LEARN SQL FROM SCRATCH

Calculating Churn Rates

By

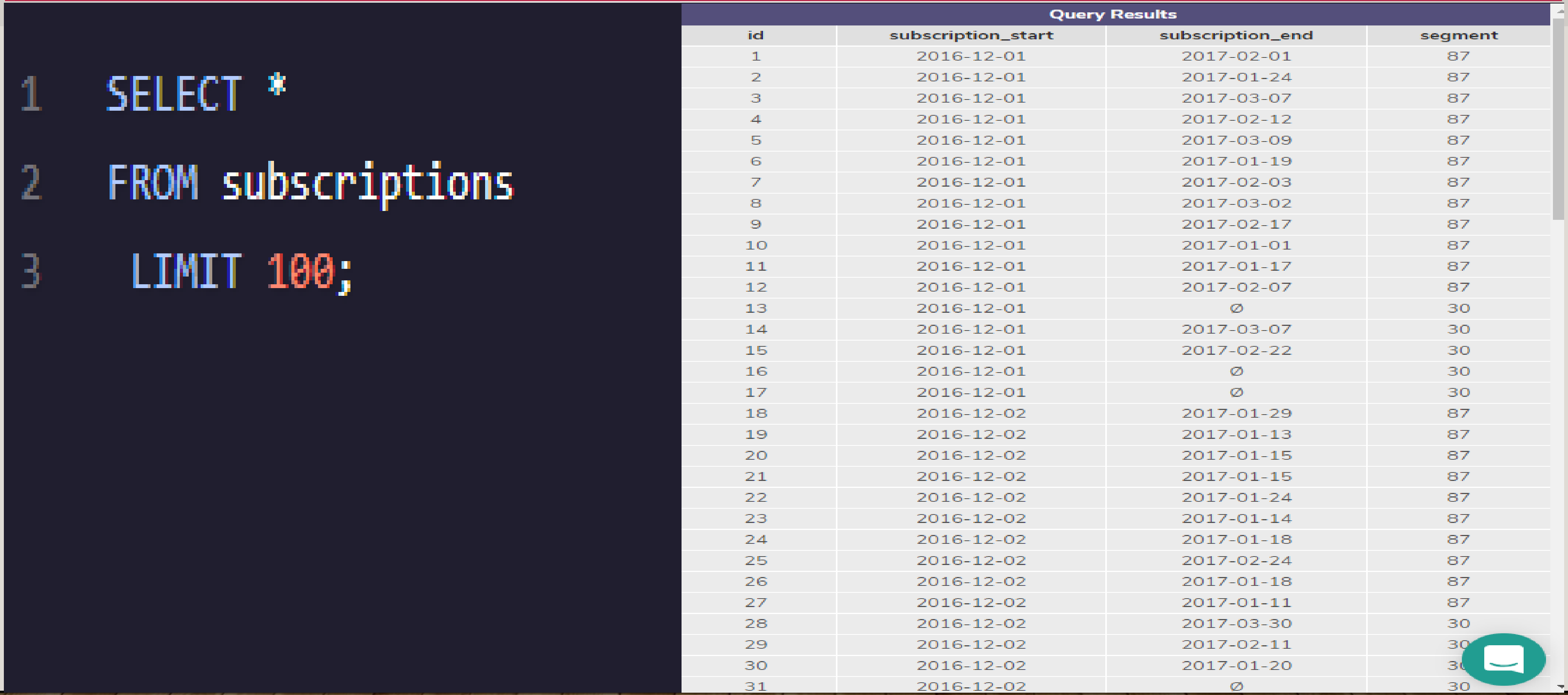
Jean Baptiste Aunial

TABLE OF CONTENTS

1. GET FAMILIAR WITH THE COMPANY.

TAKE A LOOK AT THE FIRST 100 ROWS OF DATA IN THE SUBSCRIPTIONS TABLE.
HOW MANY DIFFERENT SEGMENTS DO YOU SEE?
THE SUBSCRIPTION OWNER BELONG TO TWO SEGMENT, 87 AND 31.

THE COMPANY HAS BEEN OPERATING FOR FOUR MONTHS, BUT WE HAVE INFORMATION TO CALCULATE THE CHURN RATE FOR THREE MONTHS.



```
SELECT *
FROM subscriptions
 LIMIT 100;
```

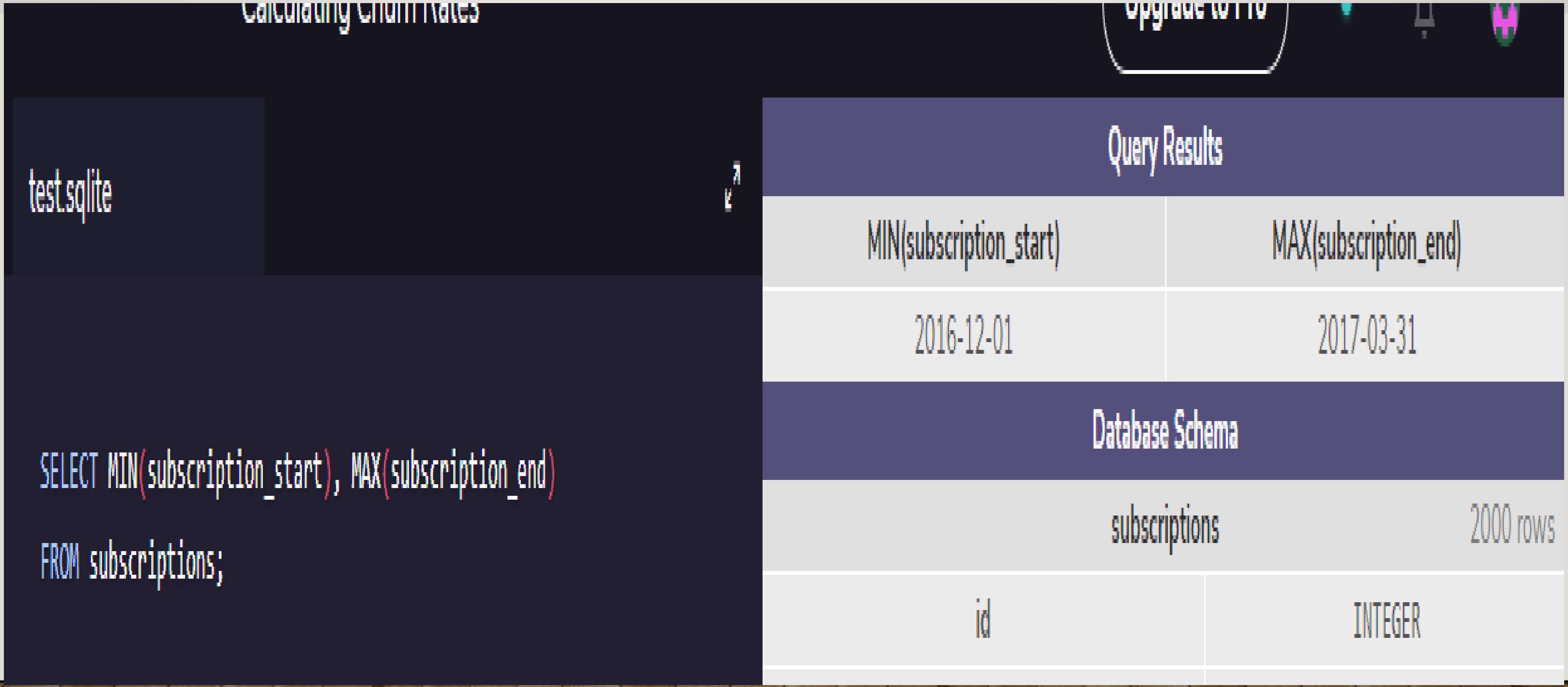
Query Results

id	subscription_start	subscription_end	segment
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87
3	2016-12-01	2017-03-07	87
4	2016-12-01	2017-02-12	87
5	2016-12-01	2017-03-09	87
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87
11	2016-12-01	2017-01-17	87
12	2016-12-01	2017-02-07	87
13	2016-12-01	∅	30
14	2016-12-01	2017-03-07	30
15	2016-12-01	2017-02-22	30
16	2016-12-01	∅	30
17	2016-12-01	∅	30
18	2016-12-02	2017-01-29	87
19	2016-12-02	2017-01-13	87
20	2016-12-02	2017-01-15	87
21	2016-12-02	2017-01-15	87
22	2016-12-02	2017-01-24	87
23	2016-12-02	2017-01-14	87
24	2016-12-02	2017-01-18	87
25	2016-12-02	2017-02-24	87
26	2016-12-02	2017-01-18	87
27	2016-12-02	2017-01-11	87
28	2016-12-02	2017-03-30	30
29	2016-12-02	2017-02-11	30
30	2016-12-02	2017-01-20	30
31	2016-12-02	∅	30

DETERMINE THE RANGE OF MONTHS OF DATA PROVIDED.

WHICH MONTHS WILL YOU BE ABLE TO CALCULATE CHURN FOR?

I WILL BE ABLE TO CALCULATE THE CHURN RATE FOR JANUARY, FEBRUARY AND MARCH.



```
SELECT MIN(subscription_start), MAX(subscription_end)
FROM subscriptions;
```

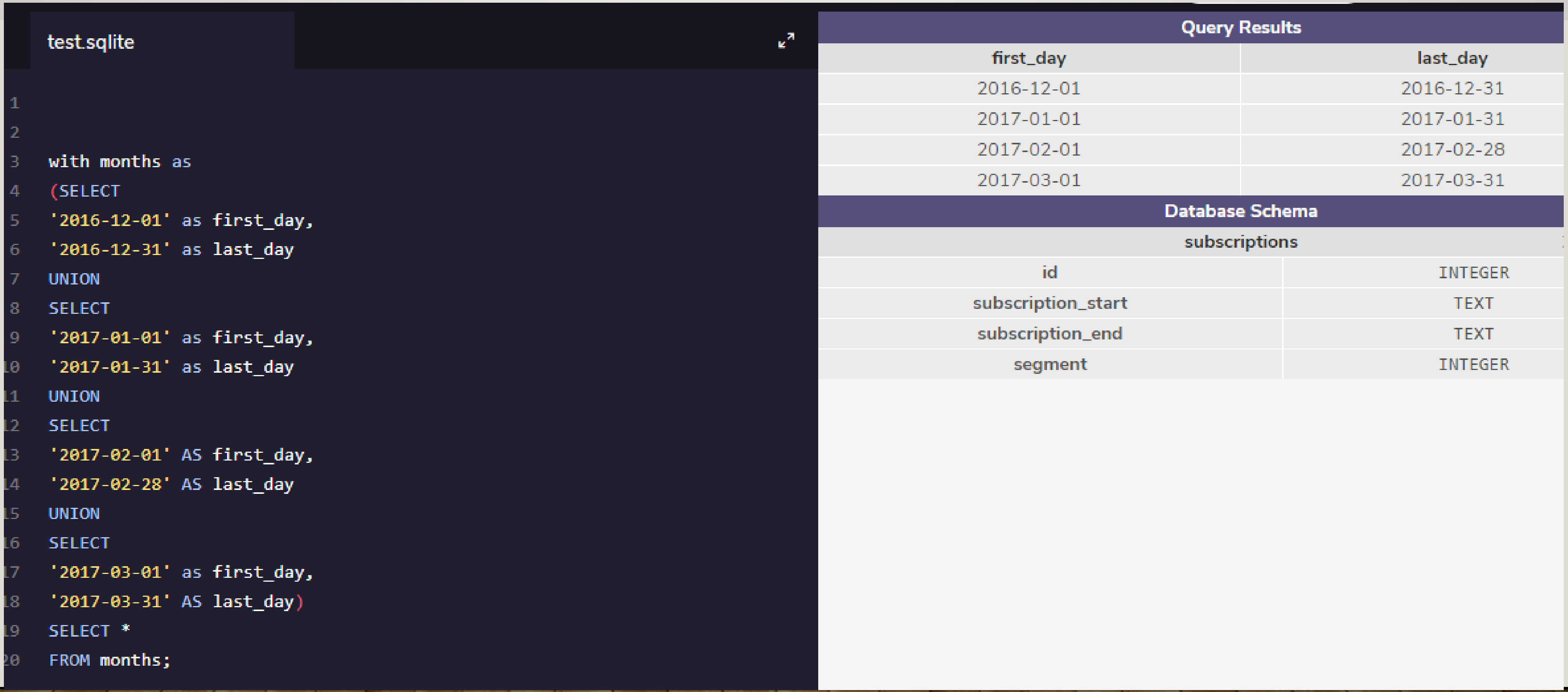
MIN(subscription_start)	MAX(subscription_end)
2016-12-01	2017-03-31

subscriptions 2000 rows

id	INTEGER

CALCULATE CHURN RATE FOR EACH SEGMENT

YOU'LL BE CALCULATING THE CHURN RATE FOR BOTH SEGMENTS (87 AND 30) OVER THE FIRST 3 MONTHS OF 2017 (YOU CAN'T CALCULATE IT FOR DECEMBER, SINCE THERE ARE NO SUBSCRIPTION END VALUES YET). TO GET STARTED, CREATE A TEMPORARY TABLE OF MONTHS.



test.sqlite

```


with months as
(SELECT
'2016-12-01' as first_day,
'2016-12-31' as last_day
UNION
SELECT
'2017-01-01' as first_day,
'2017-01-31' as last_day
UNION
SELECT
'2017-02-01' AS first_day,
'2017-02-28' AS last_day
UNION
SELECT
'2017-03-01' as first_day,
'2017-03-31' AS last_day)
SELECT *
FROM months;
```

Query Results

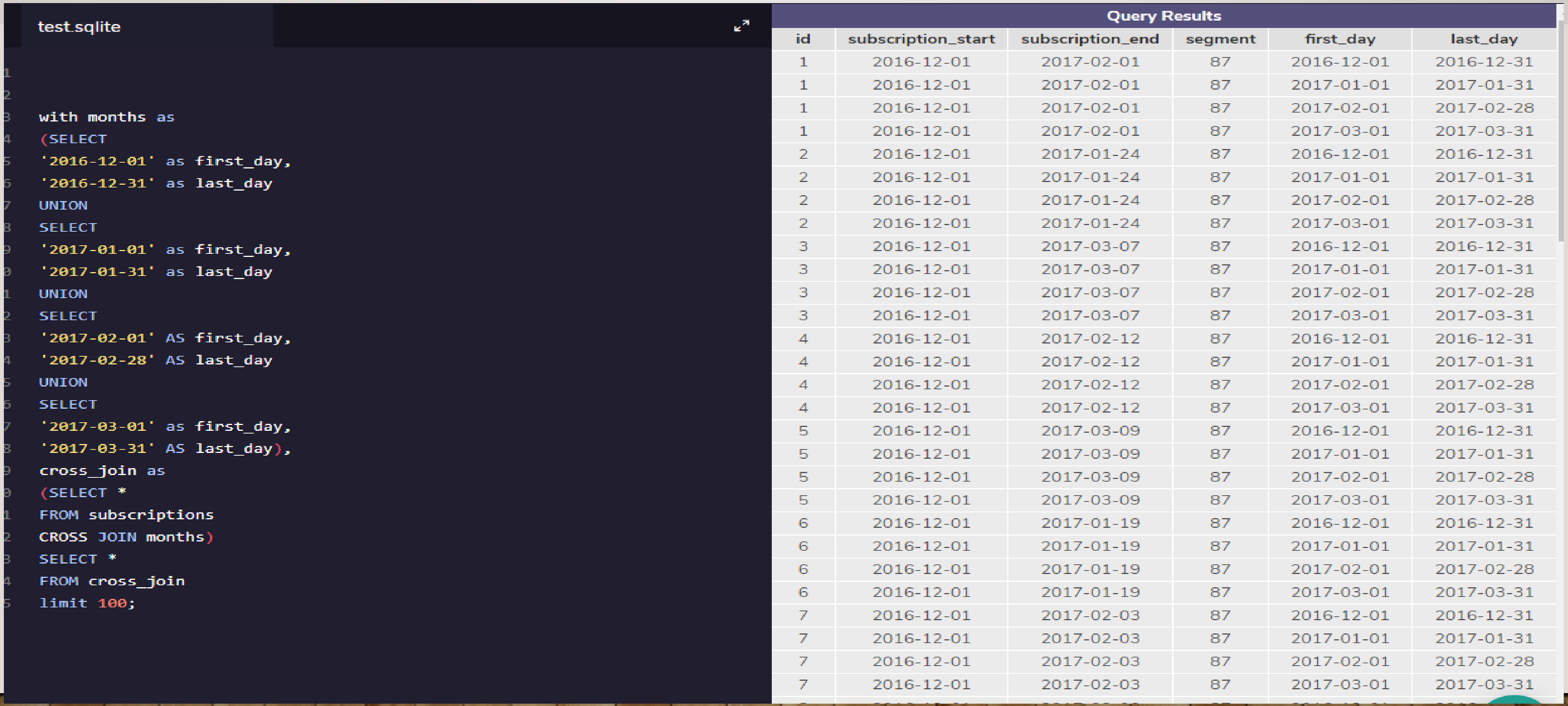
first_day	last_day
2016-12-01	2016-12-31
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

Database Schema

subscriptions

id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

CREATE A TEMPORARY TABLE, CROSS_JOIN, FROM SUBSCRIPTIONS AND YOUR MONTHS. BE SURE TO SELECT EVERY COLUMN.


test.sqlite

```
with months as
(SELECT
'2016-12-01' as first_day,
'2016-12-31' as last_day
UNION
SELECT
'2017-01-01' as first_day,
'2017-01-31' as last_day
UNION
SELECT
'2017-02-01' AS first_day,
'2017-02-28' AS last_day
UNION
SELECT
'2017-03-01' as first_day,
'2017-03-31' AS last_day),
cross_join as
(SELECT *
FROM subscriptions
CROSS JOIN months)
SELECT *
FROM cross_join
limit 100;
```

Query Results

id	subscription_start	subscription_end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2016-12-01	2016-12-31
1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
1	2016-12-01	2017-02-01	87	2017-02-01	2017-02-28
1	2016-12-01	2017-02-01	87	2017-03-01	2017-03-31
2	2016-12-01	2017-01-24	87	2016-12-01	2016-12-31
2	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31
3	2016-12-01	2017-03-07	87	2016-12-01	2016-12-31
3	2016-12-01	2017-03-07	87	2017-01-01	2017-01-31
3	2016-12-01	2017-03-07	87	2017-02-01	2017-02-28
3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31
4	2016-12-01	2017-02-12	87	2016-12-01	2016-12-31
4	2016-12-01	2017-02-12	87	2017-01-01	2017-01-31
4	2016-12-01	2017-02-12	87	2017-02-01	2017-02-28
4	2016-12-01	2017-02-12	87	2017-03-01	2017-03-31
5	2016-12-01	2017-03-09	87	2016-12-01	2016-12-31
5	2016-12-01	2017-03-09	87	2017-01-01	2017-01-31
5	2016-12-01	2017-03-09	87	2017-02-01	2017-02-28
5	2016-12-01	2017-03-09	87	2017-03-01	2017-03-31
6	2016-12-01	2017-01-19	87	2016-12-01	2016-12-31
6	2016-12-01	2017-01-19	87	2017-01-01	2017-01-31
6	2016-12-01	2017-01-19	87	2017-02-01	2017-02-28
6	2016-12-01	2017-01-19	87	2017-03-01	2017-03-31
7	2016-12-01	2017-02-03	87	2016-12-01	2016-12-31
7	2016-12-01	2017-02-03	87	2017-01-01	2017-01-31
7	2016-12-01	2017-02-03	87	2017-02-01	2017-02-28
7	2016-12-01	2017-02-03	87	2017-03-01	2017-03-31

CREATE A TEMPORARY TABLE, STATUS, FROM THE CROSS_JOIN TABLE YOU CREATED.

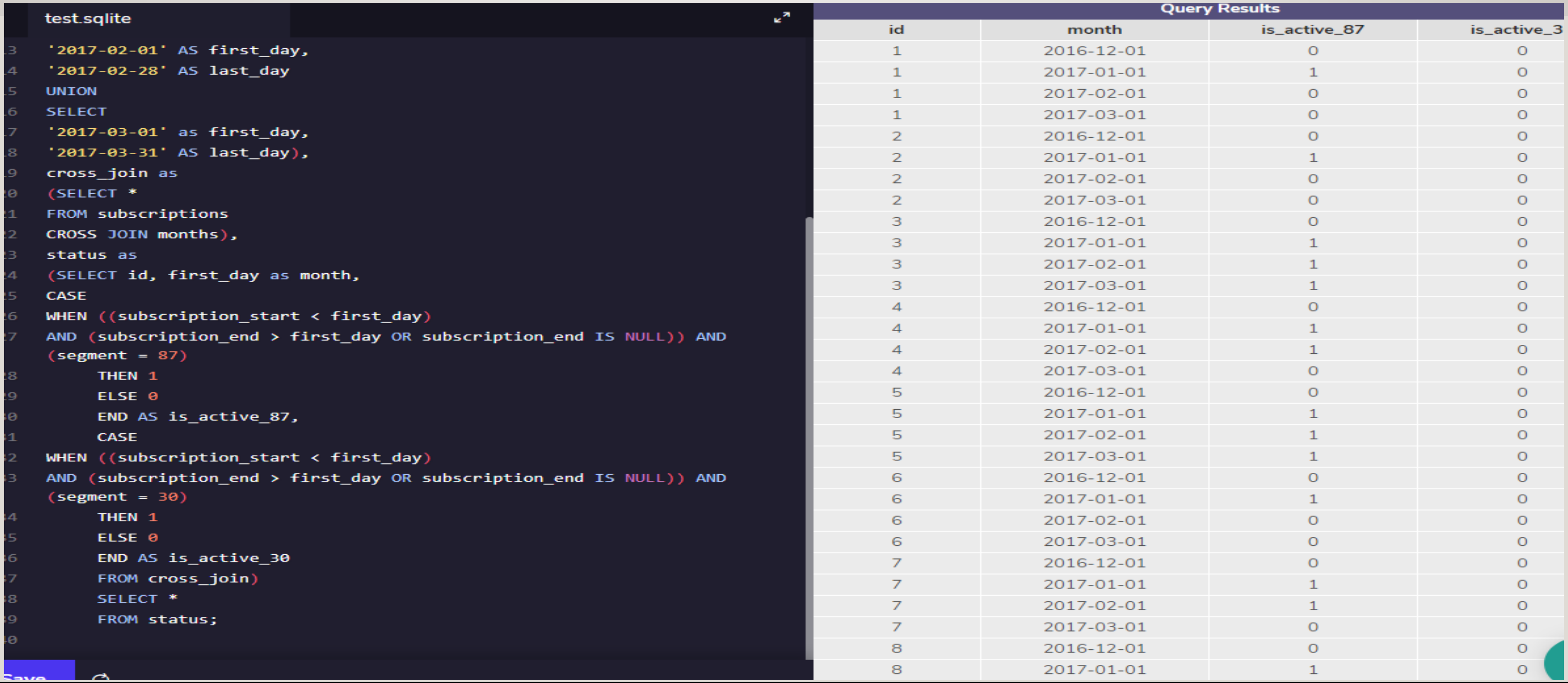
THIS TABLE SHOULD CONTAIN:

•ID SELECTED FROM CROSS_JOIN

•MONTH AS AN ALIAS OF FIRST_DAY

•IS_ACTIVE_87 CREATED USING A CASE WHEN TO FIND ANY USERS FROM SEGMENT 87 WHO EXISTED PRIOR TO THE BEGINNING OF THE MONTH. THIS IS 1 IF TRUE AND 0 OTHERWISE.

•IS_ACTIVE_30 CREATED USING A CASE WHEN TO FIND ANY USERS FROM SEGMENT 30 WHO EXISTED PRIOR TO THE BEGINNING OF THE MONTH. THIS IS 1 IF TRUE AND 0 OTHERWISE



test.sqlite

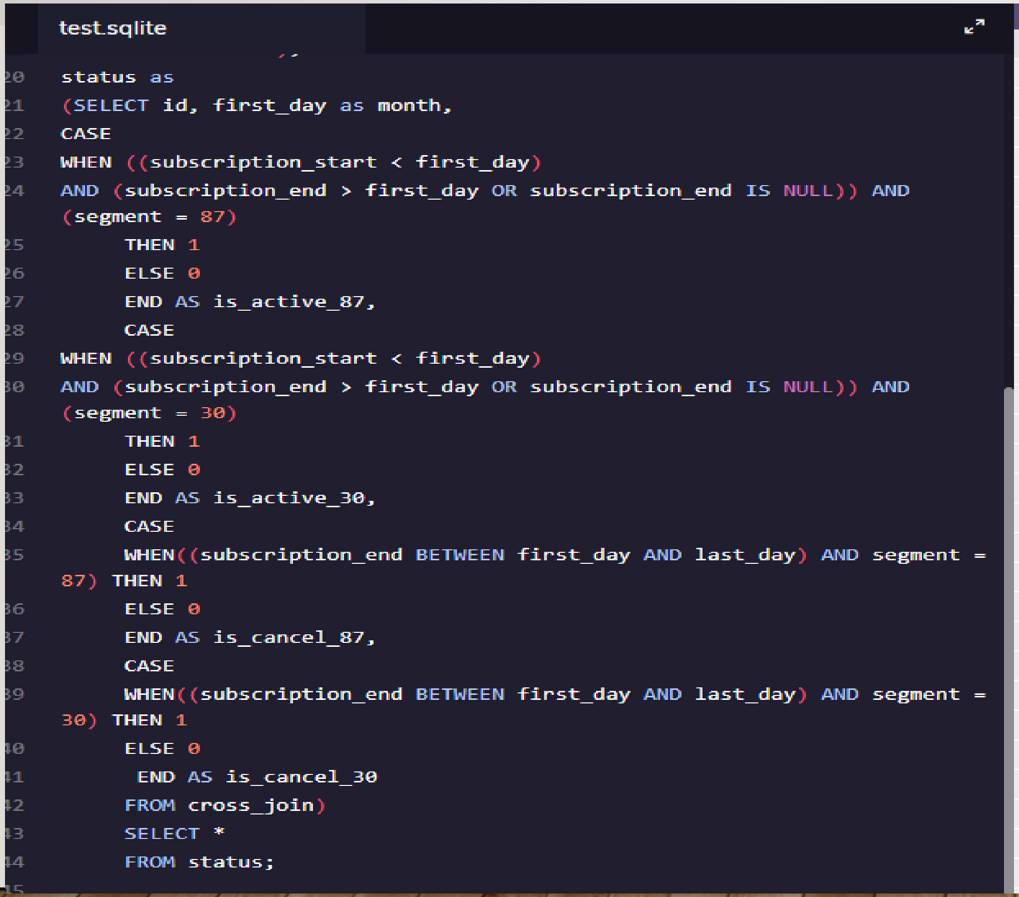
```
'2017-02-01' AS first_day,
'2017-02-28' AS last_day
UNION
SELECT
'2017-03-01' as first_day,
'2017-03-31' AS last_day),
cross_join as
(SELECT *
FROM subscriptions
CROSS JOIN months),
status as
(SELECT id, first_day as month,
CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 87)
     THEN 1
     ELSE 0
     END AS is_active_87,
     CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 30)
     THEN 1
     ELSE 0
     END AS is_active_30
     FROM cross_join)
     SELECT *
     FROM status;
```

Query Results

id	month	is_active_87	is_active_3
1	2016-12-01	0	0
1	2017-01-01	1	0
1	2017-02-01	0	0
1	2017-03-01	0	0
2	2016-12-01	0	0
2	2017-01-01	1	0
2	2017-02-01	0	0
2	2017-03-01	0	0
3	2016-12-01	0	0
3	2017-01-01	1	0
3	2017-02-01	1	0
3	2017-03-01	1	0
4	2016-12-01	0	0
4	2017-01-01	1	0
4	2017-02-01	1	0
4	2017-03-01	0	0
5	2016-12-01	0	0
5	2017-01-01	1	0
5	2017-02-01	1	0
5	2017-03-01	1	0
6	2016-12-01	0	0
6	2017-01-01	1	0
6	2017-02-01	0	0
6	2017-03-01	0	0
7	2016-12-01	0	0
7	2017-01-01	1	0
7	2017-02-01	1	0
7	2017-03-01	0	0
8	2016-12-01	0	0
8	2017-01-01	1	0

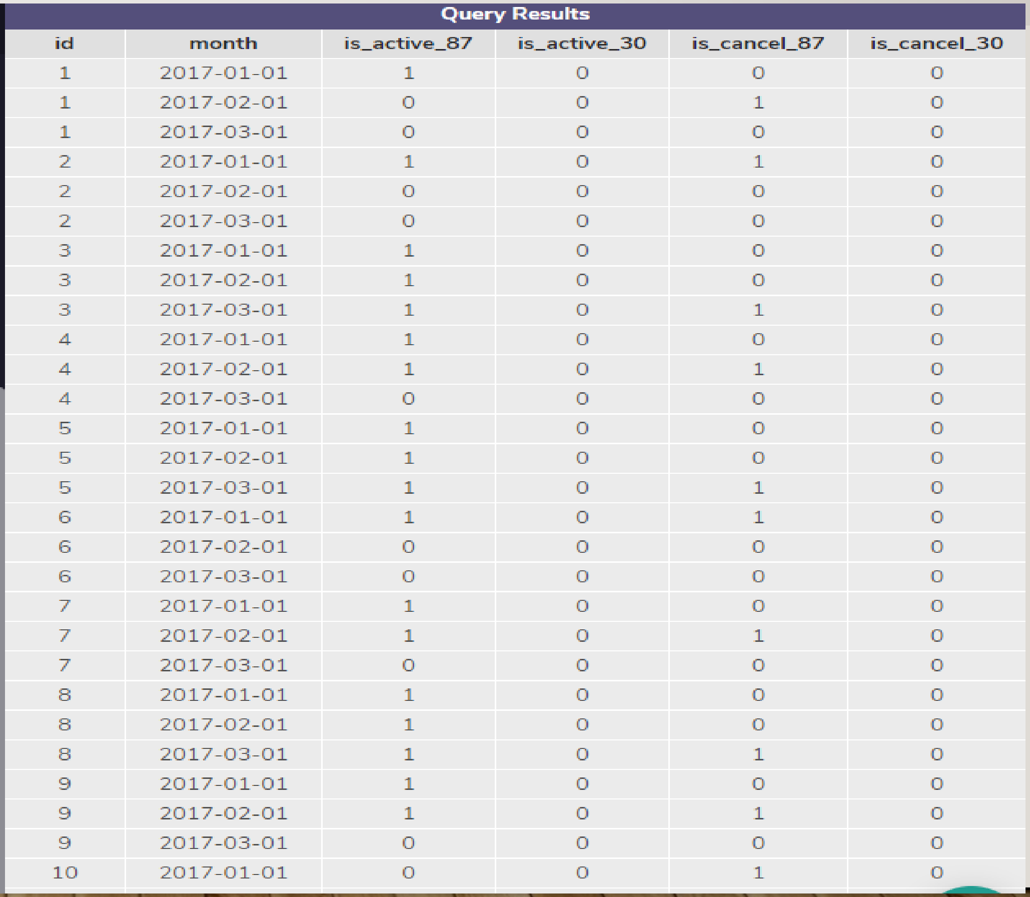
ADD AN IS_CANCELED_87 AND AN IS_CANCELED_30 COLUMN TO THE STATUS TEMPORARY TABLE.

THIS SHOULD BE 1 IF THE SUBSCRIPTION IS CANCELED DURING THE MONTH AND 0 OTHERWISE



test.sqlite

```sql
status as
(SELECT id, first_day as month,
CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 87)
     THEN 1
     ELSE 0
     END AS is_active_87,
     CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 30)
     THEN 1
     ELSE 0
     END AS is_active_30,
     CASE
     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
87) THEN 1
     ELSE 0
     END AS is_cancel_87,
     CASE
     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
30) THEN 1
     ELSE 0
      END AS is_cancel_30
     FROM cross_join)
     SELECT *
     FROM status;
```

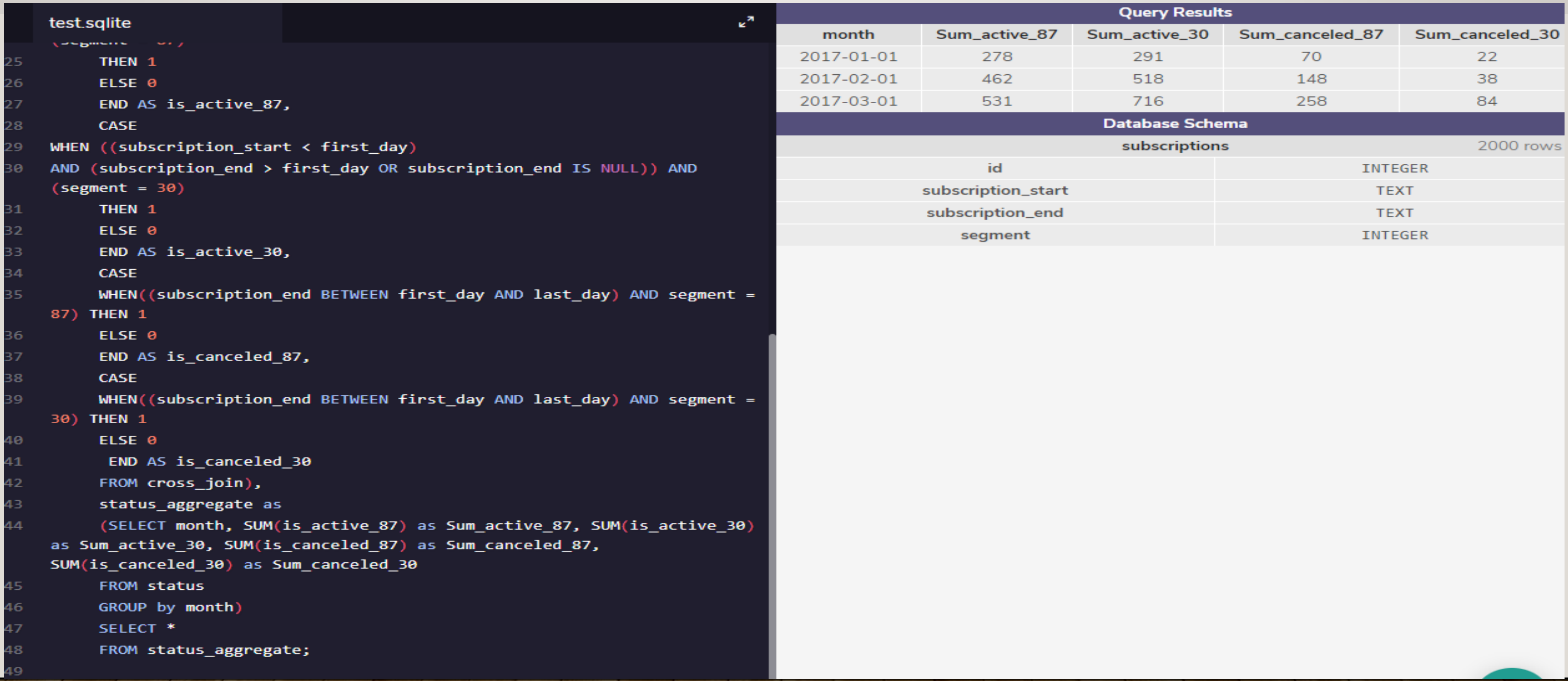


Query Results

id	month	is_active_87	is_active_30	is_cancel_87	is_cancel_30
1	2017-01-01	1	0	0	0
1	2017-02-01	0	0	1	0
1	2017-03-01	0	0	0	0
2	2017-01-01	1	0	1	0
2	2017-02-01	0	0	0	0
2	2017-03-01	0	0	0	0
3	2017-01-01	1	0	0	0
3	2017-02-01	1	0	0	0
3	2017-03-01	1	0	1	0
4	2017-01-01	1	0	0	0
4	2017-02-01	1	0	1	0
4	2017-03-01	0	0	0	0
5	2017-01-01	1	0	0	0
5	2017-02-01	1	0	0	0
5	2017-03-01	1	0	1	0
6	2017-01-01	1	0	1	0
6	2017-02-01	0	0	0	0
6	2017-03-01	0	0	0	0
7	2017-01-01	1	0	0	0
7	2017-02-01	1	0	1	0
7	2017-03-01	0	0	0	0
8	2017-01-01	1	0	0	0
8	2017-02-01	1	0	0	0
8	2017-03-01	1	0	1	0
9	2017-01-01	1	0	0	0
9	2017-02-01	1	0	1	0
9	2017-03-01	0	0	0	0
10	2017-01-01	0	0	1	0

CREATE A STATUS AGGREGATE TEMPORARY TABLE THAT IS A SUM OF THE ACTIVE AND CANCELED SUBSCRIPTIONS FOR EACH SEGMENT, FOR EACH MONTH.

THE RESULTING COLUMNS SHOULD BE:
SUM_ACTIVE_87
SUM_ACTIVE_30
SUM_CANCELED_87
SUM_CANCELED_30



test.sqlite

```
     THEN 1
     ELSE 0
     END AS is_active_87,
     CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 30)
     THEN 1
     ELSE 0
     END AS is_active_30,
     CASE
     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
87) THEN 1
     ELSE 0
     END AS is_canceled_87,
     CASE
     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
30) THEN 1
     ELSE 0
      END AS is_canceled_30
     FROM cross_join),
     status_aggregate as
     (SELECT month, SUM(is_active_87) as Sum_active_87, SUM(is_active_30)
as Sum_active_30, SUM(is_canceled_87) as Sum_canceled_87,
SUM(is_canceled_30) as Sum_canceled_30
     FROM status
     GROUP by month)
     SELECT *
     FROM status_aggregate;
```

Query Results

month	Sum_active_87	Sum_active_30	Sum_canceled_87	Sum_canceled_30
2017-01-01	278	291	70	22
2017-02-01	462	518	148	38
2017-03-01	531	716	258	84

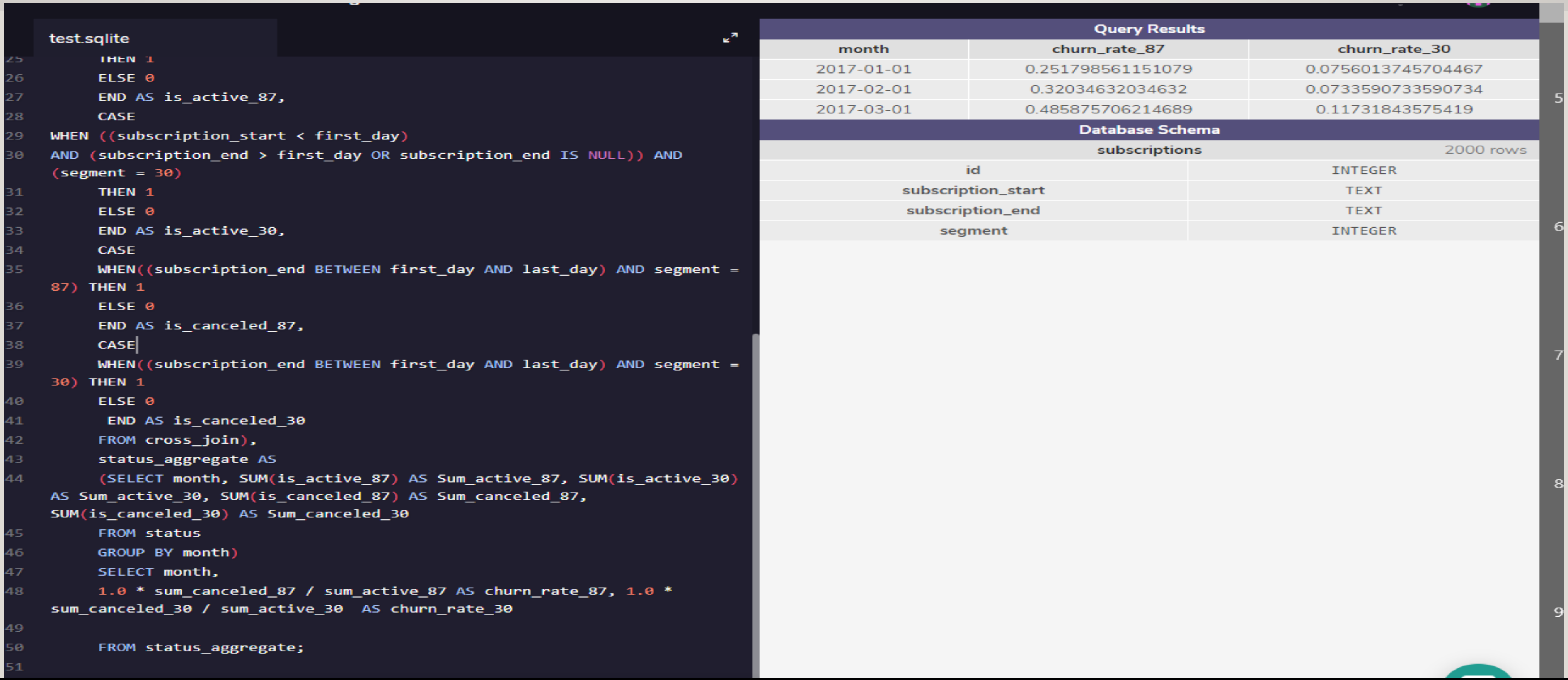
Database Schema

subscriptions 2000 rows

id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

CALCULATE THE CHURN RATES FOR THE TWO SEGMENTS OVER THE THREE MONTH PERIOD.

WHICH SEGMENT HAS A LOWER CHURN RATE?
THE 31 SEGMENT HAS A LOWER CHURN RATE



test.sqlite

```
      THEN 1
      ELSE 0
      END AS is_active_87,
      CASE
WHEN ((subscription_start < first_day)
AND (subscription_end > first_day OR subscription_end IS NULL)) AND
(segment = 30)
      THEN 1
      ELSE 0
      END AS is_active_30,
      CASE
      WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
87) THEN 1
      ELSE 0
      END AS is_canceled_87,
      CASE
      WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
30) THEN 1
      ELSE 0
       END AS is_canceled_30
      FROM cross_join),
      status_aggregate AS
      (SELECT month, SUM(is_active_87) AS Sum_active_87, SUM(is_active_30)
AS Sum_active_30, SUM(is_canceled_87) AS Sum_canceled_87,
SUM(is_canceled_30) AS Sum_canceled_30
      FROM status
      GROUP BY month)
      SELECT month,
      1.0 * sum_canceled_87 / sum_active_87 AS churn_rate_87, 1.0 *
sum_canceled_30 / sum_active_30  AS churn_rate_30

      FROM status_aggregate;
```

Query Results

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

Database Schema

subscriptions 2000 rows

id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER