**FIT5137 Individual Assignment Report**

**Semester 2, 2019**

**Students Name: Dawei Gu**

**Students ID: 29910226**

**Tutorial Section: FIT5137 Laboratory 05,**

**12:00pm-14:00pm Tue**

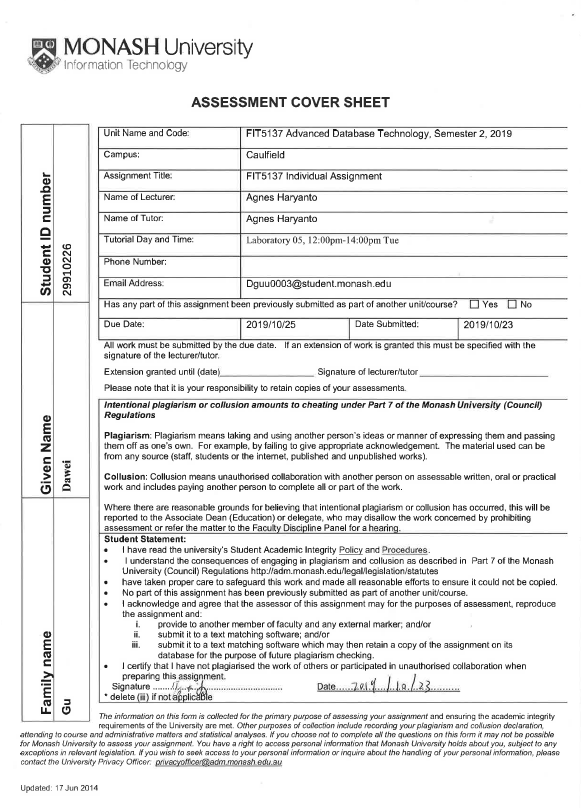
**Tutor: Agnes Haryanto**

**Contents**

1. **Signed Cover Sheet………………………………………….…… 2**
2. **Report**

* **C.1 Database Design………………………………………………. 3**
* **C.2 Queries………………………………………………………… 6**
* **C.3 Database Modifications……….………………………..…… 17**
* **C.4 Advanced Topic………………………………………….…... 23**

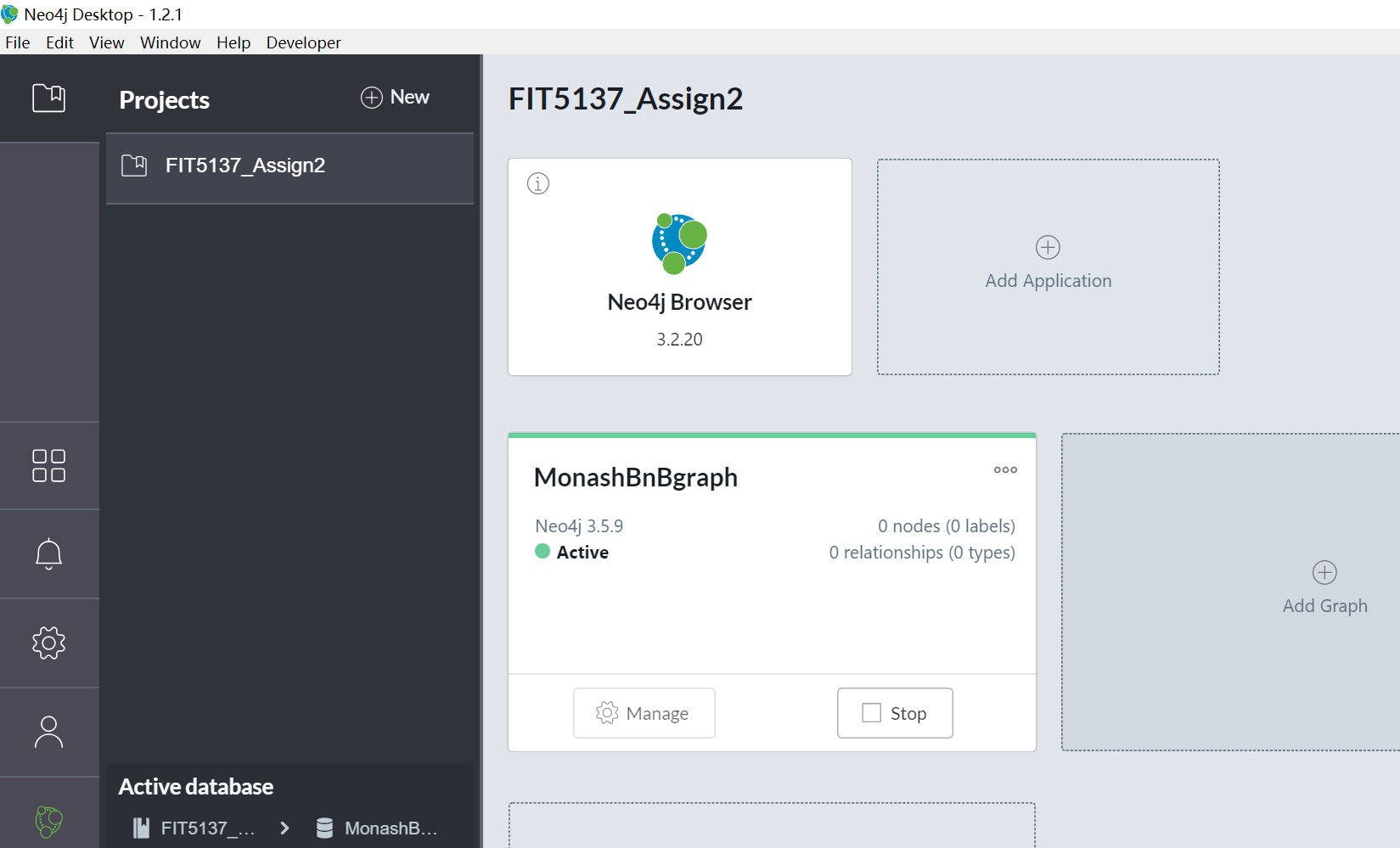
**Signed Cover Sheet**



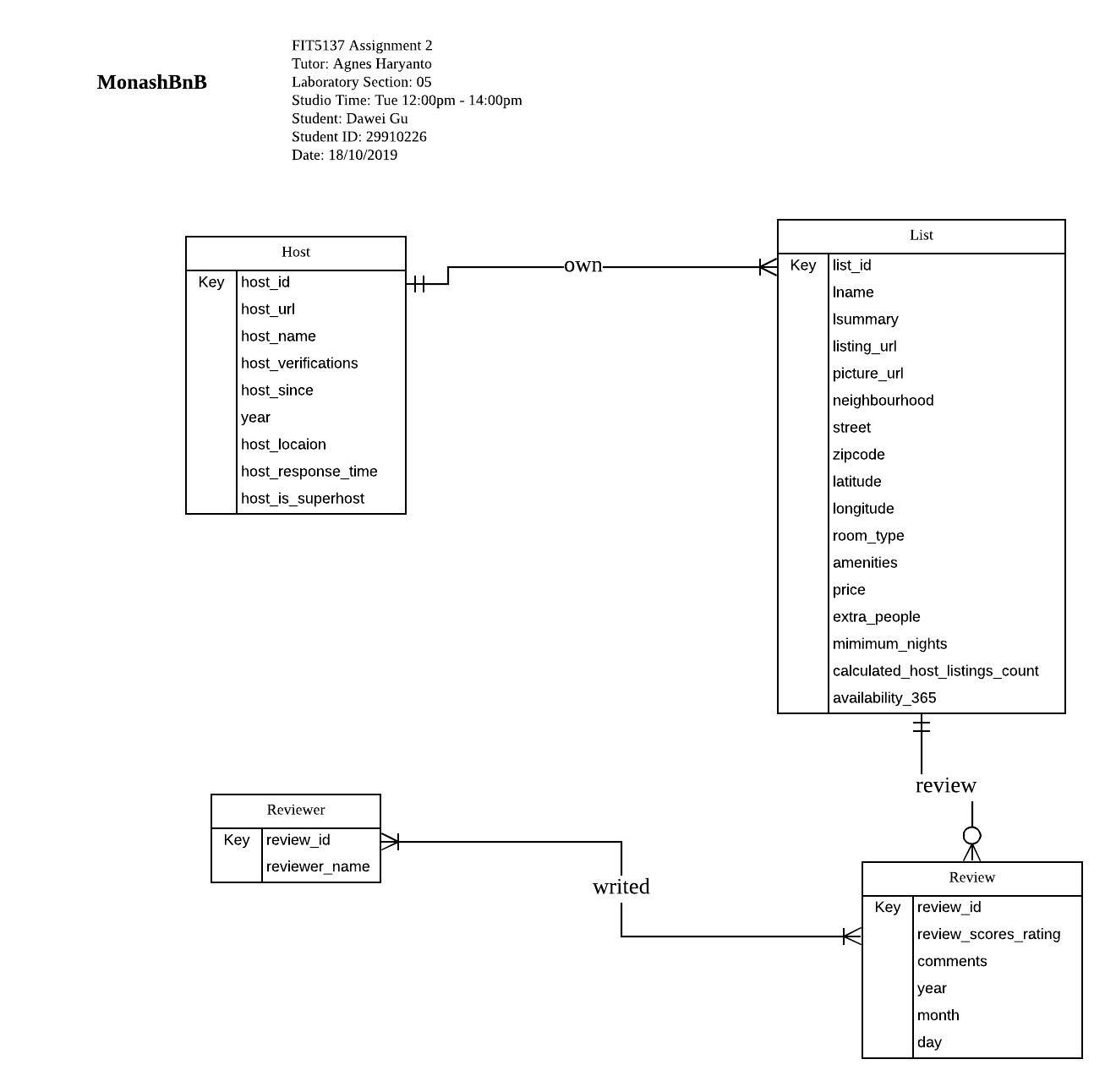
**Report**

**C.1 Database Design**

* **Create a new project called *FIT5137\_Assign2*.**
* **Create a new graph called *MonashBnBgraph*.**



There will be 4 labels’ nodes, they are Host, List, Reviewer, Review. There will be 3 labels’ relation amount the 4 label’s nodes, the host own list(house), the list(house) review by review, and the reviewer writed review. Following diagram show the design.



**Cypher Script**

**#loading host nodes**

LOAD CSV WITH HEADERS FROM "file:///host\_v2.csv"

AS row

WITH row WHERE row.host\_id IS NOT NULL

MERGE (h:Host {host\_id: row.host\_id})

ON CREATE SET h.host\_url = row.host\_url,

h.host\_name = row.host\_name,

h.host\_verifications = split(split(split(row.host\_verifications, '[')[1], ']')[0], ', '),

h.host\_since = row.host\_since,

h.year = toInteger(split(row.host\_since,'-')[0]),

h.host\_location = row.host\_location,

h.host\_response\_time = row.host\_response\_time,

h.host\_is\_superhost = row.host\_is\_superhost;

**#load list**

LOAD CSV WITH HEADERS FROM "file:///listing\_v2.csv" AS row

WITH row WHERE row.id IS NOT NULL

MERGE (l:List {list\_id: row.id})

ON CREATE SET l.name = row.name,

l.summary = row.summary,

l.listing\_url = row.listing\_url,

l.picture\_url = row.picture\_url,

l.neighbourhood = row.neighbourhood,

l.street= row.street,

l.zipcode = row.zipcode,

l.latitude = toFloat(row.latitude),

l.longitude = toFloat(row.longitude),

l.room\_type = row.room\_type,

l.amenities = split(split(split(row.amenities, '{')[1], '}')[0], ','),

l.price = toInteger(row.price),

l.extra\_people = toFloat(replace(row.extra\_people,'$', '')),

l.minimum\_nights = toInteger(row.minimum\_nights),

l.calculated\_host\_listings\_count = toInteger(row.calculated\_host\_listings\_count),

l.availability\_365 = toInteger(row.availability\_365);

**#loading reviewer nodes**

LOAD CSV WITH HEADERS FROM "file:///review\_v2.csv"

AS row

WITH row WHERE row.reviewer\_id IS NOT NULL

MERGE (r:Reviewer {reviewer\_id: row.reviewer\_id})

ON CREATE SET r.reviewer\_name = row.reviewer\_name;

**#loading review nodes**

LOAD CSV WITH HEADERS FROM "file:///review\_v2.csv"

AS row

WITH row WHERE row.id IS NOT NULL

MERGE (r:Review {review\_id: row.id})

ON CREATE SET r.year = toInteger(split(row.date,'-')[0]),

r.month = toInteger(split(row.date,'-')[1]),

r.day = toInteger(split(row.date,'-')[2]),

r.review\_scores\_rating = toInteger(row.review\_scores\_rating),

r.comments = row.comments;

**#add relationship between list and host**

LOAD CSV WITH HEADERS FROM "file:///listing\_v2.csv" AS csvLine

MATCH (h:Host {host\_id: csvLine.host\_id})

MATCH (l:List {list\_id: csvLine.id})

CREATE (h)-[:own]->(l);

**#add relationship between review and reviewer**

LOAD CSV WITH HEADERS FROM "file:///review\_v2.csv" AS csvLine

MATCH (p:Reviewer {reviewer\_id: csvLine.reviewer\_id})

MATCH (r:Review {review\_id: csvLine.id})

CREATE (p)-[:write]->(r);

**#add relationship between review and list**

LOAD CSV WITH HEADERS FROM "file:///review\_v2.csv" AS csvLine

MATCH (l:List {list\_id: csvLine.listing\_id})

MATCH (r:Review {review\_id: csvLine.id})

CREATE (r)-[:review]->(l);

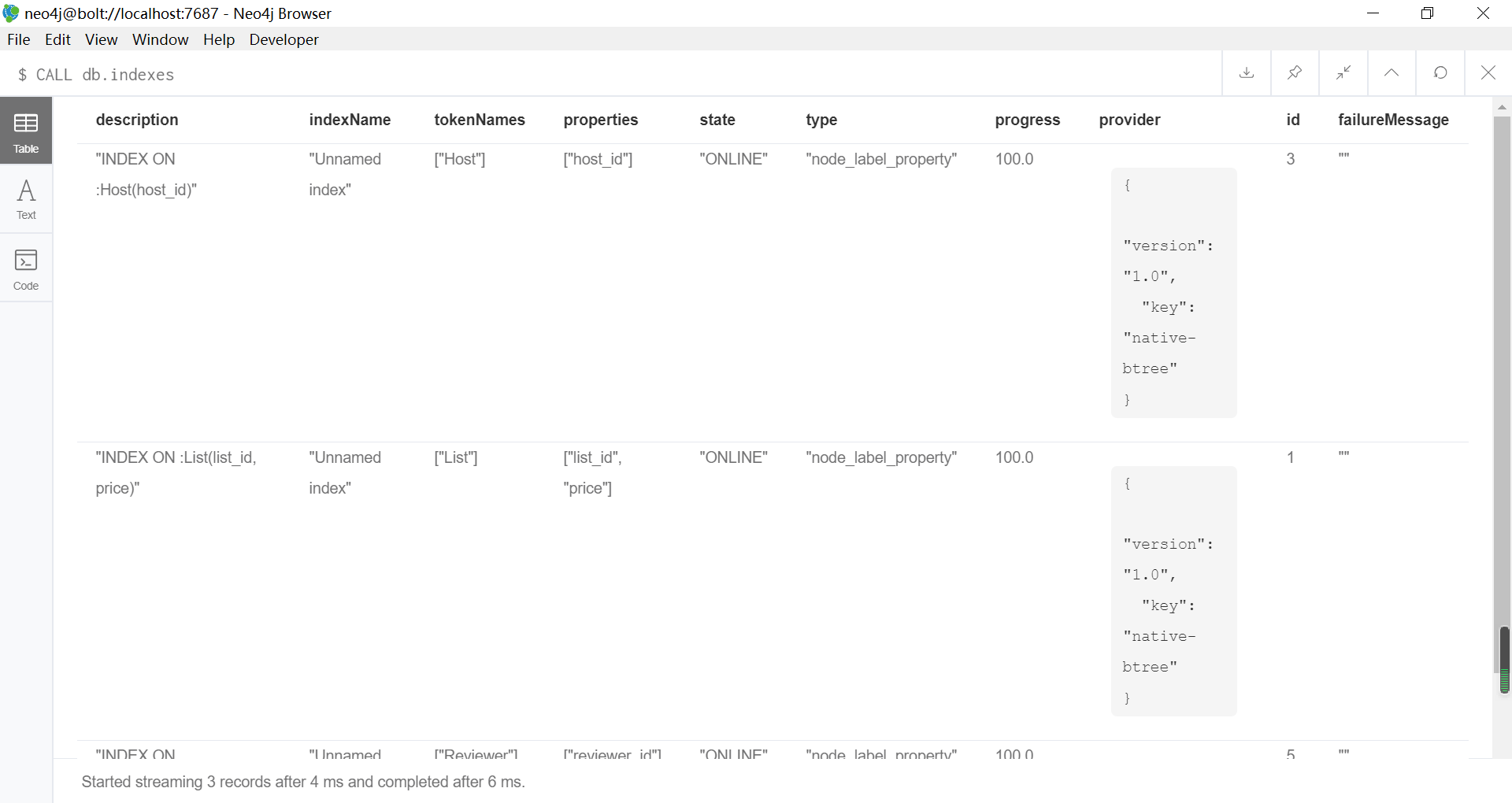
**C.2 Queries**

**#Create index**

CREATE INDEX ON :List(list\_id, price);

CREATE INDEX ON :Host(host\_id);

CREATE INDEX ON :Reviewer(reviewer\_id);



**1. How many reviews does “Sunny 1950s Apartment, St Kilda East” have?**

MATCH (l:List) -- (r:Review)

WHERE l.name CONTAINS 'Sunny 1950s Apartment, St Kilda East'

RETURN count(r);



**2. Show all reviews in Port Phillip.**

MATCH (l:List{neighbourhood:'Port Phillip'}) -- (r:Review)

RETURN l.neighbourhood, r;



**3. Can you recommend accommodations that Jerome (reviewer 4162110) has never been but Sandy & Pete (reviewer 317848) have stayed and gave ratings above 90?**

MATCH (l:List) -- (r:Review) -- (p:Reviewer)

WHERE p.reviewer\_id = '317848'

AND NOT p.reviewer\_id = '4162110'

AND r.review\_scores\_rating > 90

RETURN l.name, r.review\_scores\_rating, p.reviewer\_name;

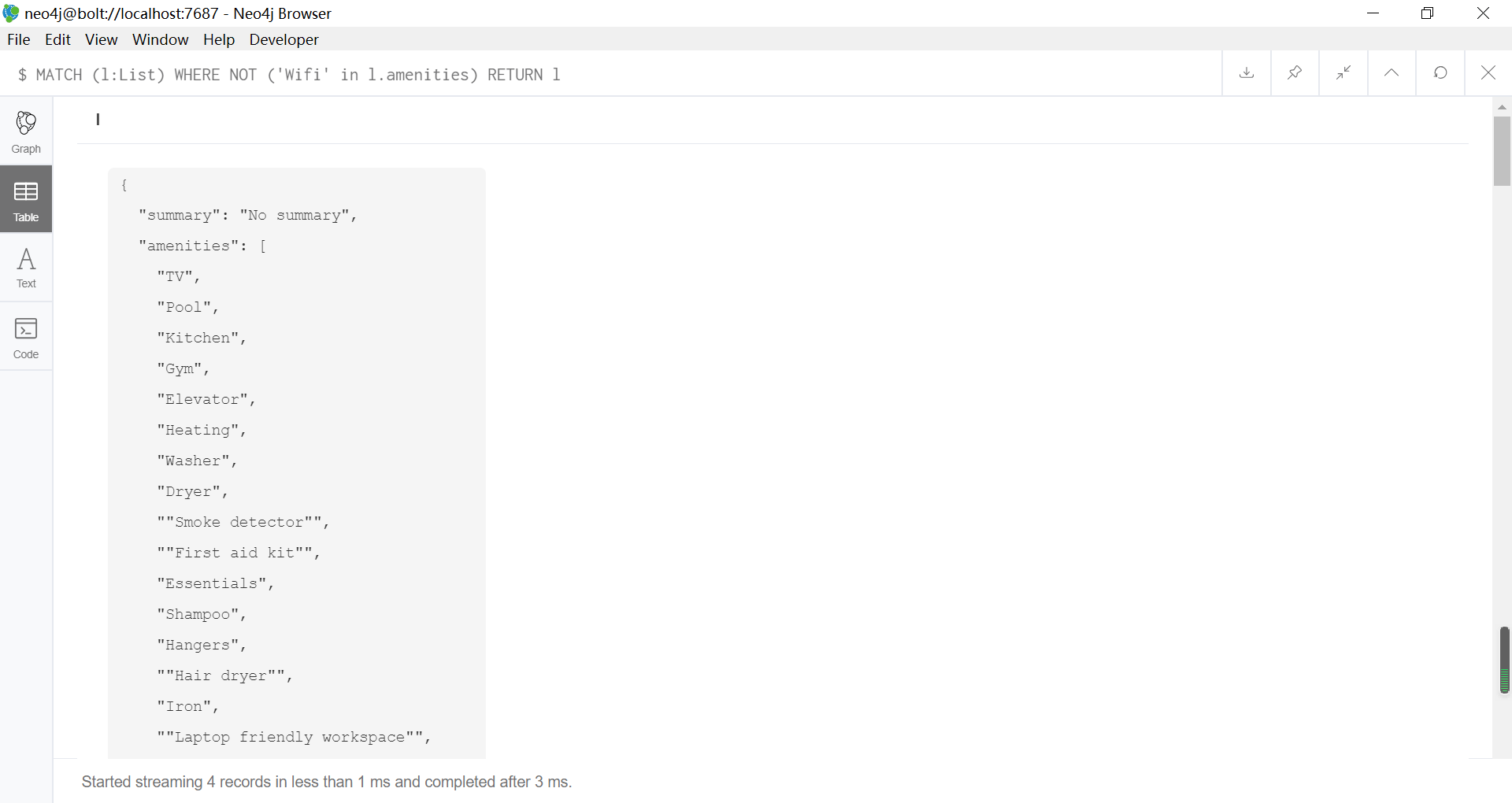


**4. List all accommodation names and locations that do not provide Wi-Fi.**

MATCH (l:List)

WHERE NOT ('Wifi' in l.amenities)

RETURN l



**5. Count how many times a reviewer left reviews.**

MATCH (p:Reviewer) -- (r:Review)

RETURN p.reviewer\_id AS id, p.reviewer\_name AS Name, count(r) AS Num\_review;



**6. Display a list of pairs of accommodations having more than three amenities in common.**

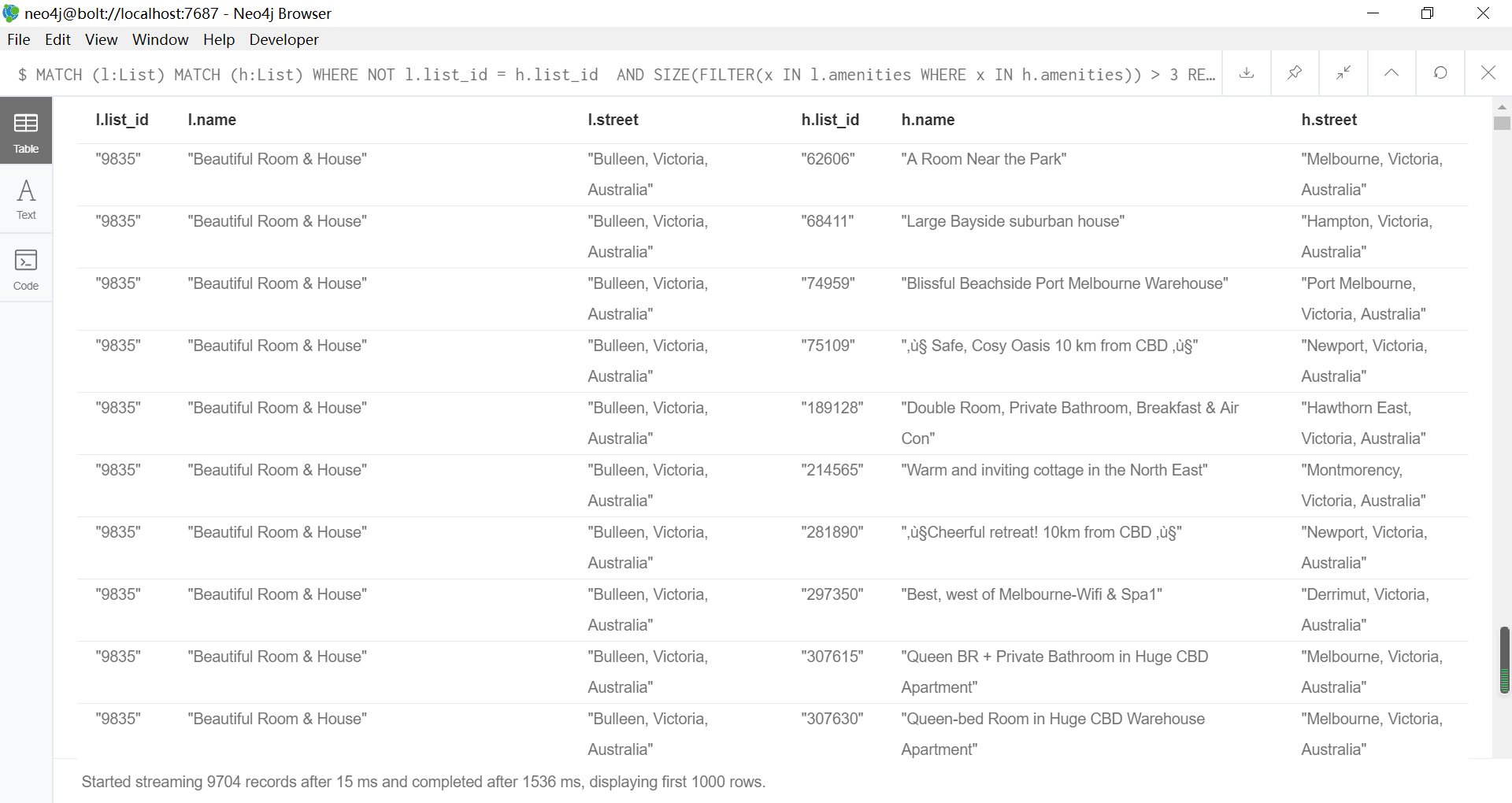
MATCH (l:List)

MATCH (h:List)

WHERE NOT l.list\_id = h.list\_id

AND SIZE(FILTER(x IN l.amenities WHERE x IN h.amenities)) > 3

RETURN l.list\_id, l.name, l.street, h.list\_id, h.name, h.street;

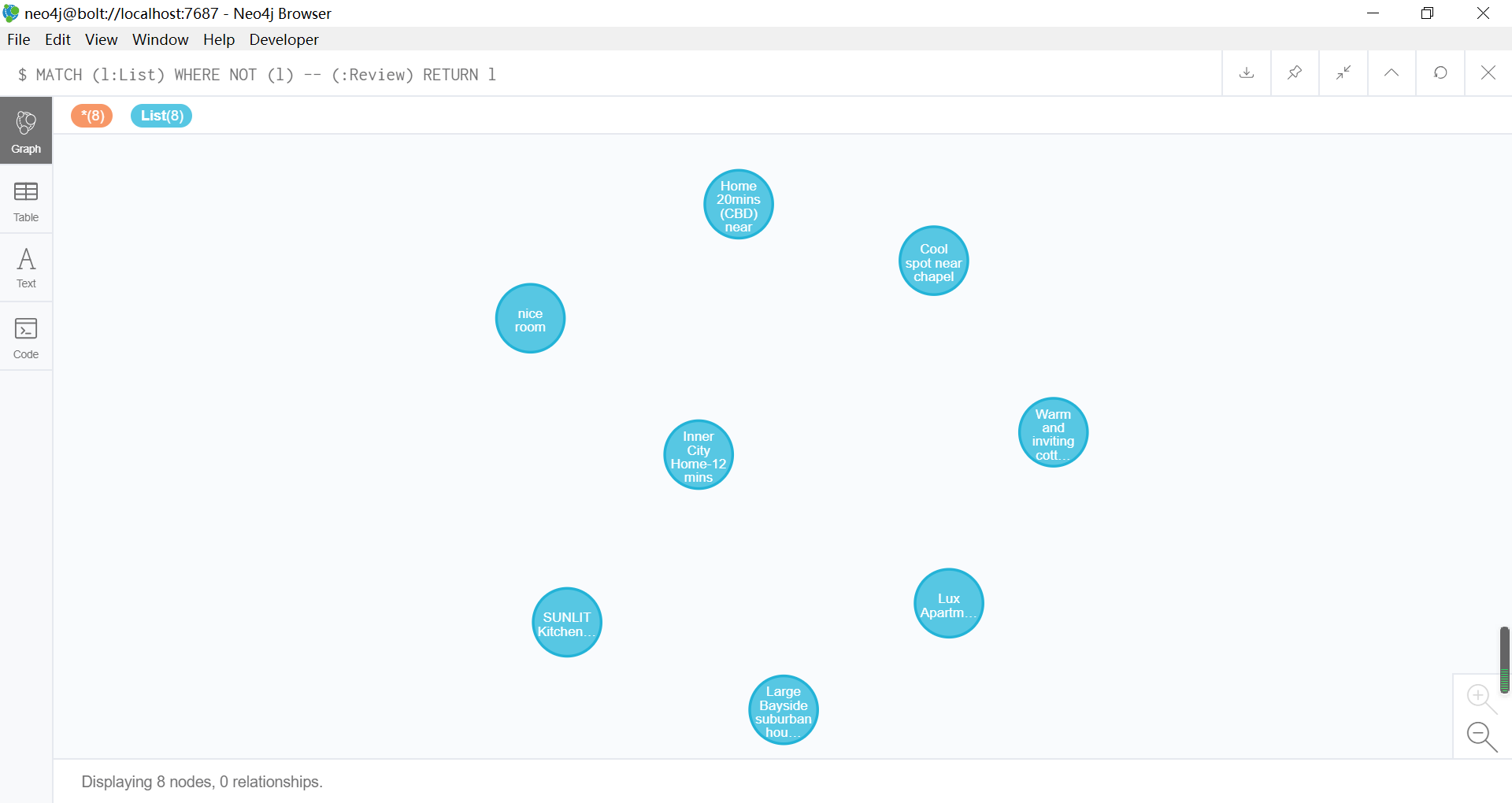


**7. Which listings do not have any review?**

MATCH (l:List)

WHERE NOT (l) -- (:Review)

RETURN l

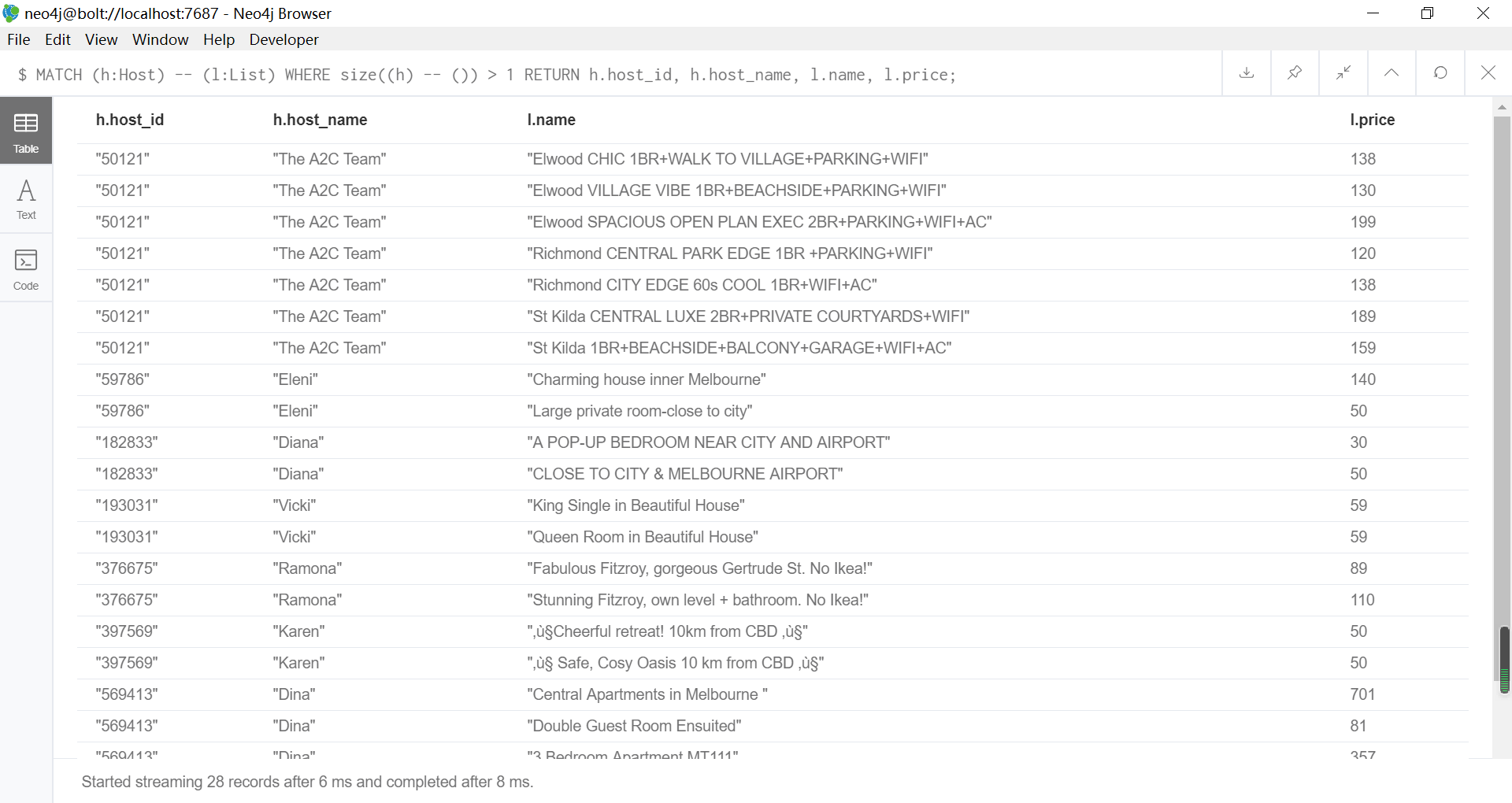


**8. Show all hosts who have multiple listings. Display both the host details and the listing name and price.**

MATCH (h:Host) -- (l:List)

WHERE size((h) -- ()) > 1

RETURN h.host\_id, h.host\_name, l.name, l.price;

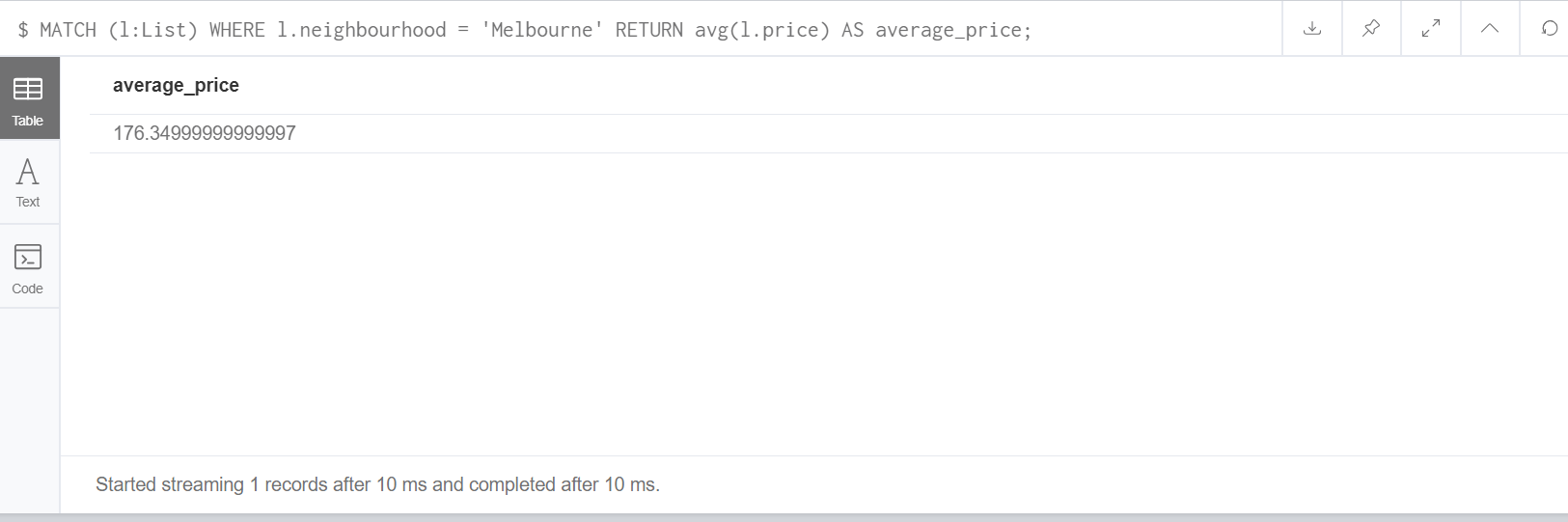


**9. What is the average price for accommodations in Melbourne neighbourhood?**

MATCH (l:List)

WHERE l.neighbourhood = 'Melbourne'

RETURN avg(l.price) AS average\_price;



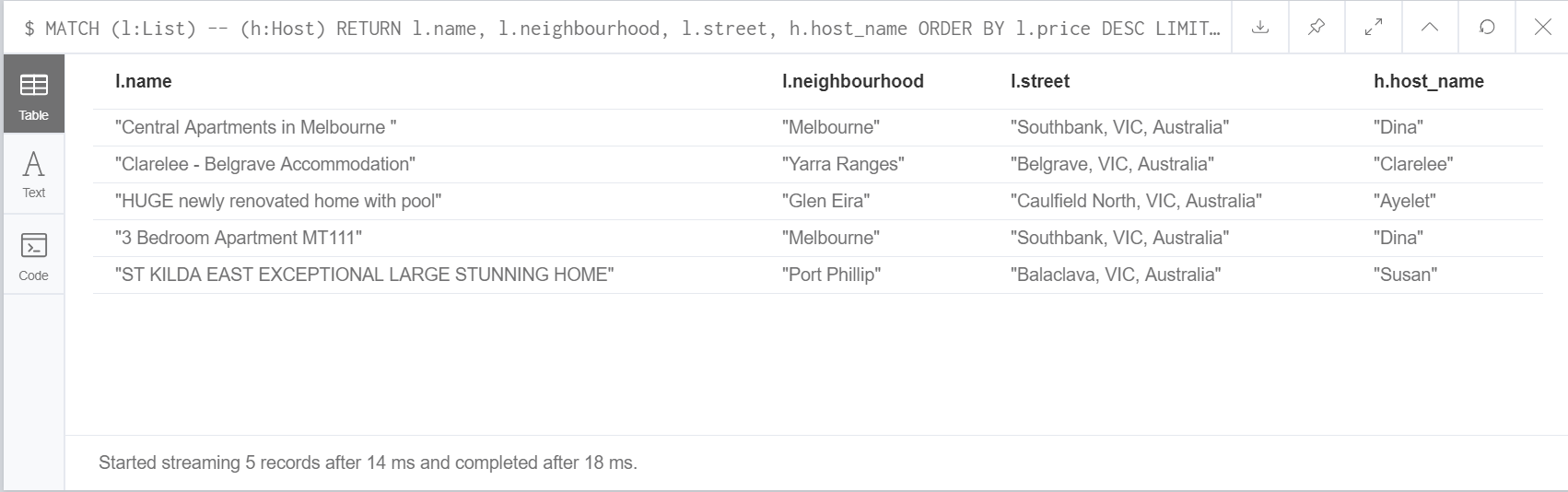
**10. Where are the top 5 most expensive accommodations? Display the locations, host information, and names of those accommodation.**

MATCH (l:List) -- (h:Host)

RETURN l.name, l.neighbourhood, l.street, h.host\_name

ORDER BY l.price DESC

LIMIT 5;



**11. How many accommodations were reviewed in 2017?**

MATCH (l:List) -- (r:Review)

WHERE r.year = 2017

RETURN COUNT(DISTINCT(l));



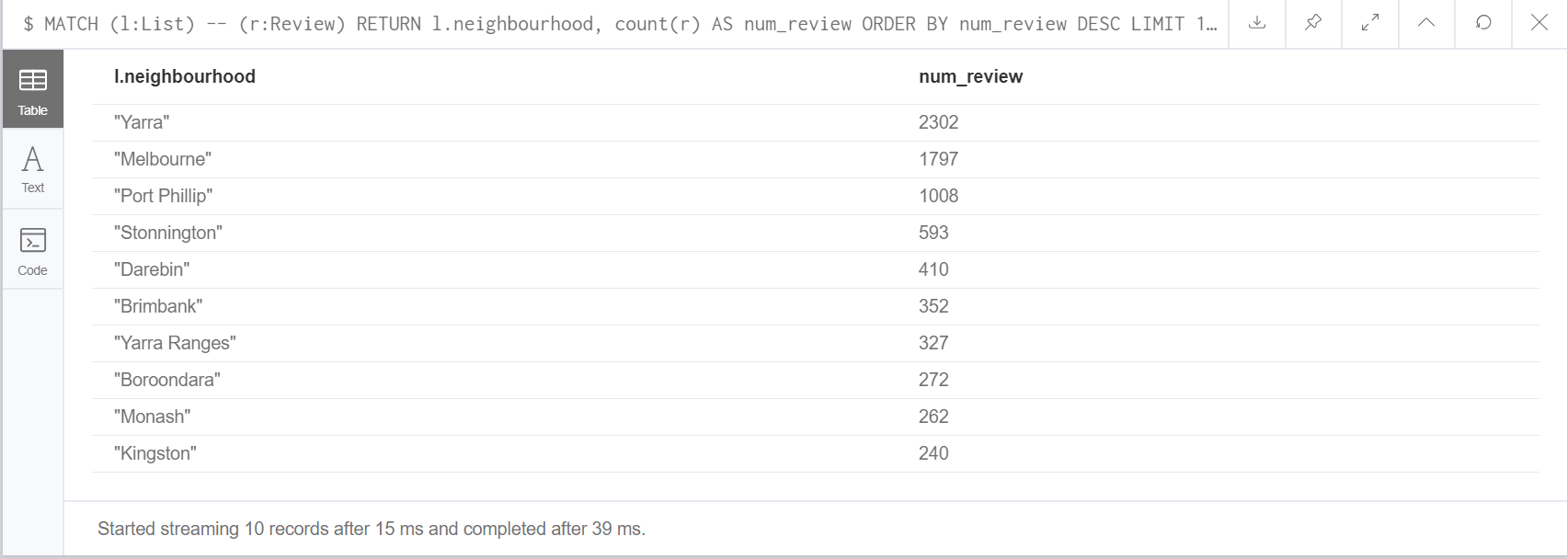
**12. What are the top 10 most popular neighbourhoods based on the total average reviews?**

MATCH (l:List) -- (r:Review)

RETURN l.neighbourhood, count(r) AS num\_review

ORDER BY num\_review DESC

LIMIT 10;

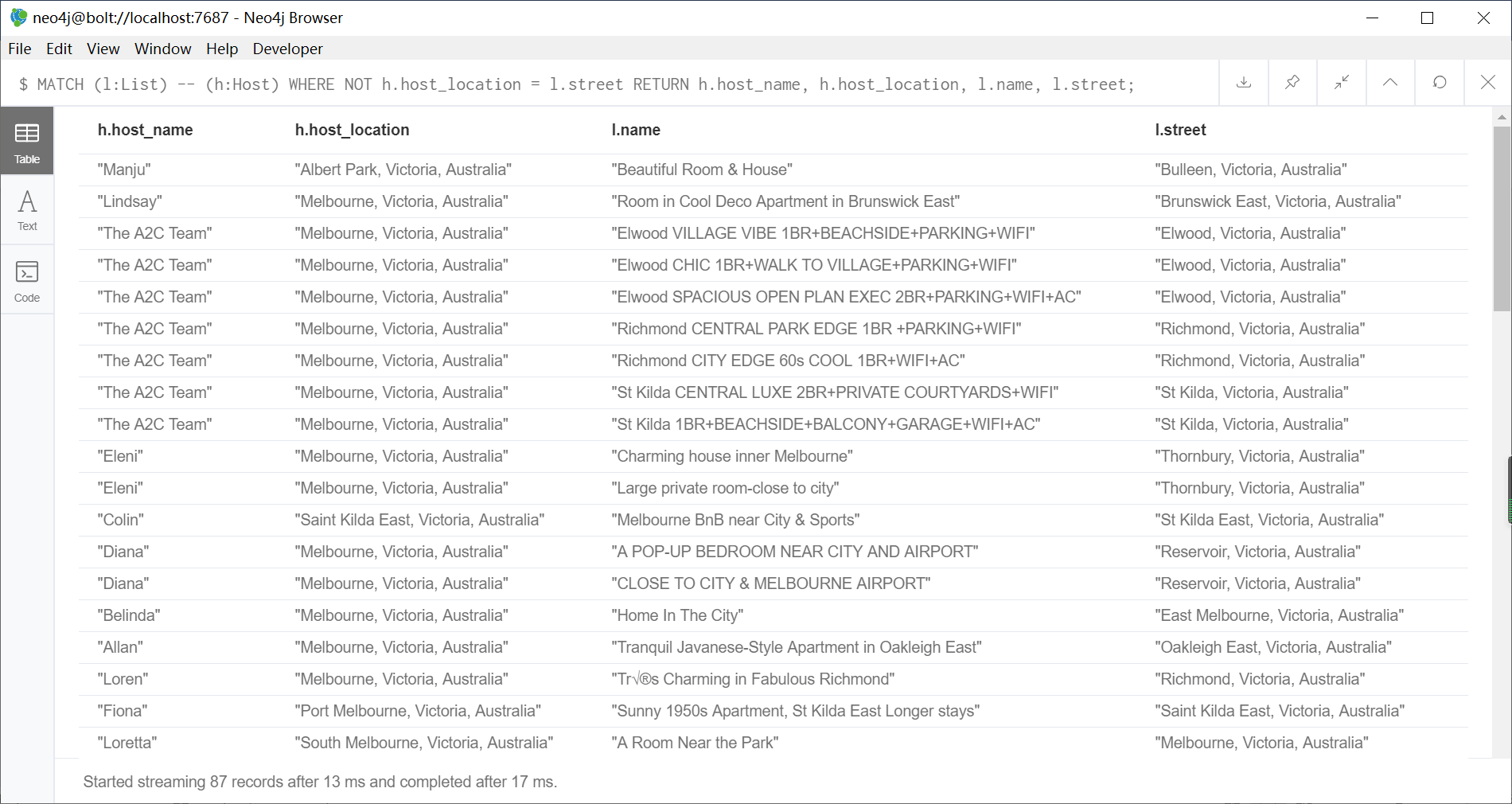


**13. Find hosts whose location are different from their listings. Show the host name, host location, listing name, and listing location.**

MATCH (l:List) -- (h:Host)

WHERE NOT h.host\_location = l.street

RETURN h.host\_name, h.host\_location, l.name, l.street;



**14. Assuming that each accommodation only accepts two guests, calculate the price of each accommodation for four people staying for five nights. Display only the accommodation name, location, price per night, extra people charge, and total price. Rank the accommodation from the cheapest price.**

MATCH (l:List)

WHERE l.minimum\_nights < 6

AND l.availability\_365 > 4

RETURN l.name, l.street, l.price, l.extra\_people, (5\*l.price + 5\*2\*l.extra\_people) AS total\_price

ORDER BY total\_price; 

**15. For each listing, rank other listings that are close to each other by their locations. You will need to use the longitude and latitude to calculate the distance between listings.**

MATCH (l:List)

MATCH (h:List)

WHERE NOT l.list\_id = h.list\_id

WITH point({ longitude: l.longitude, latitude: l.latitude }) AS pointa,

l.list\_id AS lista\_id,

l.name AS lista\_name,

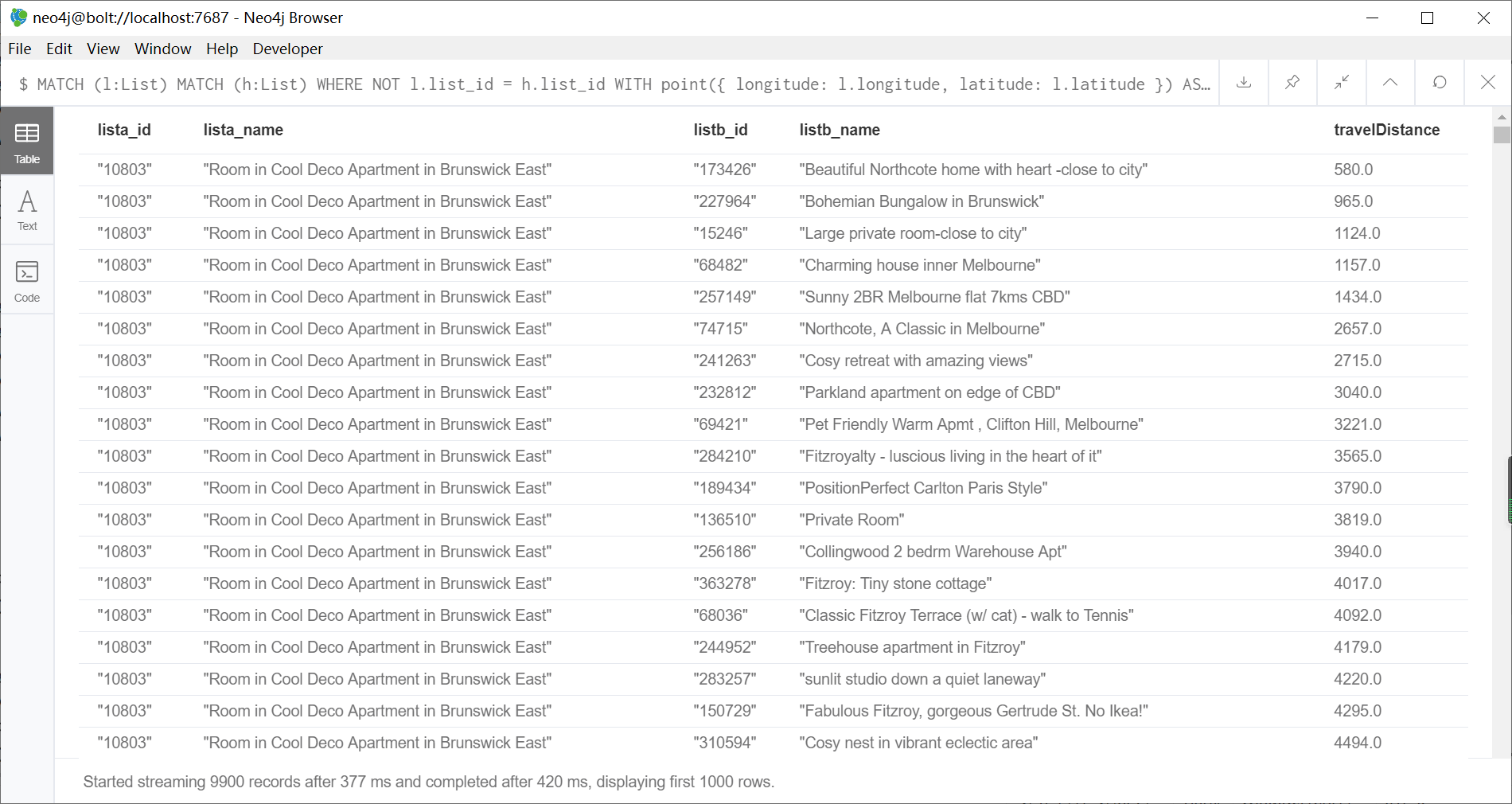
point({ longitude: h.longitude, latitude: h.latitude }) AS pointb,

h.list\_id AS listb\_id,

h.name AS listb\_name

RETURN lista\_id, lista\_name, listb\_id, listb\_name, round(distance(pointa, pointb)) AS travelDistance

ORDER BY lista\_id, travelDistance;



**Additional 5 Queries:**

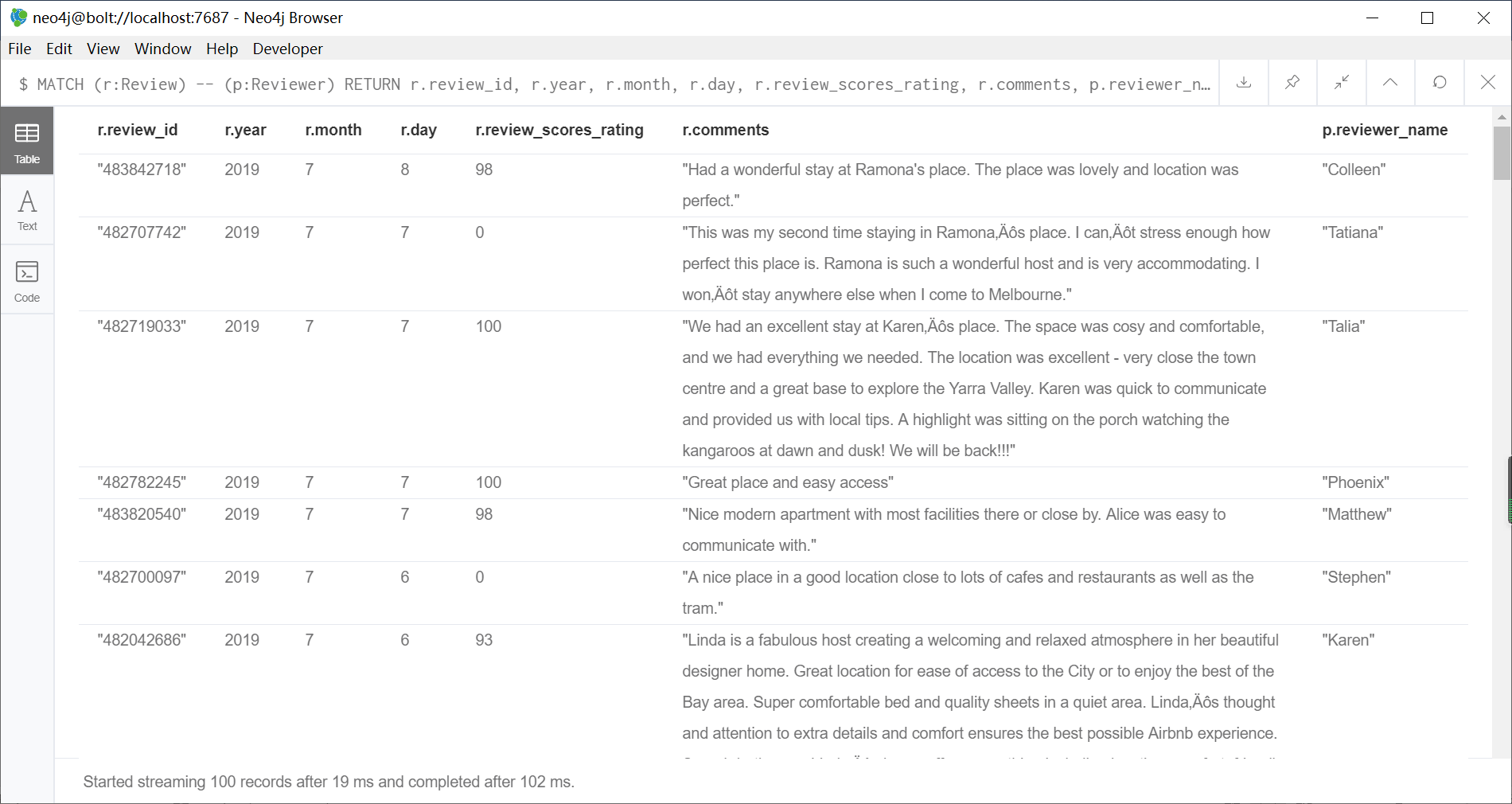
**Addional Q1. What is most recent 100 reviews, and how is reviewer?**

MATCH (r:Review) – (p:Reviewer)

RETURN r.review\_id, r.year, r.month, r.day, r.review\_scores\_rating, r.comments, p.reviewer\_name

ORDER BY r.year DESC, r.month DESC, r.day DESC

LIMIT 100;

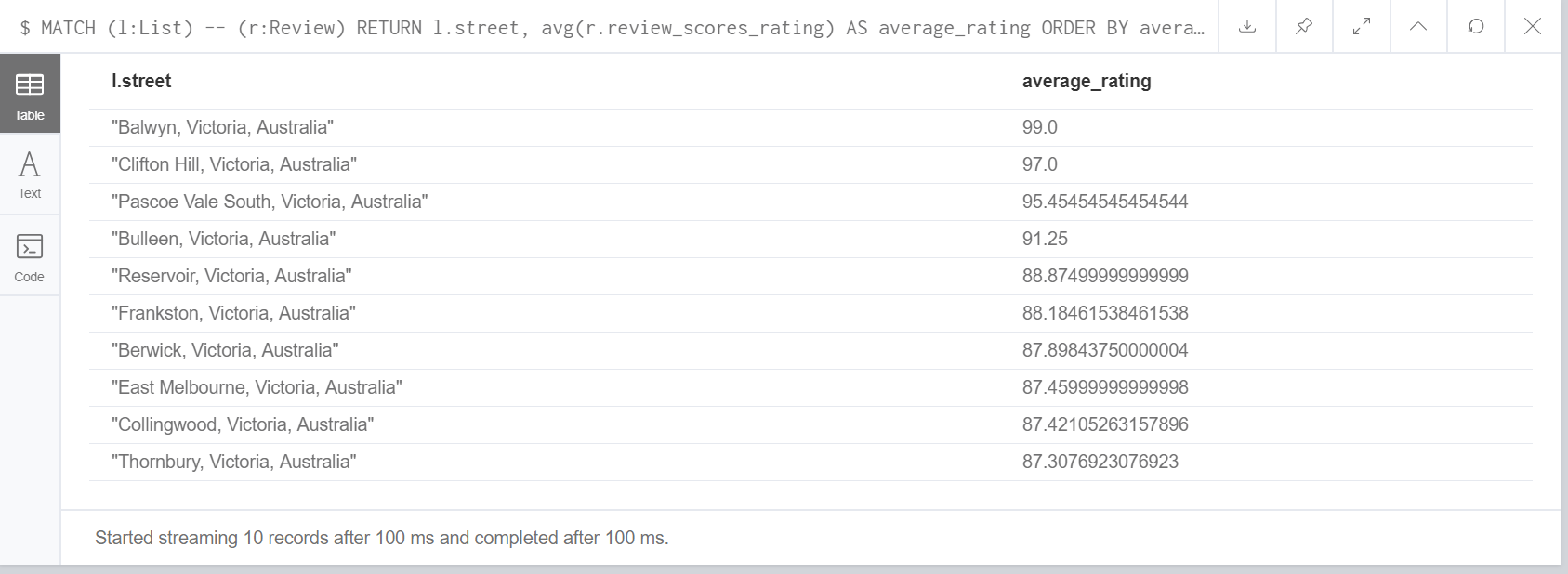


**Addional Q2. What is the top 10 cities have the highest review rating?**

MATCH (l:List) -- (r:Review)

RETURN l.street, avg(r.review\_scores\_rating) AS average\_rating

ORDER BY average\_rating DESC

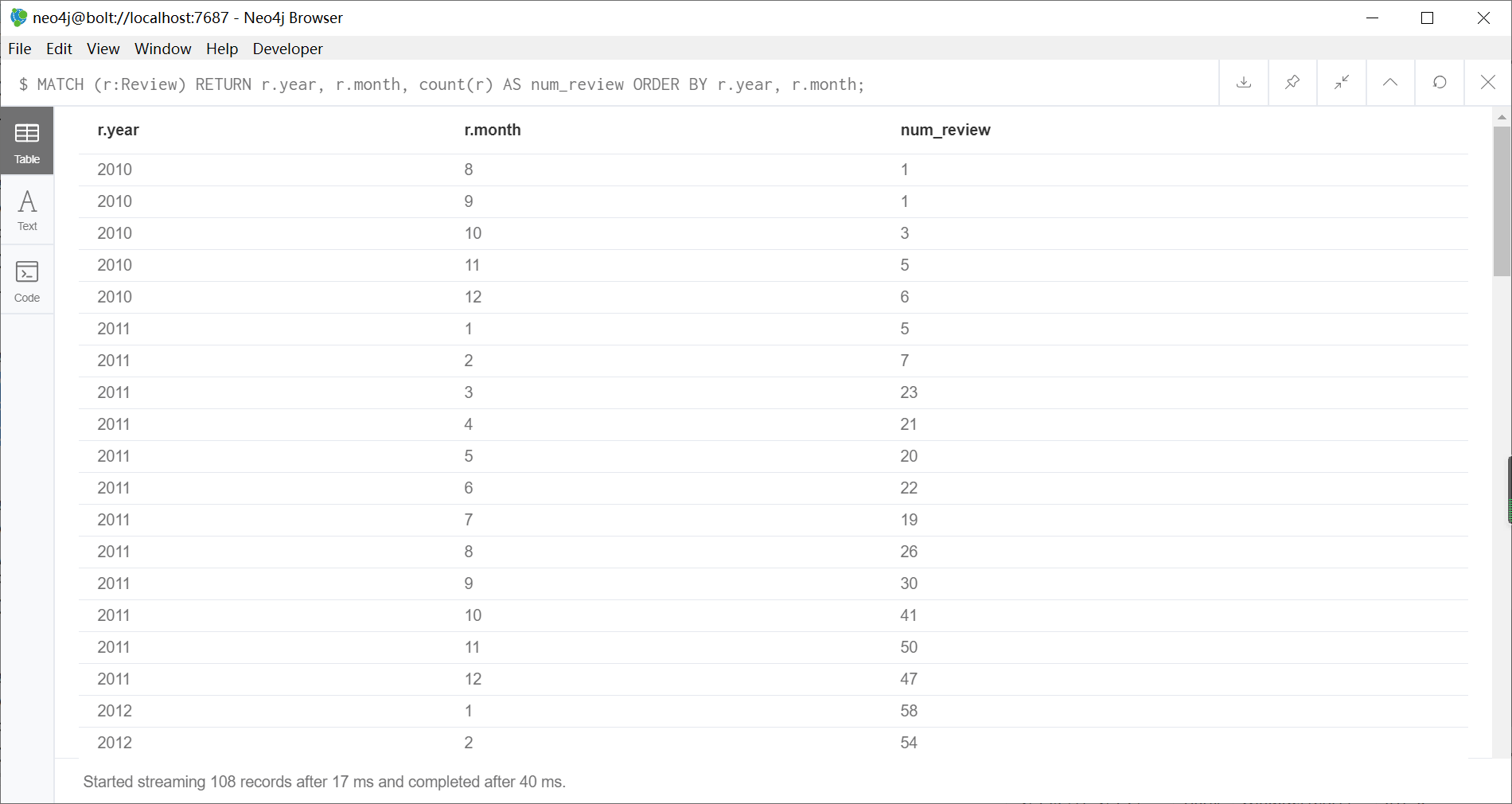
LIMIT 10; 

**Addional Q3. What is number of reviews per month?**

MATCH (r:Review)

RETURN r.year, r.month, count(r) AS num\_review

ORDER BY r.year, r.month;



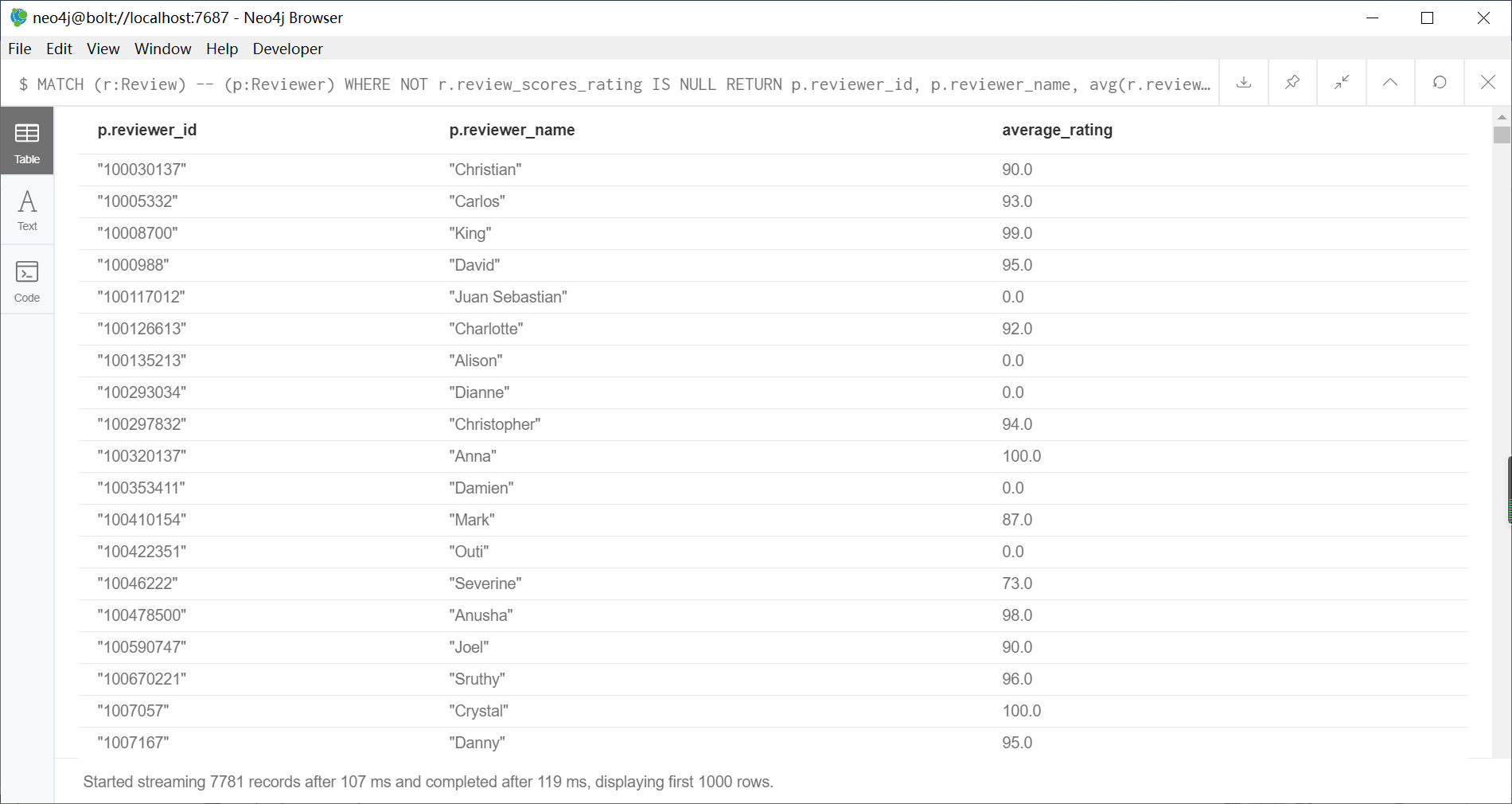
**Addional Q4. What the average rate that each reviewer give?**

MATCH (r:Review) -- (p:Reviewer)

WHERE NOT r.review\_scores\_rating IS NULL

RETURN p.reviewer\_id, p.reviewer\_name, avg(r.review\_scores\_rating) AS average\_rating

ORDER BY p.reviewer\_id;

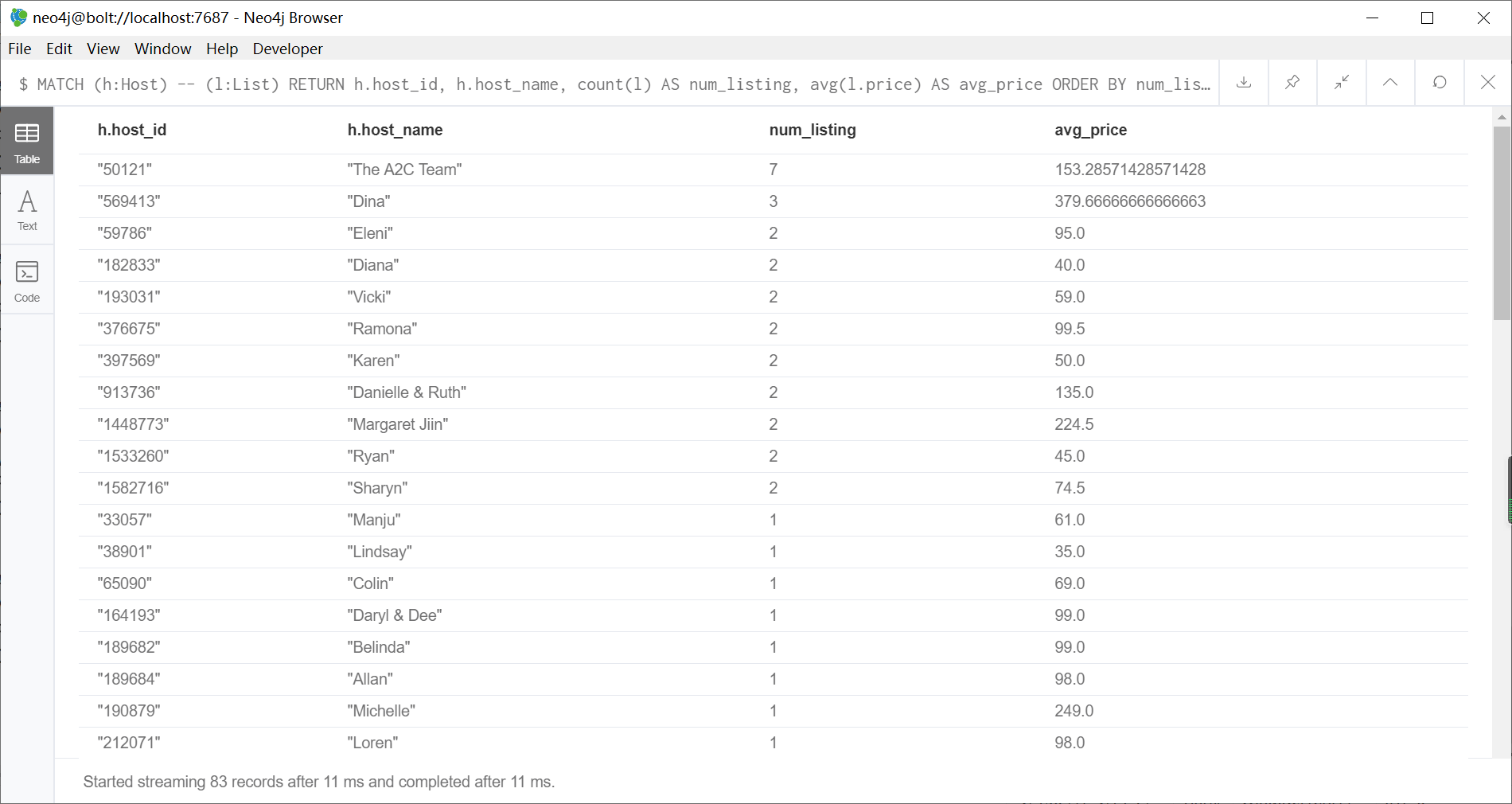


**Addional Q5. What is number of accommodation and avgerage price of accommodation of each host?**

MATCH (h:Host) -- (l:List)

RETURN h.host\_id, h.host\_name, count(l) AS num\_listing, avg(l.price) AS avg\_price

ORDER BY num\_listing DESC;



**C.3 Database Modifications**

**1. Go to AirBnB website and add three new listings, including the hosts details and some related reviews of the listings you chose. The IDs in this case can be assigned manually by yourself.**

Links of 3 listing:

1.<https://www.airbnb.com.au/rooms/17465305?location=Melbourne%2C%20Victoria&source_impression_id=p3_1571471684_x5DzWDLotJ0wpaON>

2.<https://www.airbnb.com.au/rooms/28907077?location=Melbourne%2C%20Victoria&source_impression_id=p3_1571472315_yRtC4GK5bctqyn7A>

3.<https://www.airbnb.com.au/rooms/28630130?source_impression_id=p3_1571566656_GNiZmhPKNPlp5wNS>

Links of 2 host (listing 1 belong to host 1, and listing 2 belong to host 2, list 3 belong to host 3)

1.<https://www.airbnb.com.au/users/show/98798133>

2.<https://www.airbnb.com.au/users/show/41369546>

3. <https://www.airbnb.com.au/users/show/153057309>

CREATE (h:Host {host\_id: '98798133',

host\_url: 'https://www.airbnb.com.au/users/show/98798133',

host\_name: 'Hadi',

host\_verifications: ['Government ID', 'Email address', 'Phone number'],

host\_since: '2016/10/08',

year: 2016,

host\_location: 'Melbourne, Australia',

host\_response\_time: 'N/A',

host\_is\_superhost: 'f'});

CREATE (h:Host {host\_id: '41369546',

host\_url: 'https://www.airbnb.com.au/users/show/41369546',

host\_name: 'Wendy',

host\_verifications: ['Government ID', 'Selfie', 'Email address', 'Phone number'],

host\_since: '2015/08/13',

year: 2015,

host\_location: 'Melbourne, Australia',

host\_response\_time: 'N/A',

host\_is\_superhost: 'f'});

CREATE (h:Host {host\_id: '153057309',

host\_url: 'https://www.airbnb.com.au/users/show/153057309',

host\_name: 'Bryan And Soraida',

host\_verifications: ["Government ID","Selfie","Email address","Phone number"],

host\_since: '2017/10/03',

year: 2017,

host\_location: 'Melbourne, Australia',

host\_response\_time: 'N/A',

host\_is\_superhost: 't'});

CREATE (l:List {list\_id: '1',

name: 'Cityview Master Bedroom in The Green Abode',

summary: 'Located in the heart of Melbourne with a vibrant and thriving lifestyle. Guest(s) will stay in the master bedroom furnished with king size mattress complete with 2 pillows and bed sheets. The common spaces such as living room and kitchen area are fully furnished dominated by minimalist style furnitures and greeneries. Perfect for solo nomad or couple. This unit opens its door to any types of couple or individual.',

listing\_url: 'https://www.airbnb.com.au/rooms/17465305',

picture\_url: 'N/A',

neighbourhood: 'Melbourne',

street: 'Melbourne, VIC, Australia',

zipcode: '3000',

latitude: -37.816114,

longitude: 144.953123,

room\_type: 'Private room',

amenities: ['Wifi', 'Dryer', 'Air conditioning', 'Washing mashine', 'Essentials', 'TV', 'Heating', 'Hot water', 'Lift', 'Gym', 'Pool', 'Paid parking off premisses', 'Free street parking', 'Microwave', 'Refrigerator', 'Oven', 'Stove'],

price: 50,

extra\_people: 0,

minimum\_nights: 1,

calculated\_host\_listings\_count: 1,

availability\_365: 120});

CREATE (l:List {list\_id: '2',

name: 'Eco-friendly Studio\*Private bathroom\*Wifi\*Pool\*Gym',

summary: 'A cozy and homey studio located in the CBD, perfect for tourists and professionals. Easily accessible by public transports (free tram zone, 100m Melbourne Central Station). Close to some top rated attractions (State Library, Royal Exhibition Building, Parliament House), two steps away from QV Center, China Town, restaurants, cafes, pubs. Swimming pool, sauna, gym are available in the building.' ,

listing\_url: 'https://www.airbnb.com.au/rooms/28907077',

picture\_url: 'N/A',

neighbourhood: 'Melbourne',

street: 'Melbourne, VIC, Australia',

zipcode: '3000',

latitude: -37.810159,

longitude: 144.967390,

room\_type: 'Entire home/apt',

amenities: ['Wifi', 'Essentials', 'TV', 'Heating', 'Hot water', 'Lift', 'Gym', 'Pool', 'Paid parking off premisses', 'Microwave', 'Refrigerator', 'Stove'],

price: 67,

extra\_people: 20,

minimum\_nights: 1,

calculated\_host\_listings\_count: 2,

availability\_365: 90});

CREATE (l:List {list\_id: '3',

name: 'Studio~Best Memories in CBD ❤ \*Free Tram Zone!',

summary: 'N/A',

listing\_url: 'https://www.airbnb.com.au/rooms/28630130',

picture\_url: 'N/A',

neighbourhood: 'Melbourne',

street: 'Melbourne, VIC, Australia',

zipcode: '3000',

latitude: -37.809941,

longitude: 144.958056,

room\_type: 'Entire home/apt',

amenities: ['Wifi', 'Essentials', 'TV', 'Heating', 'Hot water', 'Lift', 'Gym', 'Pool', 'Paid parking off premisses', 'Microwave', 'Refrigerator', 'Stove'],

price: 50,

extra\_people: 0,

minimum\_nights: 2,

calculated\_host\_listings\_count: 2,

availability\_365: 120});

CREATE (r:Review {review\_id: '1',

year: 2019,

month: 8,

day: 1,

review\_scores\_rating: 95,

comments: 'Very clean place and supe close to everything. So much great food and bars around'});

CREATE (r:Reviewer {reviewer\_id: '1', reviewer\_name: 'Joel'});

CREATE (r:Review {review\_id: '2',

year: 2019,

month: 10,

day: 1,

review\_scores\_rating: 96,

comments: "It's a good place for a short trip to Melbourne. Very convenient to get around everywhere."});

CREATE (r:Reviewer {reviewer\_id: '2', reviewer\_name: 'Chia Jane'});

CREATE (r:Review {review\_id: '3',

year: 2019,

month: 10,

day: 04,

review\_scores\_rating: 98,

comments: 'Absolutely amazing spot for our trip :D'});

CREATE (r:Reviewer {reviewer\_id: '3', reviewer\_name: 'Alex'});

MATCH (l:List{list\_id: '1'})

MATCH (h:Host{host\_id: '98798133'})

CREATE (h)-[:own]->(l);

MATCH (l:List{list\_id: '2'})

MATCH (h:Host{host\_id: '41369546'})

CREATE (h)-[:own]->(l);

MATCH (l:List{list\_id: '3'})

MATCH (h:Host{host\_id: '153057309'})

CREATE (h)-[:own]->(l);

MATCH (l:List{list\_id: '1'})

MATCH (r:Review{review\_id: '1'})

CREATE (r)-[:review]->(l);

MATCH (l:List{list\_id: '2'})

MATCH (r:Review{review\_id: '2'})

CREATE (r)-[:review]->(l);

MATCH (l:List{list\_id: '3'})

MATCH (r:Review{review\_id: '3'})

CREATE (r)-[:review]->(l);

MATCH (p:Reviewer {reviewer\_id: '1'})

MATCH (r:Review {review\_id: '1'})

CREATE (p)-[:write]->(r);

MATCH (p:Reviewer {reviewer\_id: '2'})

MATCH (r:Review {review\_id: '2'})

CREATE (p)-[:write]->(r);

MATCH (p:Reviewer {reviewer\_id: '3'})

MATCH (r:Review {review\_id: '3'})

CREATE (p)-[:write]->(r);

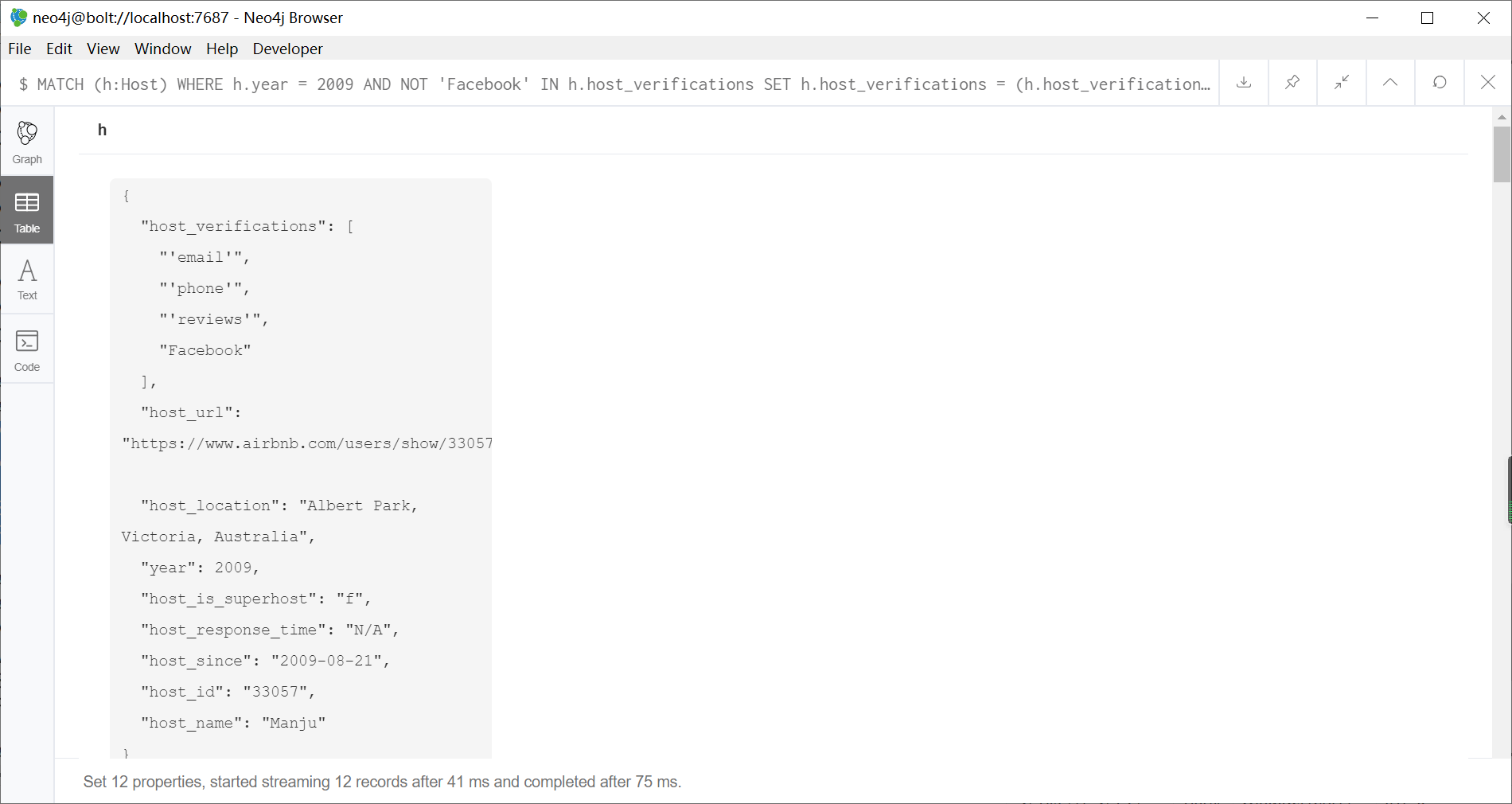
**2. Update the host verification for those who registered in 2009 and add Facebook to the list of existing verifications.**

MATCH (h:Host)

WHERE h.year = 2009

AND NOT 'Facebook' IN h.host\_verifications

SET h.host\_verifications = (h.host\_verifications + 'Facebook')

RETURN h;

**3. Update hosts who respond “within an hour” to a superhost. For this update you may only use the “host response time” and “host is a super host” information.**

MATCH (h:Host)

WHERE h.host\_response\_time = 'within an hour'

SET h.host\_is\_superhost = 't'

RETURN h;

**4. Update hosts who do not receive any reviews for their accommodation since 2017 and add a new property called active. This new property accepts Boolean value.**

MATCH (h:Host) -- (l:List)

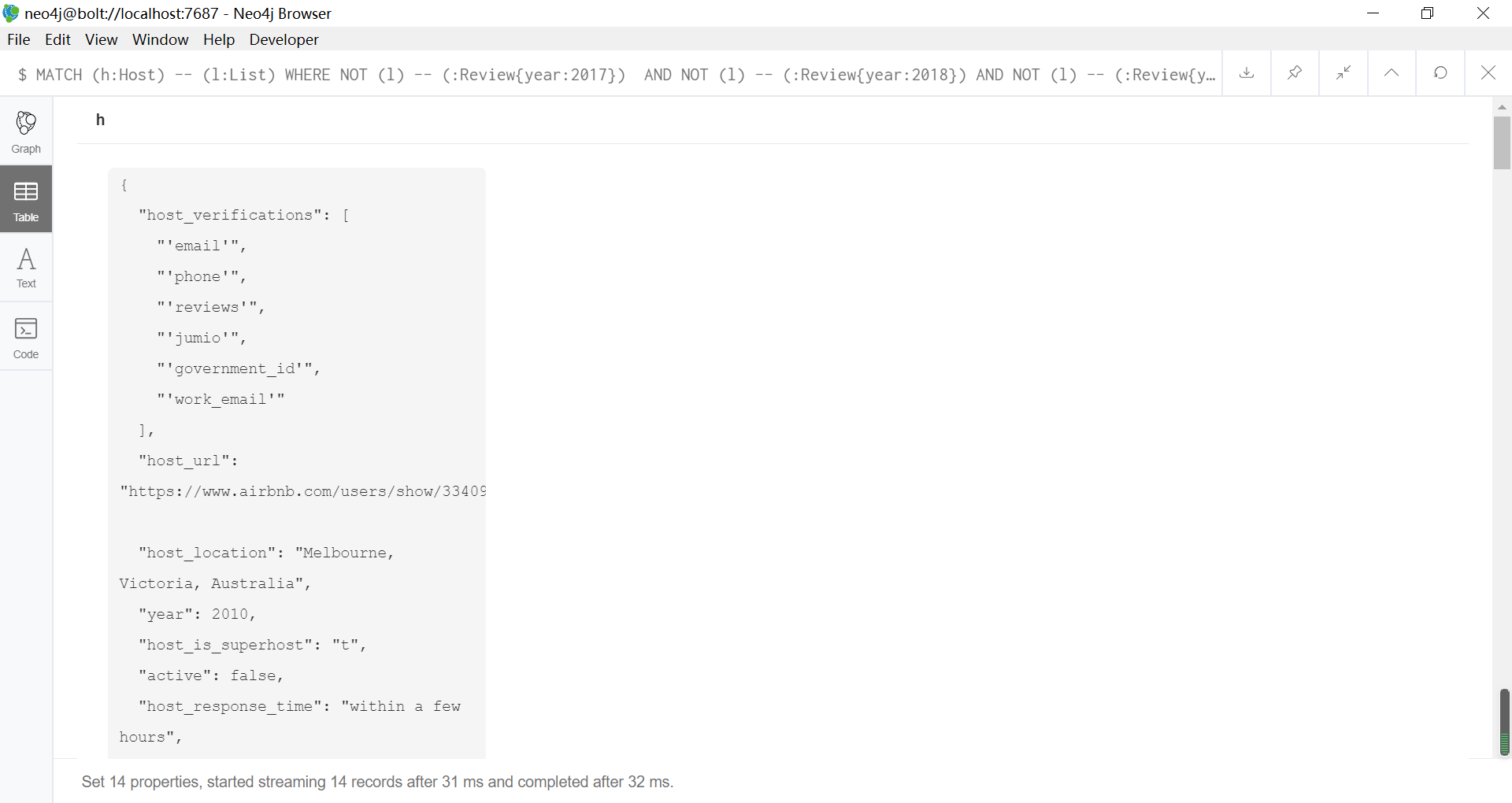
WHERE NOT (l) -- (:Review{year:2017})

AND NOT (l) -- (:Review{year:2018})

AND NOT (l) -- (:Review{year:2019})

SET h.active = false

RETURN h;



**5. Delete all listings with zero availability and have no reviews.**

MATCH (l:List)

WHERE l.availability\_365 = 0

OR NOT (l) -- (:Review)

DETACH DELETE l

RETURN l;



**C.4 Advanced Topic**

As we design the MoanshBnB, the most similar, successful and famous travel accommodation booking systems is AirBnB. AirBnB provide the service all over world. They are facing large volume, high velocity, high variety data every day. Since the data in AirBnB is not just data itself, AirBnB also more focus on the relationship of data. In that case, the graph database is very suitable for store the data and its relationship. Since the Neo4j is the one of best graph database, so AirBnB use Neo4j for store the data.

In the blog of Bodley, J & Williams, C, AirBnB start up to store data into Hive data warehouse, trying to create the database similar as graph database in the Hive data warehouse. Then, the data from Hive data warehouse have been through a tool called Airflow to transform the data, Python did the further processing to help data fit into neo4j. The neo4j driver input the data into neo4j, the user can use a python web framework called ‘Flask’ to connect to a search engine called ‘Elasticsearch’ to search, or in other word querying. In this framework, the Hive data warehouse is used to store the raw data. Airflow and Python are used as data stream process and data wrangling. Neo4j is the main database used to store the data. The Flask and Elasticsearch are used for query or search.

For our system MonashBnB, the volume of data may not be very big, and the velocity of incoming data also may not be very high, but we need to prepare for big data situation. We can build similar framework as AirBnB. Since our data volume and velocity is not big or high as AirBnB, we may not need data warehouse to store the data first, we can use framework like apache spark to handle the income data. Since Python is very suitable for neo4j, we can use Python for data wrangling and transfer the data, and finally storing data into neo4j. That is my first suggestion.

For second suggestion, the data form in current data is inconsistent. For example, the location of some host contains city, state and country like host 38901 in ‘Melbourne, Victoria, Australia’, but some host like host 33057 only write ‘AU’, it will cause problem when we do some query. We may create the property like city, state and country to make it clear.

The third suggestion is about the standard of data in different table. In the listing data, this problem also exists. The location recorded in street. Although it stores location as in host, but the standard is different. The state is stored as abbreviation in listing, but full name in host., it causes the inconsistent to cause problem in future query. The name of city also is inconsistent, for example, there is another name ‘Saint Kilda East’ for ‘St. Kilda East’. As solution, we need to perform the data wrangling of data before storing it in database to ensure the data have same standard.

For fourth suggestion, we may need to combine the reviewer and host table together. Currently, we have two separate tables (labels) for host and reviewer, but the host also can be reviewer. For example, the host named ‘Timmy’, and system have recorded the name ‘Timmy’ and unique id in host table (label), and recorded the name ‘Tim’ and another unique id in reviewer table (label). As result, it may cause inconsistency and duplication in the database, the solution is that create table (label) called user to combine the data from host and reviewer. That also avoid the host use another name to review their own listing.

Reference

Bodley, J & Williams, C. (2018). Democratizing Data Discovery at Airbnb. Retrieved from https://neo4j.com/blog/democratizing-data-discovery-airbnb/