Final Project: Milestone 4

CS 327E

April 16, 2018

Announcements:

Today: Last regular class.
Today: Last quiz.
Next Friday: Demo Day in WAG 420. Schedule link.
Discuss: Milestones 5 and 6 guidelines.

1) What makes traditional MapReduce suitable for batch processing?

A) The inputs to the Mapper are bounded / finite.

B) The inputs to the Reducer are bounded / finite.

C) The job is run at fixed time slices (e.g. now, hourly, daily, etc.)

D) All of the above.

2) What is the one **crucial** difference between a batch job and a streaming job?

A) The batch job processes larger collections of data.

B) The batch job goes through a multi-stage pipeline.

C) The event stream never ends.

D) None of the above.

3) Consider the Star Wars movies and their release timeline. The episode number is equivalent to __________ whereas the release year is equivalent to ____________.

Episode IV:	1977
Episode V:	1980
Episode VI:	1983
Episode I:	1999
Episode II:	2002
Episode III:	2005

A) Processing time; Event time
B) Event time; Processing time

4) The paper discusses 3 types of windows: **Fixed**, **Sliding**, and **Sessions**. Which notion of time are these windows based on?

A) Event time

B) Processing time

C) Neither

5) The paper discusses 3 options for handling straggler events that arrive after the window has been declared complete: **Discarding**, **Accumulating**, and **Accumulating & Retracting**. Which option(s) require the consumer to handle updated results for the windows?

A) Discarding
B) Accumulating
C) Accumulating & Retracting
D) All of the above
E) Only B and C

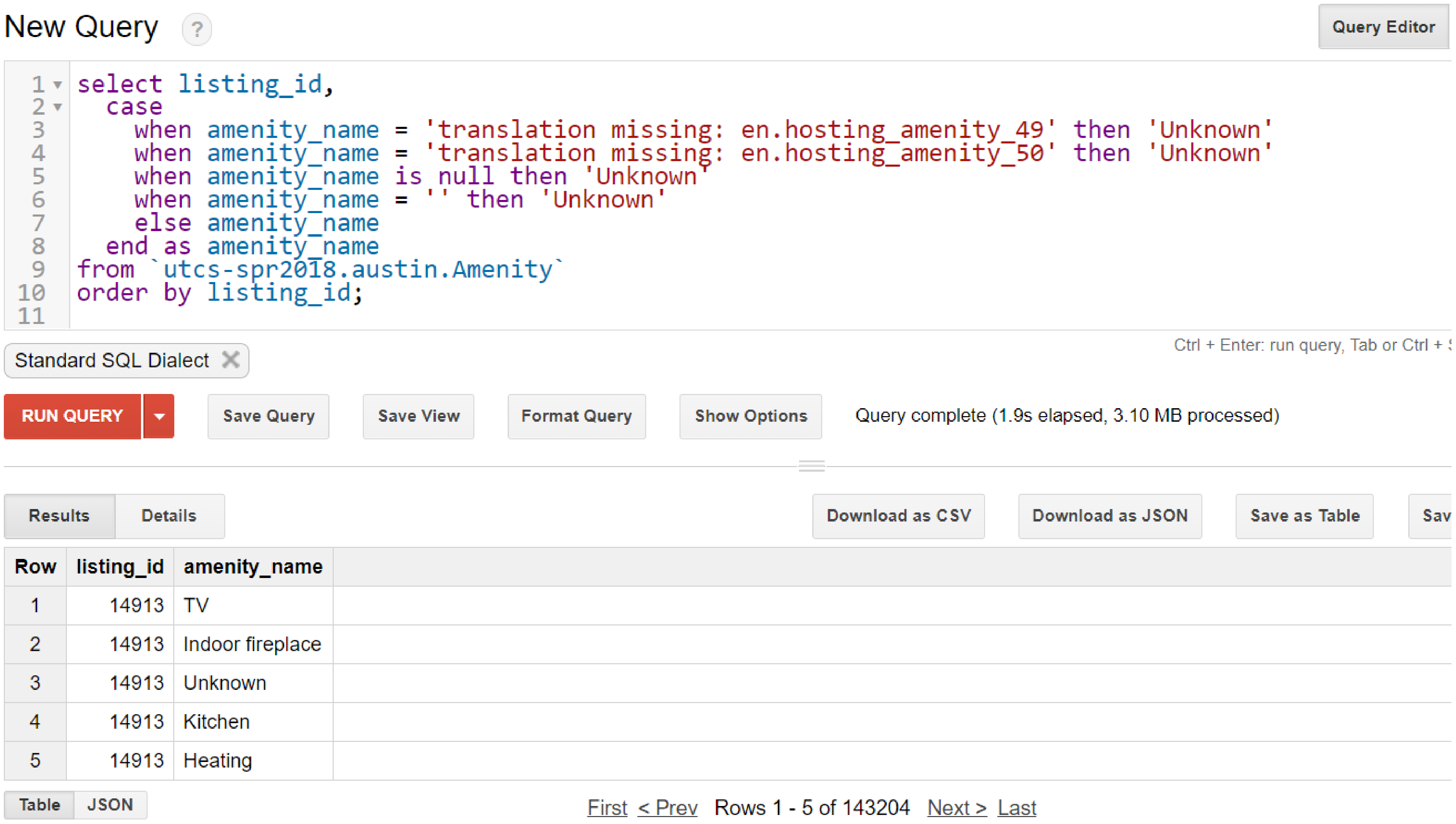
Case Expressions in SQL

- Conditional logic
- Since SQL:92 Standard
- Appear in SELECT clause
- Return scalar value for each record
- Return values of same type
- Used in SELECT statements
- Also used in UPDATE, INSERT, DELETE statements

General Form:

```
CASE
    WHEN c1 THEN e1
    WHEN c2 THEN e2
    …
    WHEN cn THEN en
    [ELSE ed]
END
```

Case Expression Example



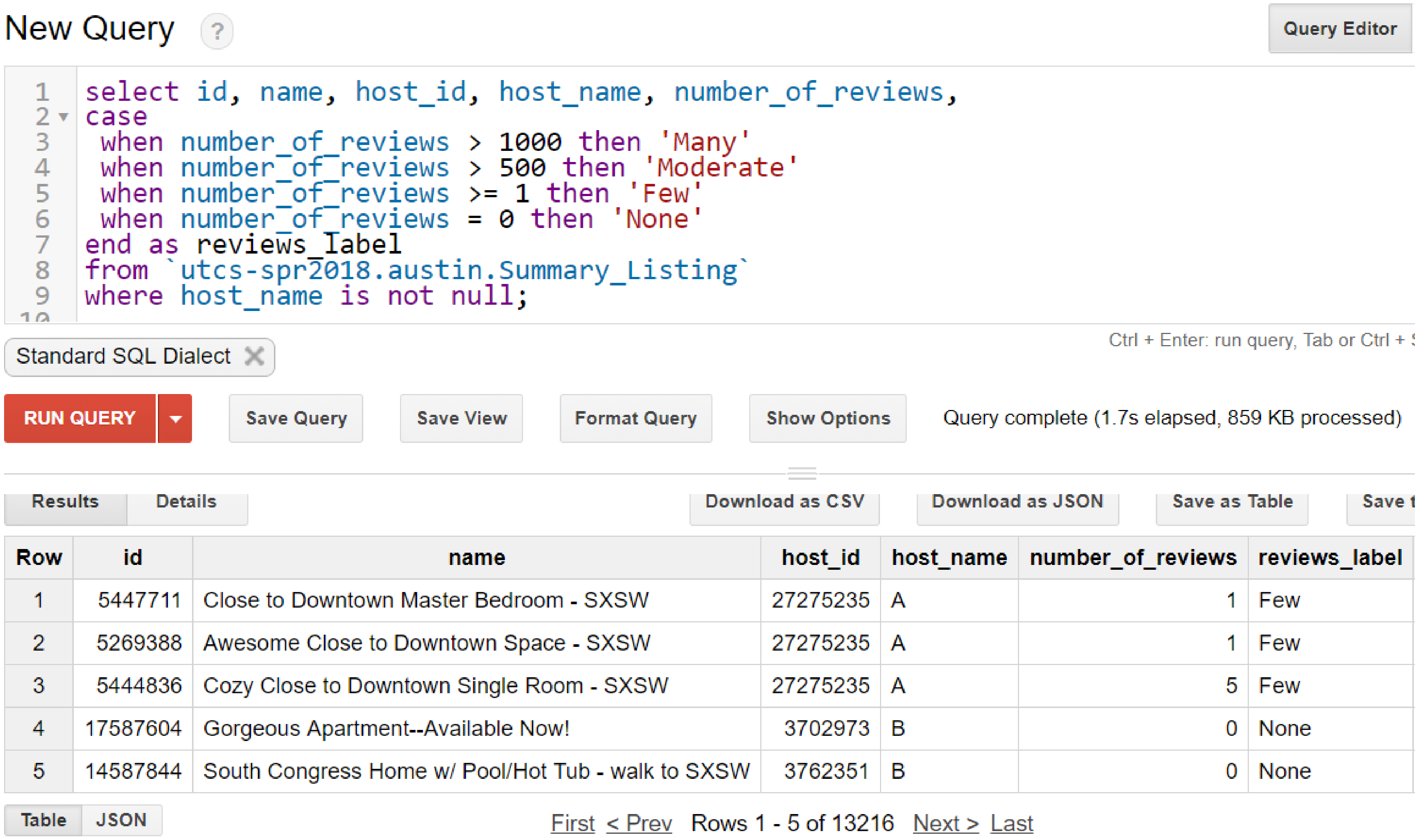
```
select listing_id,
  case
    when amenity_name = 'translation missing: en.hosting_amenity_49' then 'Unknown'
    when amenity_name = 'translation missing: en.hosting_amenity_50' then 'Unknown'
    when amenity_name is null then 'Unknown'
    when amenity_name = '' then 'Unknown'
    else amenity_name
  end as amenity_name
from `utcs-spr2018.austin.Amenity`
order by listing_id;
```

Query complete (1.9s elapsed, 3.10 MB processed)

Row	listing_id	amenity_name
1	14913	TV
2	14913	Indoor fireplace
3	14913	Unknown
4	14913	Kitchen
5	14913	Heating

Rows 1 - 5 of 143204

Another Case Expression Example



New Query

Query Editor

```
select id, name, host_id, host_name, number_of_reviews,
case
 when number_of_reviews > 1000 then 'Many'
 when number_of_reviews > 500 then 'Moderate'
 when number_of_reviews >= 1 then 'Few'
 when number_of_reviews = 0 then 'None'
end as reviews_label
from `utcs-spr2018.austin.Summary_Listing`
where host_name is not null;
```

Standard SQL Dialect

Ctrl + Enter: run query, Tab or Ctrl +

Query complete (1.7s elapsed, 859 KB processed)

Row	id	name	host_id	host_name	number_of_reviews	reviews_label
1	5447711	Close to Downtown Master Bedroom - SXSW	27275235	A	1	Few
2	5269388	Awesome Close to Downtown Space - SXSW	27275235	A	1	Few
3	5444836	Cozy Close to Downtown Single Room - SXSW	27275235	A	5	Few
4	17587604	Gorgeous Apartment--Available Now!	3702973	B	0	None
5	14587844	South Congress Home w/ Pool/Hot Tub - walk to SXSW	3762351	B	0	None

First < Prev Rows 1 - 5 of 13216 Next > Last

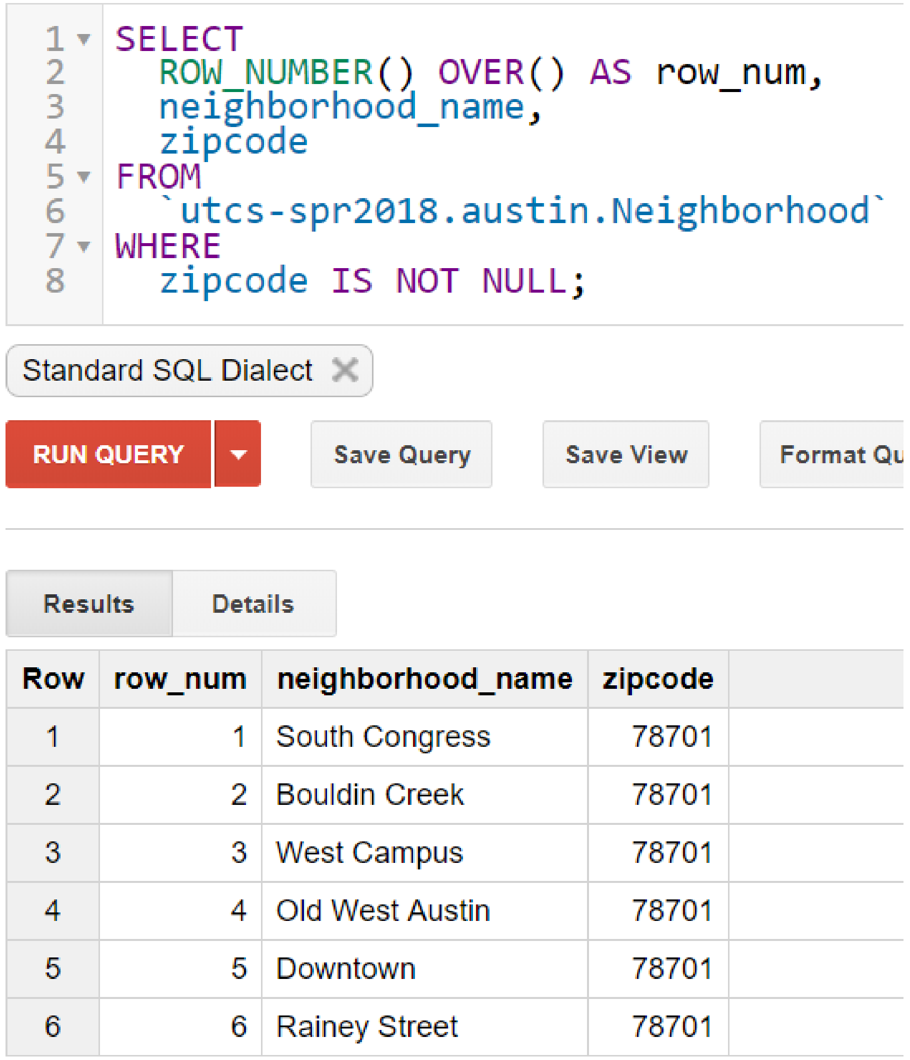
Window Clause in SQL

- Informally called the `OVER` clause
- Since SQL:2003 Standard
- Rows split into partitions with `PARTITION BY` predicate
- Rows are sorted within each partition with `ORDER BY` predicate
- Window function applied to each row within partition
- Example functions: `ROW_NUMBER()`, `RANK()`

General Form:

```
SELECT c1,
 f()
   OVER(
   [PARTITION BY c3
   ORDER BY c4]
   )
FROM T1
```

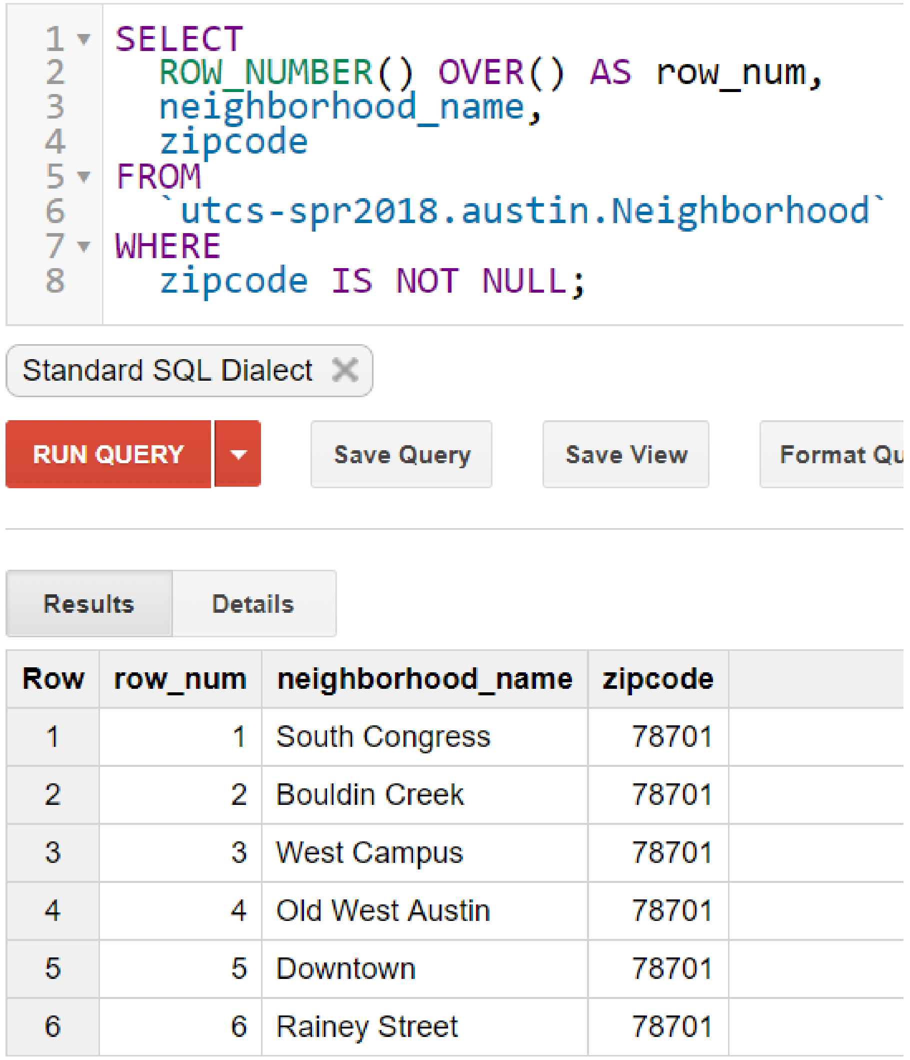
Window Example: ROW_NUMBER



```
SELECT
  ROW_NUMBER() OVER() AS row_num,
  neighborhood_name,
  zipcode
FROM
  `utcs-spr2018.austin.Neighborhood`
WHERE
  zipcode IS NOT NULL;
```

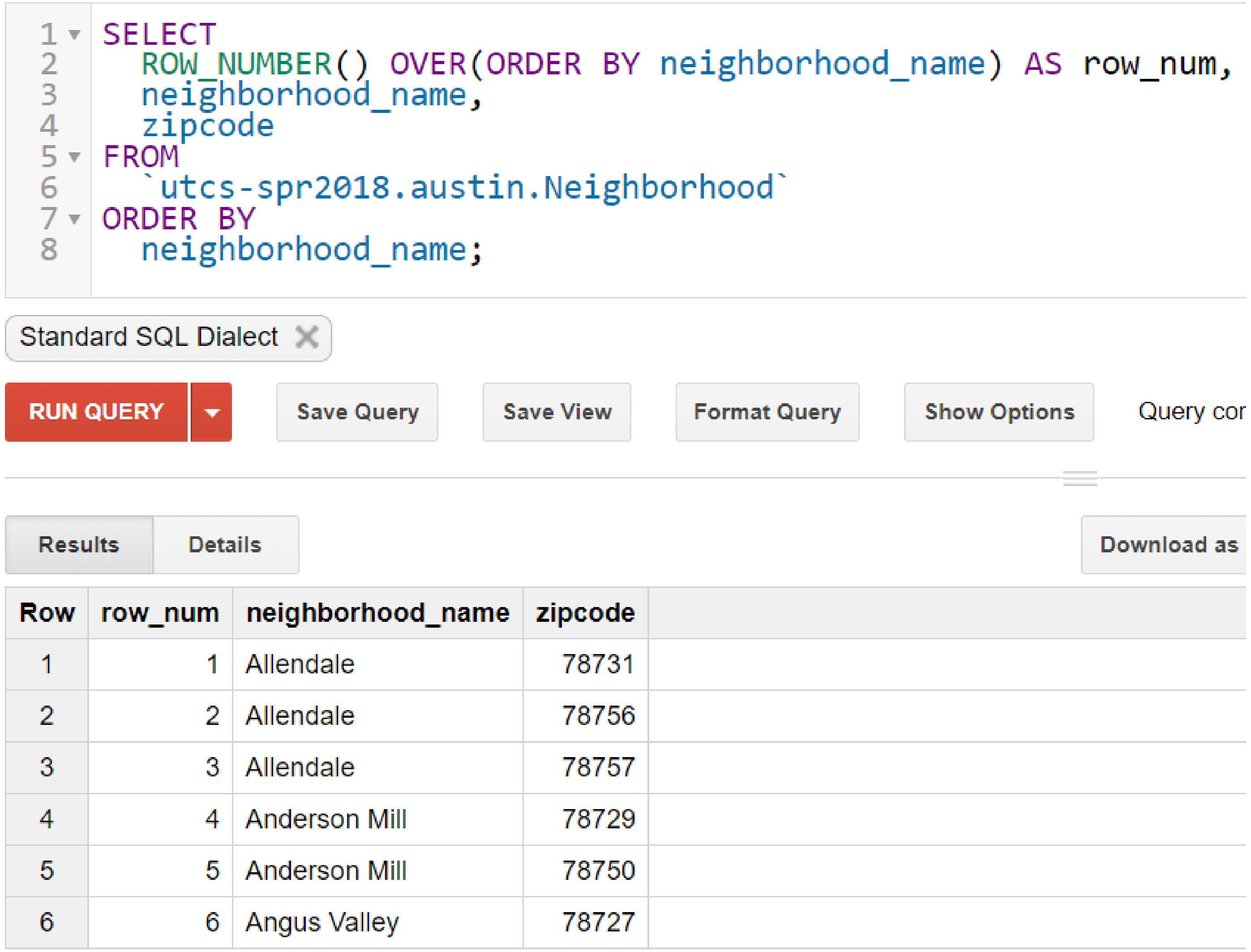
Row	row_num	neighborhood_name	zipcode
1	1	South Congress	78701
2	2	Bouldin Creek	78701
3	3	West Campus	78701
4	4	Old West Austin	78701
5	5	Downtown	78701
6	6	Rainey Street	78701

Window Example: ROW_NUMBER



```
SELECT
  ROW_NUMBER() OVER() AS row_num,
  neighborhood_name,
  zipcode
FROM
  `utcs-spr2018.austin.Neighborhood`
WHERE
  zipcode IS NOT NULL;
```

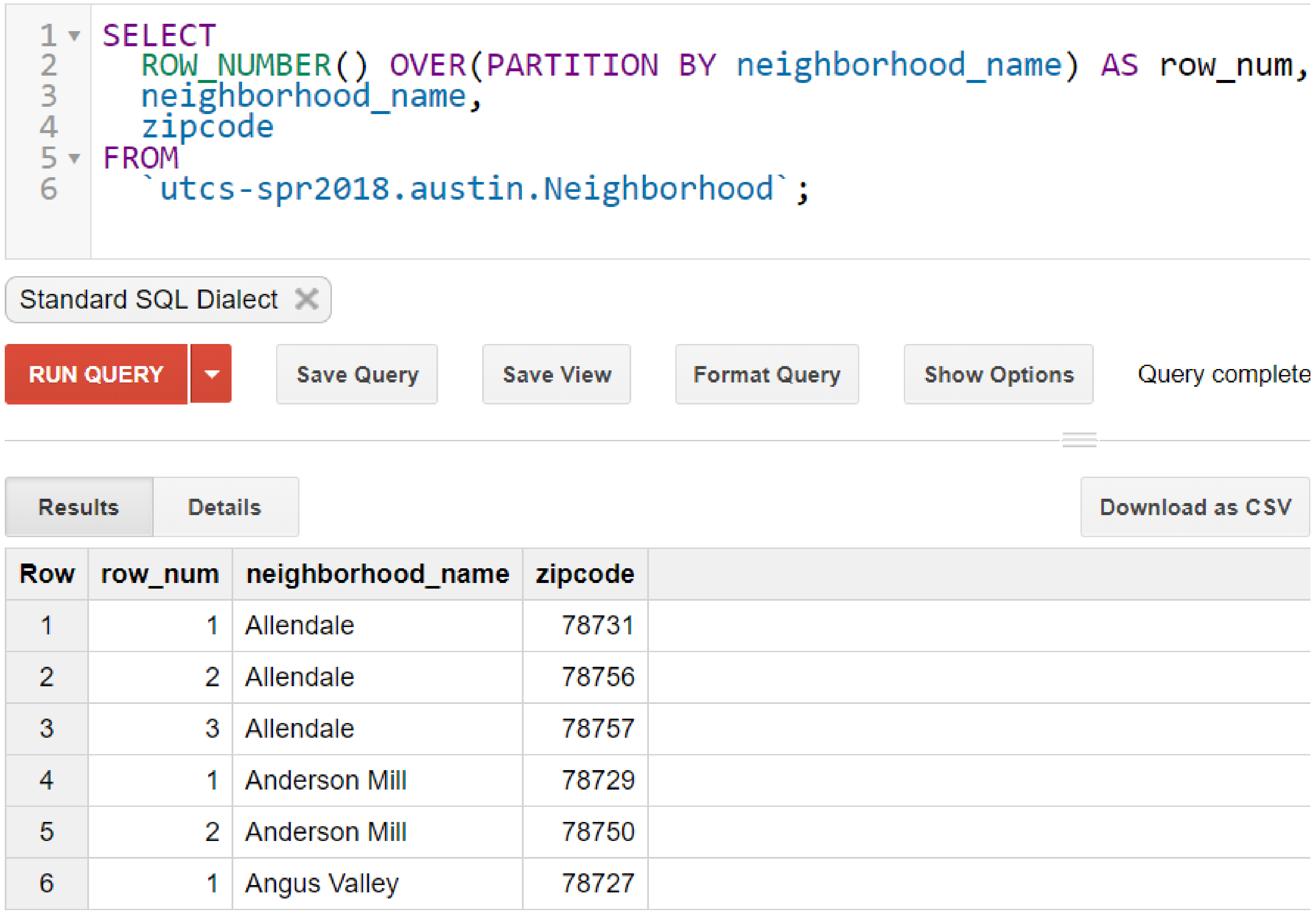
Row	row_num	neighborhood_name	zipcode
1	1	South Congress	78701
2	2	Bouldin Creek	78701
3	3	West Campus	78701
4	4	Old West Austin	78701
5	5	Downtown	78701
6	6	Rainey Street	78701



```
SELECT
  ROW_NUMBER() OVER(ORDER BY neighborhood_name) AS row_num,
  neighborhood_name,
  zipcode
FROM
  `utcs-spr2018.austin.Neighborhood`
ORDER BY
  neighborhood_name;
```

Row	row_num	neighborhood_name	zipcode
1	1	Allendale	78731
2	2	Allendale	78756
3	3	Allendale	78757
4	4	Anderson Mill	78729
5	5	Anderson Mill	78750
6	6	Angus Valley	78727

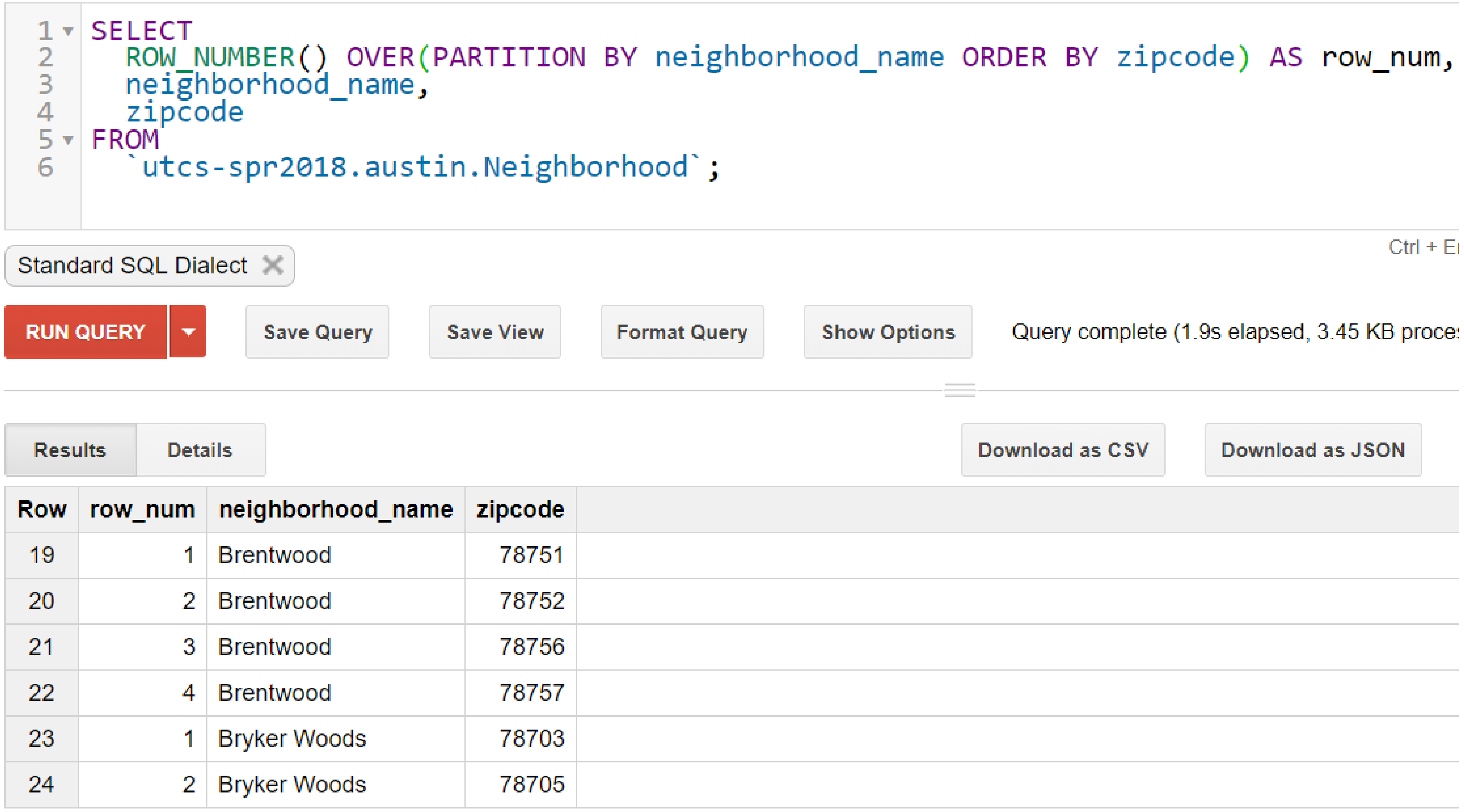
Window Example: ROW_NUMBER



```
SELECT
  ROW_NUMBER() OVER(PARTITION BY neighborhood_name) AS row_num,
  neighborhood_name,
  zipcode
FROM
  `utcs-spr2018.austin.Neighborhood`;
```

Row	row_num	neighborhood_name	zipcode
1	1	Allendale	78731
2	2	Allendale	78756
3	3	Allendale	78757
4	1	Anderson Mill	78729
5	2	Anderson Mill	78750
6	1	Angus Valley	78727

Window Example: ROW_NUMBER

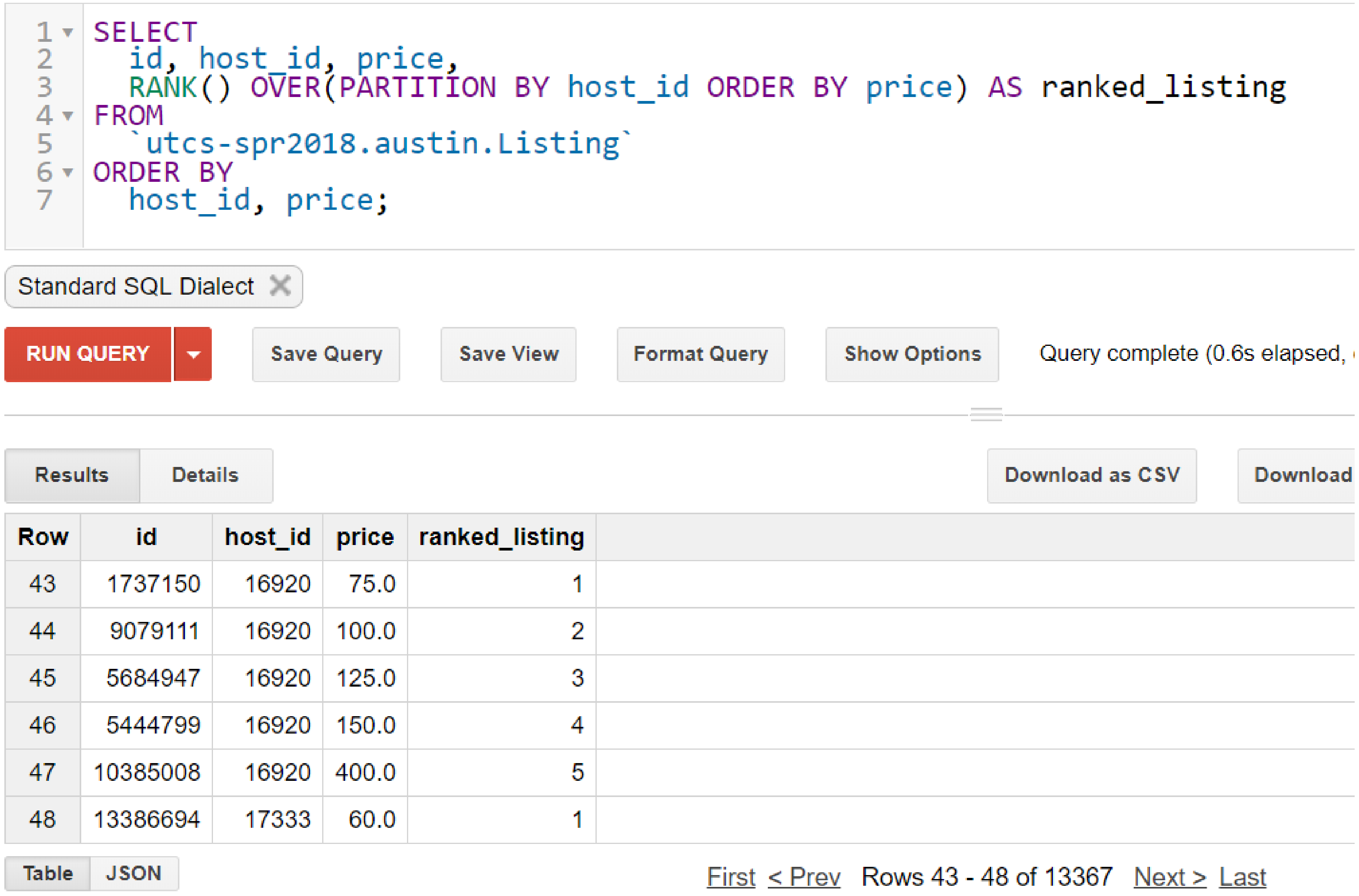


```
SELECT
  ROW_NUMBER() OVER(PARTITION BY neighborhood_name ORDER BY zipcode) AS row_num,
  neighborhood_name,
  zipcode
FROM
  `utcs-spr2018.austin.Neighborhood`;
```

Query complete (1.9s elapsed, 3.45 KB proce

Row	row_num	neighborhood_name	zipcode
19	1	Brentwood	78751
20	2	Brentwood	78752
21	3	Brentwood	78756
22	4	Brentwood	78757
23	1	Bryker Woods	78703
24	2	Bryker Woods	78705

Window Example: RANK

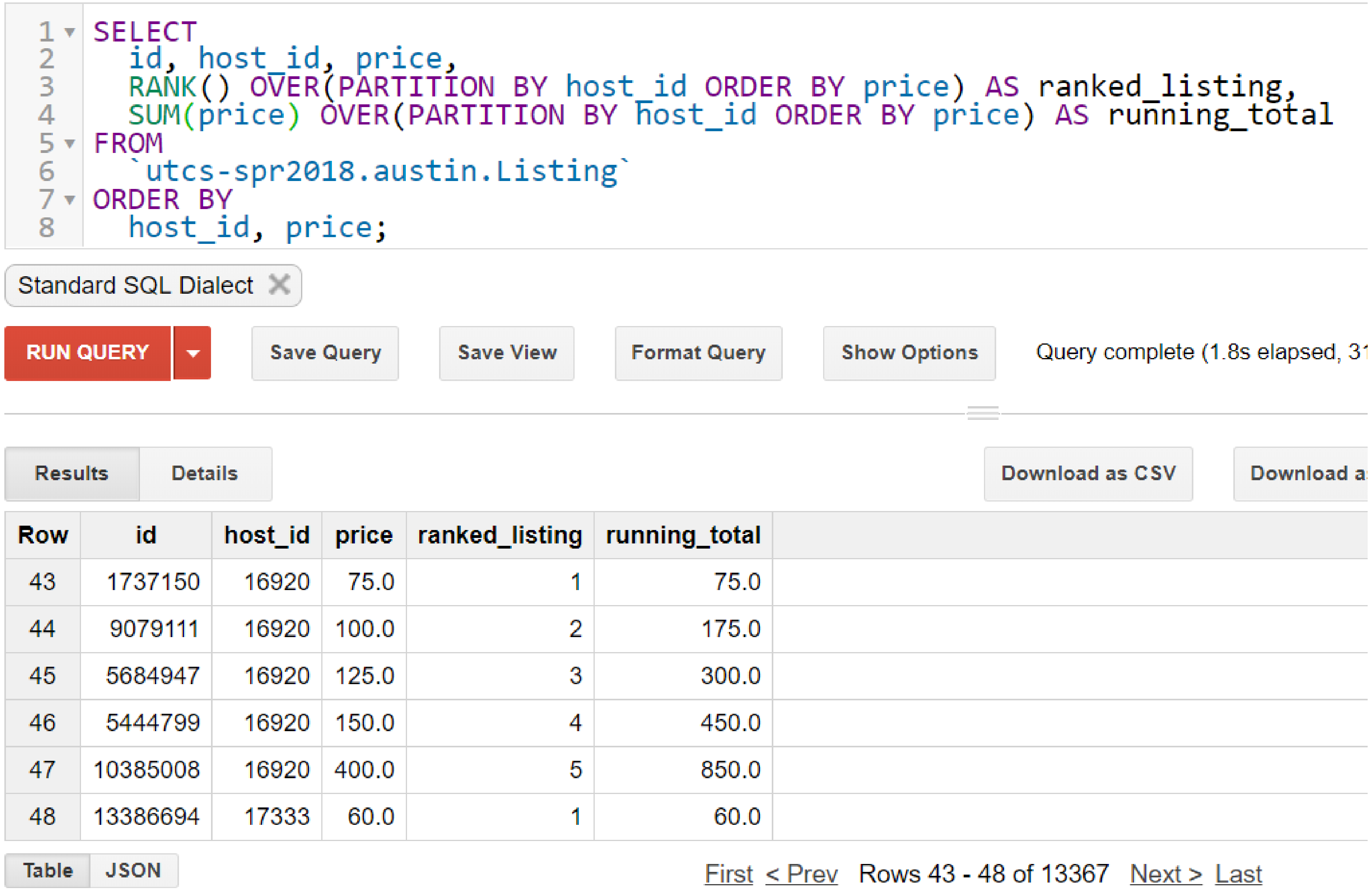


```
SELECT
  id, host_id, price,
  RANK() OVER(PARTITION BY host_id ORDER BY price) AS ranked_listing
FROM
  `utcs-spr2018.austin.Listing`
ORDER BY
  host_id, price;
```

Row	id	host_id	price	ranked_listing
43	1737150	16920	75.0	1
44	9079111	16920	100.0	2
45	5684947	16920	125.0	3
46	5444799	16920	150.0	4
47	10385008	16920	400.0	5
48	13386694	17333	60.0	1

First < Prev Rows 43 - 48 of 13367 Next > Last

Window Example: RANK and SUM



```
SELECT
  id, host_id, price,
  RANK() OVER(PARTITION BY host_id ORDER BY price) AS ranked_listing,
  SUM(price) OVER(PARTITION BY host_id ORDER BY price) AS running_total
FROM
  `utcs-spr2018.austin.Listing`
ORDER BY
  host_id, price;
```

Query complete (1.8s elapsed, 31

Row	id	host_id	price	ranked_listing	running_total
43	1737150	16920	75.0	1	75.0
44	9079111	16920	100.0	2	175.0
45	5684947	16920	125.0	3	300.0
46	5444799	16920	150.0	4	450.0
47	10385008	16920	400.0	5	850.0
48	13386694	17333	60.0	1	60.0

Rows 43 - 48 of 13367

Final Project Milestone 4

Cross-Dataset Joins:
http://www.cs.utexas.edu/~scohen/project/fp_guidelines.pdf