Find all the orders that occurred in `2015`. Your final table should have 4 columns: **occurred_at**, **account name**, **order total**, and **order total_amt_usd**.



```sql
SELECT o.occurred_at, a.name, o.total, o.total_amt_usd
FROM accounts a
JOIN orders o
ON o.account_id = a.id
WHERE o.occurred_at BETWEEN '2015-01-01' AND '2016-01-01'
ORDER BY o.occurred_at DESC
```

Success!

Output 1725 results

occurred_at	name	total	total_amt_usd
2015-12-31T23:21:15.000Z	Thermo Fisher Scientific	61	446.97
2015-12-31T23:15:35.000Z	Thermo Fisher Scientific	635	3246.90
2015-12-31T20:44:28.000Z	Coca-Cola	528	2693.54
2015-12-31T15:12:41.000Z	Computer Sciences	164	875.25
2015-12-31T15:11:15.000Z	Cameron International	513	2626.82
2015-12-31T13:29:55.000Z	eBay	52	422.24