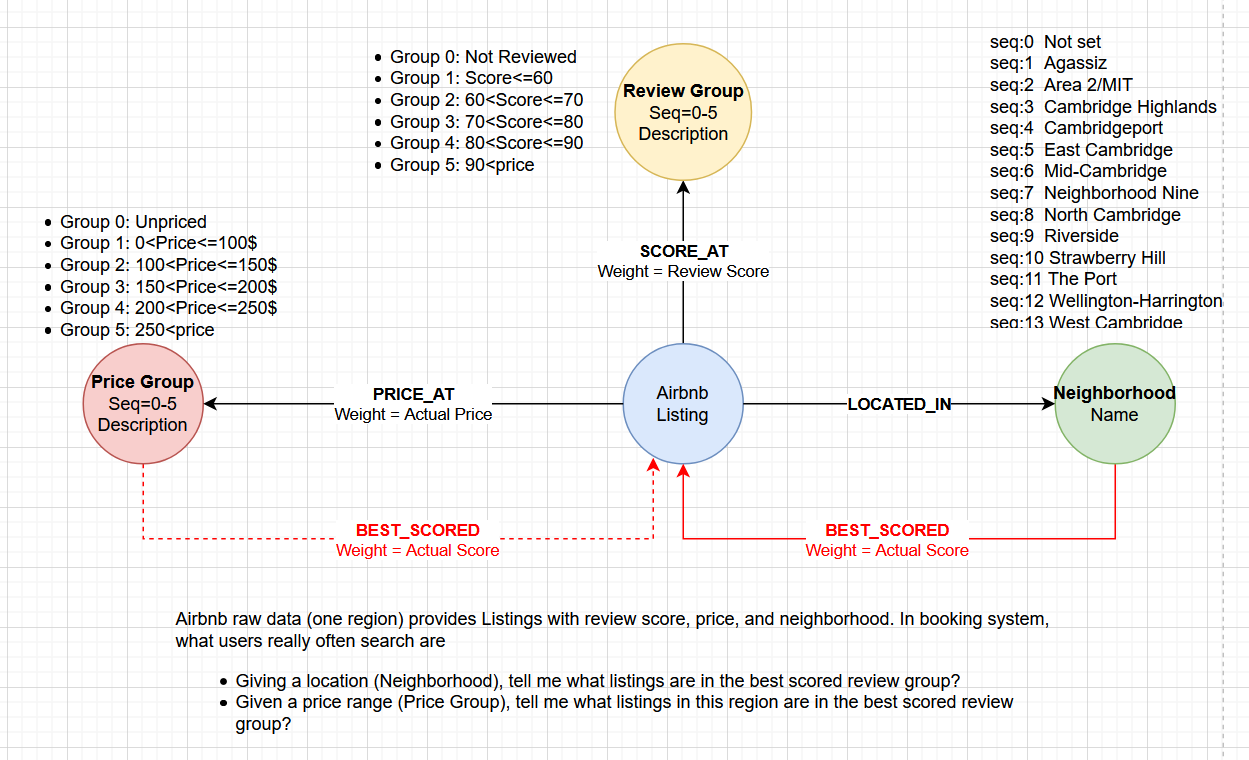
# Model & Data Preparation



## (1) Review Group

CREATE CONSTRAINT pk\_ReviewGroup ON (r:ReviewGroup) ASSERT r.seq IS UNIQUE;

LOAD CSV WITH HEADERS FROM 'file:///ReviewGroup.csv' AS row

  WITH toInteger(row.seq) as v\_seq, row.description as v\_description

  MERGE(S:ReviewGroup {seq: v\_seq}) SET S.description=v\_description RETURN COUNT(S);

# Query: MATCH(r:ReviewGroup) RETURN r;

# Delete: MATCH(r:ReviewGroup) DETACH DELETE r;

## (2) Price Group

CREATE CONSTRAINT pk\_PriceGroup ON (p:PriceGroup) ASSERT p.seq IS UNIQUE;

LOAD CSV WITH HEADERS FROM 'file:///PriceGroup.csv' AS row

  WITH toInteger(row.seq) as v\_seq, row.description as v\_description

  MERGE(P:PriceGroup {seq: v\_seq}) SET P.description=v\_description RETURN COUNT(P);

# Query: MATCH(p:PriceGroup) RETURN p;

# Delete: MATCH (n:PriceGroup) DETACH DELETE n;

## (3) Neighbourhood

CREATE CONSTRAINT pk\_Neighbourhood ON (n:Neighbourhood) ASSERT n.seq IS UNIQUE;

LOAD CSV WITH HEADERS FROM 'file:///Neighbourhood.csv' AS row

  WITH toInteger(row.seq) as v\_seq, row.name as v\_name

  MERGE(N: Neighbourhood {seq: v\_seq}) SET N.name=v\_name RETURN count(N);

# Query: MATCH(n:Neighbourhood) RETURN n;

# Delete: MATCH(n:Neighbourhood) DETACH DELETE n;

## (4) Listing

LOAD CSV WITH HEADERS FROM 'file:///Listing.csv' AS row

WITH toInteger(row.id) as ListingID, toInteger(row.host\_id) as HostID, row.name as ListingName

MERGE(L:Listing {id: ListingID}) SET L.name=ListingName, L.host\_id=HostID

RETURN COUNT(L);

## (5) Edges of PRICE\_AT between Listings and Price Groups

LOAD CSV WITH HEADERS FROM 'file:///Listing.csv' AS row

WITH toInteger(row.id) as ListingID, toInteger(row.PriceSeq) as nPriceSeq, toInteger(row.Price) as nPrice

MATCH (l:Listing),(p:PriceGroup) WHERE l.id=ListingID AND p.seq=nPriceSeq

CREATE (l)-[r:PRICE\_AT {weight: nPrice}]->(p) RETURN COUNT(r);

# match (l:Listing)-[r:PRICE\_AT]->(p:PriceGroup) delete r;

## (6) Edges of SCORE\_AT between Listings and Review Groups

LOAD CSV WITH HEADERS FROM 'file:///Listing.csv' AS row

WITH toInteger(row.id) as ListingID, toInteger(row.ReviewSeq) as nReviewSeq, toInteger(row.ReviewScore) as Score

MATCH (l:Listing),(s:ReviewGroup) WHERE l.id=ListingID AND s.seq=nReviewSeq

CREATE (l)-[r:SCORE\_AT {weight: Score}]->(s) RETURN COUNT(r);

# match (l:Listing)-[r:SCORE\_AT]->(s:ReviewGroup) delete r;

## (7) Edges of LOCATED\_IN between Listings and Neighbourhood

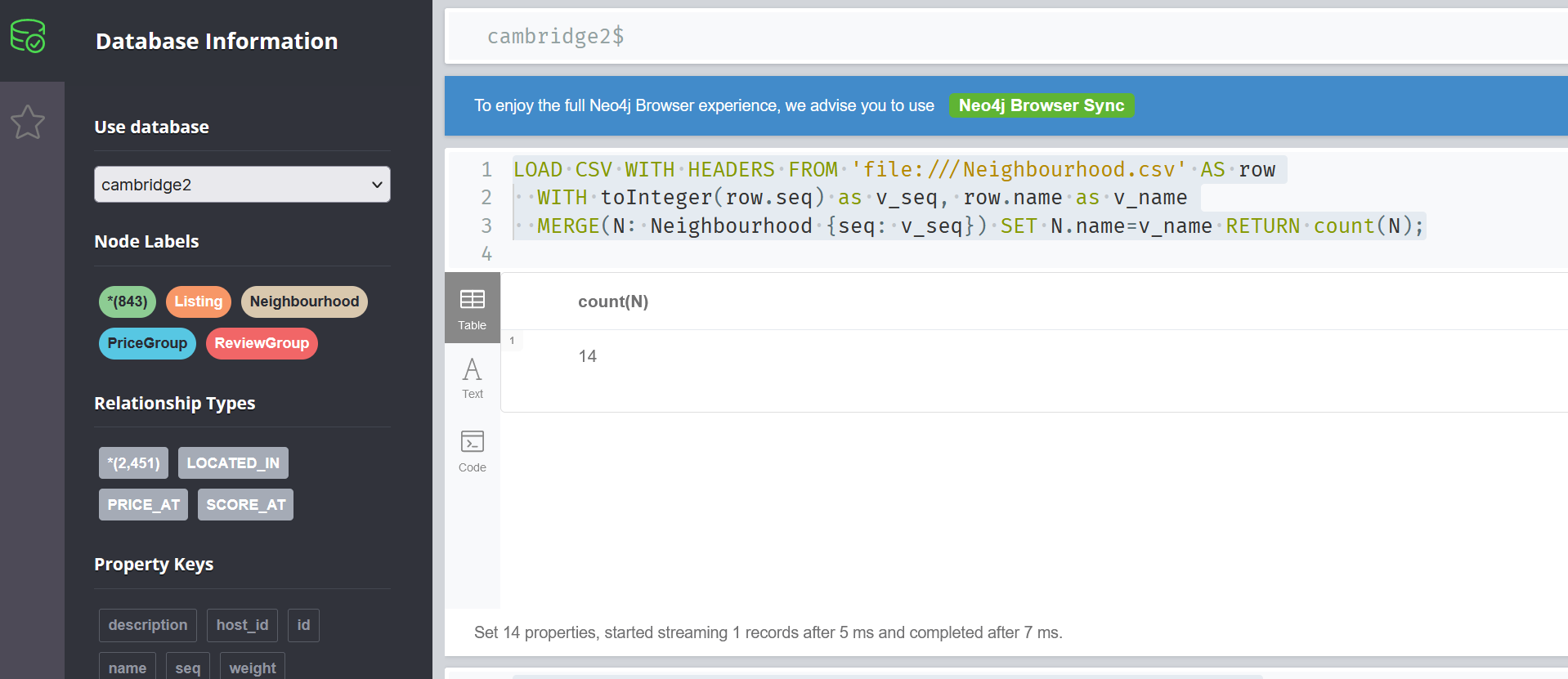
LOAD CSV WITH HEADERS FROM 'file:///Listing.csv' AS row

WITH toInteger(row.id) as ListingID, toInteger(row.NBSeq) as nNBSeq

MATCH (l:Listing),(n:Neighbourhood) WHERE l.id=ListingID AND n.seq=nNBSeq

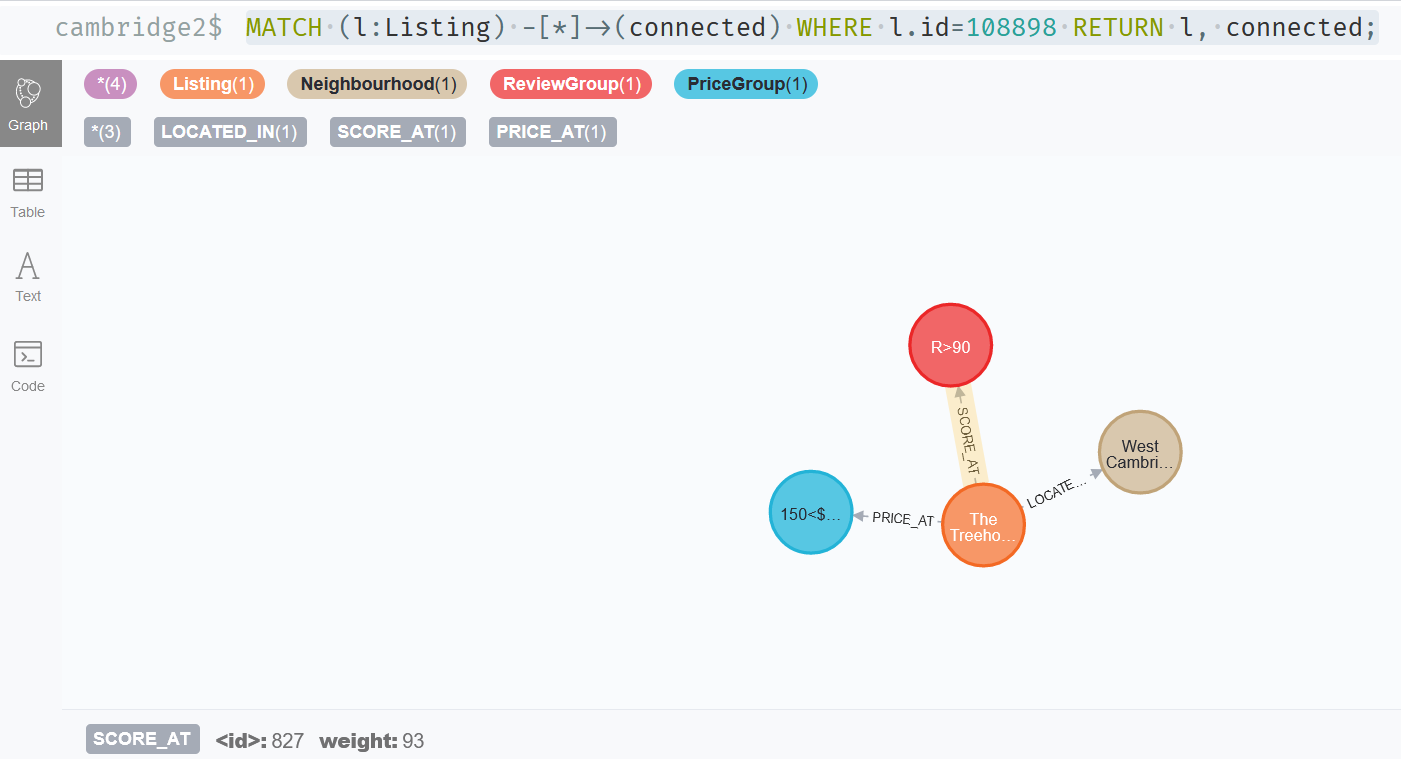
CREATE (l)-[r:LOCATED\_IN]->(n) RETURN COUNT(r);

# match (l:Listing)-[r:LOCATED\_IN]->(n:Neighbourhood) delete r;



## (8) Verification: given one listing, show its raw data relationships

MATCH (l:Listing) -[\*]->(connected) WHERE l.id=108898 RETURN l, connected;



# Cypher Actions

## (1) Given a location (Neighbourhood), show me the listings in the best review score group, so I can consider them on higher priority.

### Method one: Pure Cypher with multiple hops

with 'Riverside' as sNB

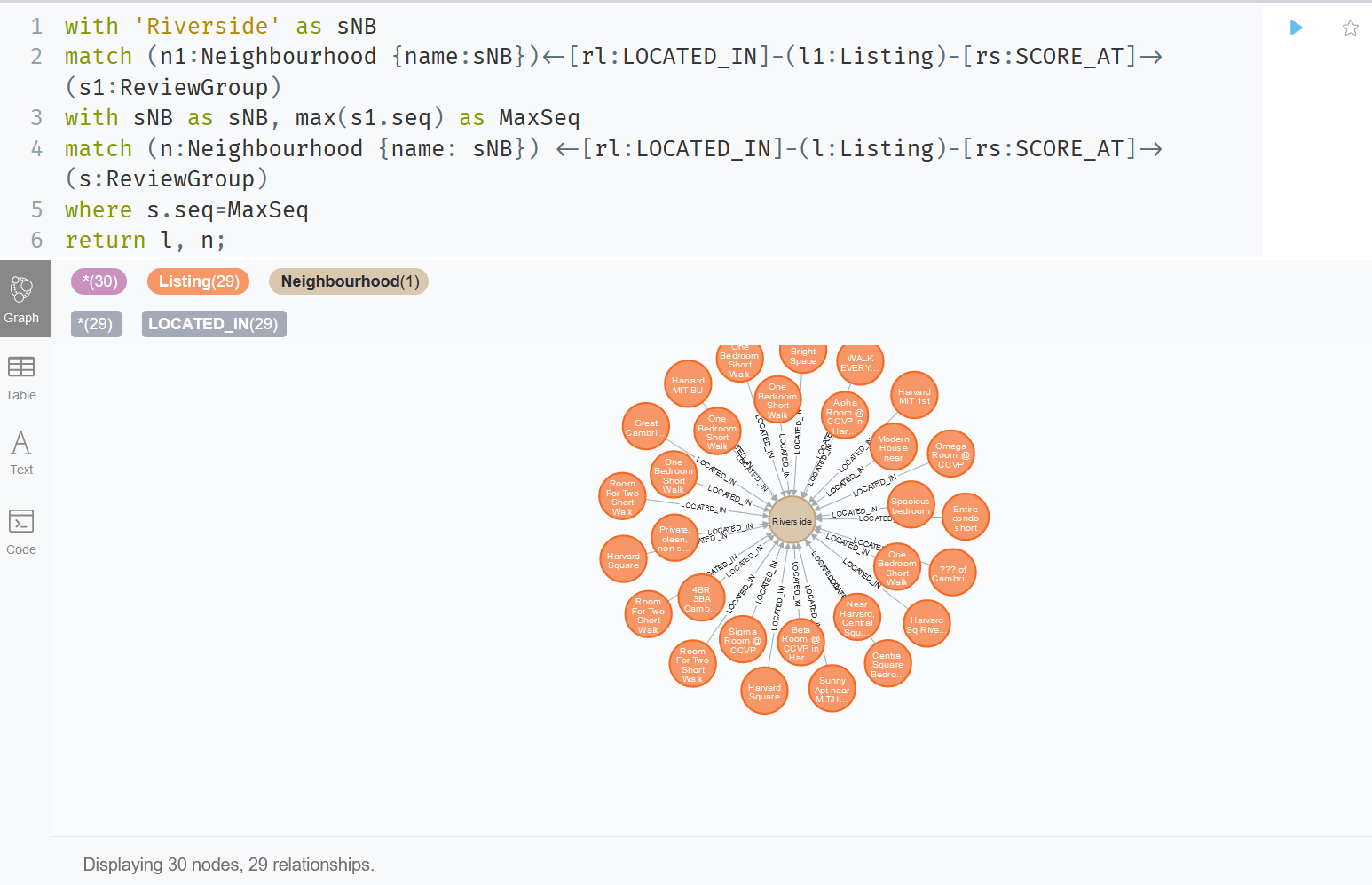
match (n1:Neighbourhood {name:sNB})<-[rl:LOCATED\_IN]-(l1:Listing)-[rs:SCORE\_AT]->(s1:ReviewGroup)

with sNB as sNB, max(s1.seq) as MaxSeq

match (n:Neighbourhood {name: sNB}) <-[rl:LOCATED\_IN]-(l:Listing)-[rs:SCORE\_AT]->(s:ReviewGroup)

where s.seq=MaxSeq

return  l, n;



Second match to find out all the listings linked with the neighbourhood and the MAX Review Group

First match to find out MAX Review Score Group which is the “Best Score”

Neighbourhood name is used in multiple places, use a variable for input once

### Method two: Use Neighbourhood.csv as a loop control to project a new set of relationship “BEST\_SCORED” from Neighbourhood back to Listings, then query with one hop.

LOAD CSV WITH HEADERS FROM 'file:///Neighbourhood.csv' AS row

  WITH toInteger(row.seq) as nb\_seq

  match (n1:Neighbourhood {seq:nb\_seq})<-[rl:LOCATED\_IN]-(l1:Listing)-[rs:SCORE\_AT]->(s1:ReviewGroup)

  with n1.seq as nb\_seq, max(s1.seq) as MaxScoreSeq

  match (n:Neighbourhood {seq: nb\_seq}) <-[rl:LOCATED\_IN]-(l:Listing)-[rs:SCORE\_AT]->(s:ReviewGroup)

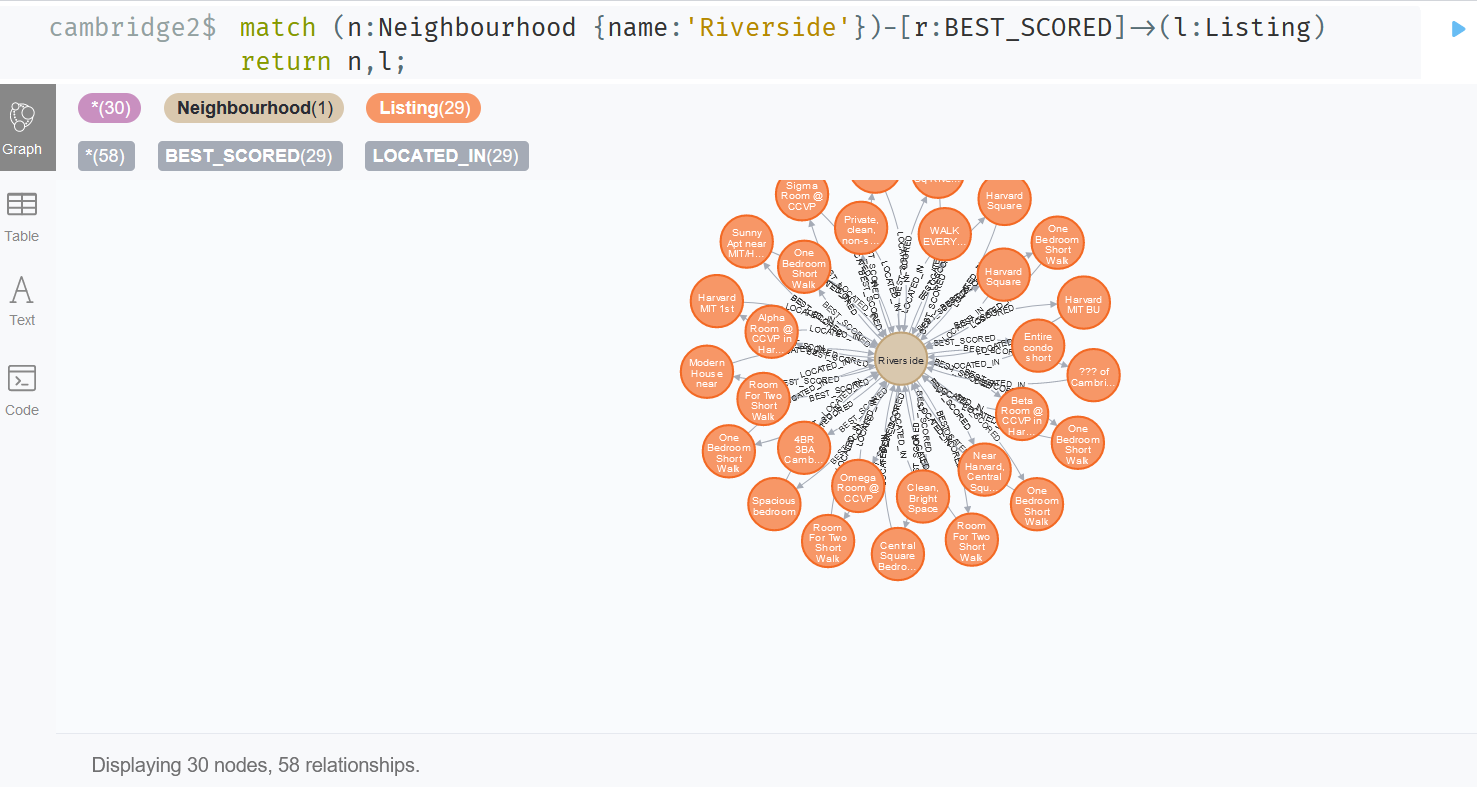
  where s.seq=MaxScoreSeq

  CREATE (n)-[r:BEST\_SCORED {weight: rs.weight}]->(l)

  RETURN COUNT(r);

match (n:Neighbourhood {name:'Riverside'})-[r:BEST\_SCORED]->(l:Listing) return n,l;





## (2) Given a price (Price Group), show me the listings within worst review score group, so I can avoid to book them.

with '1<$≤100' as sPG

MATCH (p:PriceGroup {description: sPG}) <-[rl:PRICE\_AT]-(l1:Listing)-[rs:SCORE\_AT]->(s1:ReviewGroup)

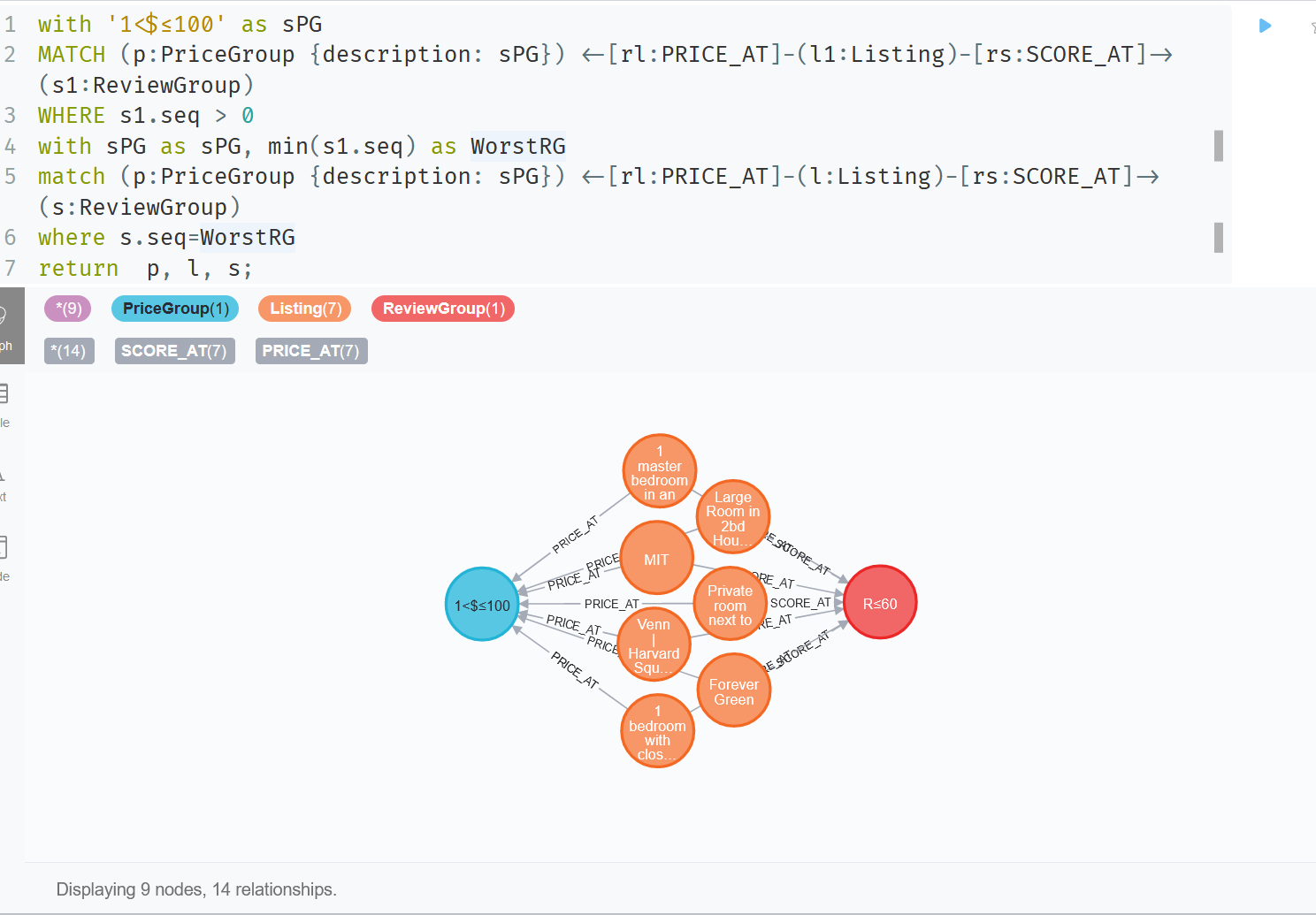
WHERE s1.seq > 0

with sPG as sPG, min(s1.seq) as WorstRG

match (p:PriceGroup {description: sPG}) <-[rl:PRICE\_AT]-(l:Listing)-[rs:SCORE\_AT]->(s:ReviewGroup)

where s.seq=WorstRG

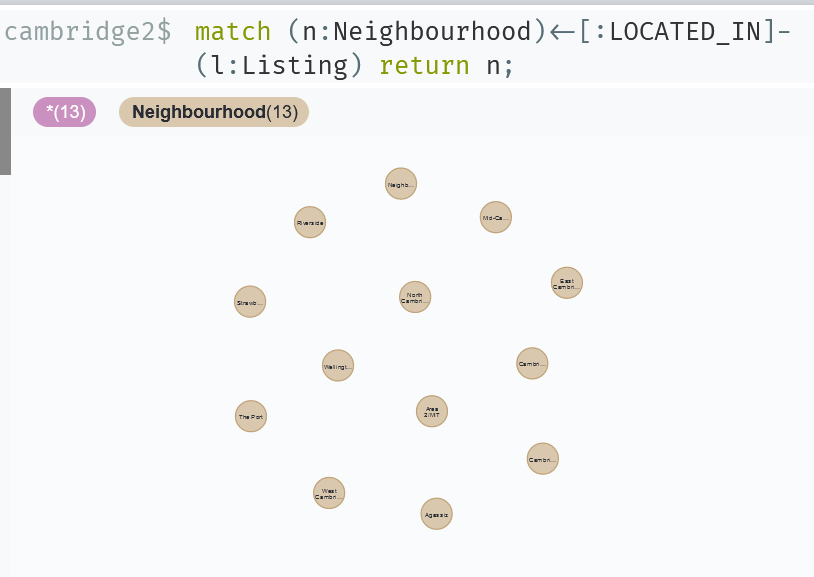
return  p, l, s;



We can adopt the same as previous question second solution to project a set of relationships to save hops, ignore it in here.

# Visualization

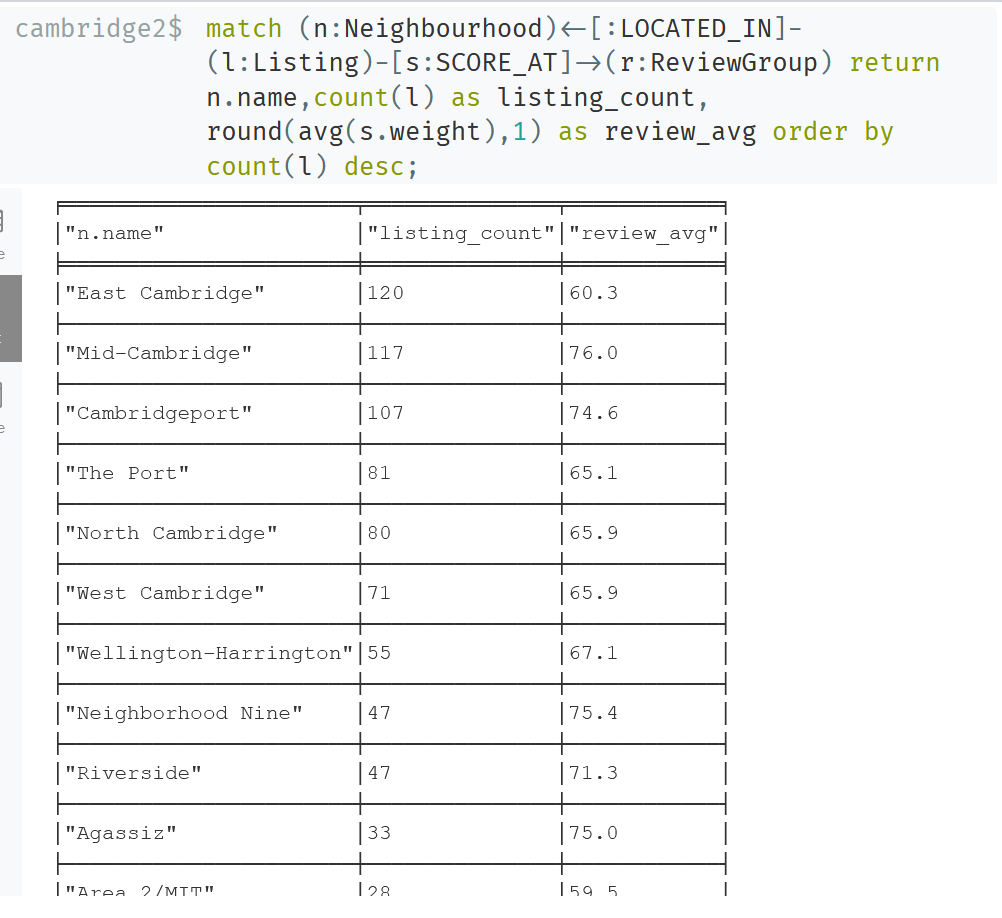
Totally there are 14 geographically spread neighbourhood in Cambridge, MA for Airbnb to detail divides the city level regional market. Below shows the neighbourhood along with their listing count.



Associated with the neighbourhood are the listing count and review score average

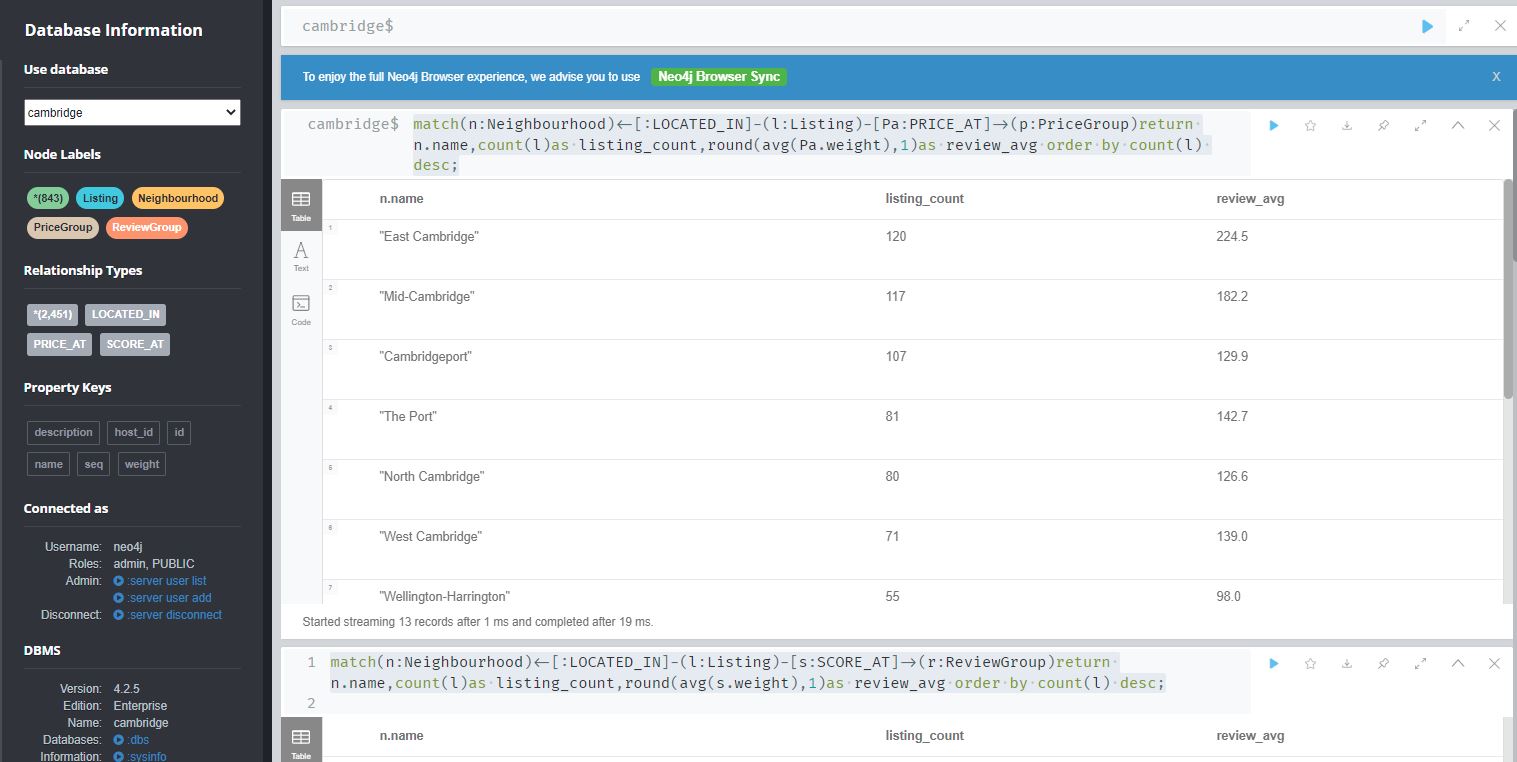
# Visualization1

match(n:Neighbourhood)<-[:LOCATED\_IN]-(l:Listing)-[s:SCORE\_AT]->(r:ReviewGroup)return n.name,count(l)as listing\_count,round(avg(s.weight),1)as review\_avg order by count(l) desc;

<

# Visualization 2

match(n:Neighbourhood)<-[:LOCATED\_IN]-(l:Listing)-[Pa:PRICE\_AT]->(p:PriceGroup)return n.name,count(l)as listing\_count,round(avg(Pa.weight),1)as review\_avg order by count(l) desc;



Among the top three large neighbourhood in terms of listing numbers, the East Cambridge is carrying the almost lowest review score. From business point of view, this is the one needs more attention, actions should be taken:

* Investigating the reason for low satisfaction;
* Giving more training to the owners in this area;
* Tightening the acceptance criteria of new listings;

Follow-up analysis could be performed with future 6 month data compare with this result to verify if the actions are effective or not.