C.1. Database Design

MongoDB:

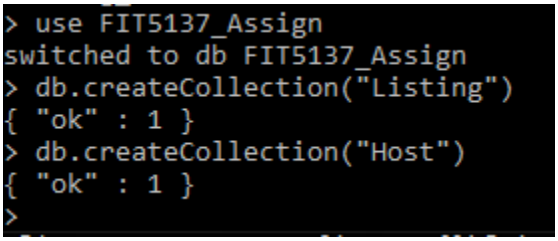
Create a database called *FIT5137_Assign*.

1. use FIT5137_Assign

Create collections for *Listing* and *Host*.

2. db.createCollection("Listing")

   db.createCollection("Host")



Insert the data provided in Table 1 and 3 into the collections

db.Host.insertMany([
{"host_id":"MONHOS01","host_name":"Manju","host_verifications":["email","phone","reviews"],"host_since":Date("21/08/2009"),"host_location":{"suburb":"Clayton","state":"Victoria","Country":"Australia"},"host_response_time":"within a day","host_is_superhost":false},
{"host_id":"MONHOS02","host_name":"Lindsay","host_verifications":["email","phone","reviews","jumio","government id"],"host_since":Date("16/09/2009"),"host_location":{"suburb":"Clifton Hill","state":"Victoria","Country":"Australia"},"host_response_time":"within an hour","host_is_superhost":true},
{"host_id":"MONHOS03","host_name":"Adam","host_verifications":["email","phone","google","reviews","jumio","government id","work email"],"host_since":Date("31/10/2009"),"host_location":{"suburb":"Port Melbourne","state":"Victoria","Country":"Australia"},"host_response_time":"within an hour","host_is_superhost":false},
{"host_id":"MONHOS04","host_name":"Eleni","host_verifications":["email","phone","facebook","reviews","jumio","offline government id","government id","work email"],"host_since":Date("3/12/2009"),"host_location":{"suburb":"Fitzroy","state":"Victoria","Country":"Australia"},"host_response_time":"within a day","host_is_superhost":false},

{"host_id":"MONHOS05","host_name":"Colin","host_verifications":["email","phone","face
book","reviews","jumio","offline government id","government
id"],"host_since":Date("22/12/2009"),"host_location":{"suburb":"Saint Kilda
East","state":"Victoria","Country":"Australia"},"host_response_time":"within an
hour","host_is_superhost":false},
{"host_id":"MONHOS06","host_name":"Daryl","host_verifications":["email","phone","man
ual online","reviews","manual offline","work
email"],"host_since":Date("12/07/2010"),"host_location":{"suburb":"Berwick","state":"Vict
oria","Country":"Australia"},"host_response_time":"within an
hour","host_is_superhost":true},
{"host_id":"MONHOS07","host_name":"Diana","host_verifications":["email","phone","fac
ebook","reviews","jumio","offline government id","government id","work
email"],"host_since":Date("27/07/2010"),"host_location":{"suburb":"Parkdale","state":"Vic
toria","Country":"Australia"},"host_response_time":"within a
day","host_is_superhost":false},
{"host_id":"MONHOS08","host_name":"Belinda","host_verifications":["email","phone","fa
cebook","reviews","jumio","offline government id","selfie","government id","identity
manual","work
email"],"host_since":Date("3/08/2009"),"host_location":{"suburb":"Prahran","state":"Victo
ria","Country":"Australia"},"host_response_time":"within a few
hours","host_is_superhost":false},
{"host_id":"MONHOS09","host_name":"Allan","host_verifications":["email","phone","face
book","reviews","jumio","offline government id","selfie","government id","identity
manual"],"host_since":Date("3/08/2009"),"host_location":{"suburb":"South
Melbourne","state":"Victoria","Country":"Australia"},"host_response_time":"within an
hour","host_is_superhost":true},
{"host_id":"MONHOS10","host_name":"Vicki","host_verifications":["email","phone","face
book","reviews","jumio","government
id"],"host_since":Date("6/08/2009"),"host_location":{"suburb":"Frankston","state":"Victori
a","Country":"Australia"},"host_response_time":"within an
hour","host_is_superhost":true}
])

db.Listing.insertMany([

{"listing_id":"MONLST01","name":"Monash Beautiful House","host_id":"MONHOS14","neighbourhood":"Manningham","address":{"suburb":"Clayton","state":"VIC","zipcode":3800},"latitude":-37.773,"longitude":145.09213,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":61,"extra_people":22,"minimum_nights":1,"availability_365":365},

{"listing_id":"MONLST02","name":"Monash Brunswick Deco","host_id":"MONHOS08","neighbourhood":"Moreland","address":{"suburb":"Brunswick East","state":"VIC","zipcode":3057},"latitude":-37.767,"longitude":144.98074,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":35,"extra_people":15,"minimum_nights":3,"availability_365":194},

{"listing_id":"MONLST03","name":"Monash Beachside Retreat","host_id":"MONHOS01","neighbourhood":"Port Phillip","address":{"suburb":"St Kilda","state":"VIC","zipcode":3182},"latitude":-37.86,"longitude":144.97737,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":159,"extra_people":0,"minimum_nights":2,"availability_365":82},

{"listing_id":"MONLST04","name":"Monash Close2City","host_id":"MONHOS04","neighbourhood":"Darebin","address":{"suburb":"Thornbury","state":"VIC","zipcode":3071},"latitude":-37.759,"longitude":144.98923,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":50,"extra_people":20,"minimum_nights":2,"availability_365":0},

{"listing_id":"MONLST05","name":"Monash City and Sports","host_id":"MONHOS05","neighbourhood":"Port Phillip","address":{"suburb":"St Kilda East","state":"VIC","zipcode":3183},"latitude":-37.865,"longitude":144.99224,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":69,"extra_people":20,"minimum_nights":1,"availability_365":274},

{"listing_id":"MONLST06","name":"Monash Trafford Apartment","host_id":"MONHOS06","neighbourhood":"Casey","address":{"suburb":"Berwick","state":"VIC","zipcode":3806},"latitude":-38.057,"longitude":145.33936,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":99,"extra_people":30,"minimum_nights":1,"availability_365":353},

{"listing_id":"MONLST07","name":"Monash Close2Airport","host_id":"MONHOS07","neighbourhood":"Darebin","address":{"suburb":"Reservoir","state":"VIC","zipcode":3073},"latitude":-37.697,"longitude":145.00082,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":50,"extra_people":20,"minimum_nights":7,"availability_365":0},

{"listing_id":"MONLST08","name":"Monash Home In The City","host_id":"MONHOS02","neighbourhood":"Melbourne","address":{"suburb":"East Melbourne","state":"VIC","zipcode":3002},"latitude":-37.81,"longitude":144.98592,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":99,"extra_people":25,"minimum_nights":15,"availability_365":62},

{"listing_id":"MONLST09","name":"Monash Japanese-Style","host_id":"MONHOS11","neighbourhood":"Monash","address":{"suburb":"Oakleigh East","state":"VIC","zipcode":3166},"latitude":-37.9,"longitude":145.11447,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":98,"extra_people":0,"minimum_nights":2,"availability_365":219},

{"listing_id":"MONLST10","name":"Beautiful Monash House","host_id":"MONHOS10","neighbourhood":"Frankston","address":{"suburb":"Frankston","state":"VIC","zipcode":3199},"latitude":-38.149,"longitude":145.14157,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":59,"extra_people":10,"minimum_nights":2,"availability_365":318},

{"listing_id":"MONLST11","name":"Fabulous Monash Richmond","host_id":"MONHOS09","neighbourhood":"Yarra","address":{"suburb":"Richmond","state":"VIC","zipcode":3121},"latitude":-37.818,"longitude":145.00442,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":98,"extra_people":30,"minimum_nights":14,"availability_365":16},

{"listing_id":"MONLST12","name":"Monash Central Lux","host_id":"MONHOS12","neighbourhood":"Port Phillip","address":{"suburb":"St Kilda","state":"VIC","zipcode":3182},"latitude":-37.861,"longitude":144.98038,"room_type":"Entire home","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":189,"extra_people":29,"minimum_nights":2,"availability_365":6},

{"listing_id":"MONLST13","name":"Central Monash Warehouse Apartment","host_id":"MONHOS13","neighbourhood":"Melbourne","address":{"suburb":"Melbourne","state":"VIC","zipcode":3000},"latitude":-37.815,"longitude":144.96267,"room_type":"Entire home","amenities":["TV","Wifi","Pets

Allowed","Family friendly","24-hour check-in","Self check-in"],"price":249,"extra_people":40,"minimum_nights":2,"availability_365":353},

{"listing_id":"MONLST14","name":"Monash Near the Park","host_id":"MONHOS03","neighbourhood":"Bayside","address":{"suburb":"Melbourne","state":"VIC","zipcode":3187},"latitude":-37.928,"longitude":145.02518,"room_type":"Private room","amenities":["TV","Wifi","Pets Allowed","Family friendly","24-hour check-in","Self check-in"],"price":40,"extra_people":11,"minimum_nights":2,"availability_365":365},

])

Cassandra:

1. CREATE KEYSPACE FIT5137_Assign WITH replication={'class':'SimpleStrategy','replication_factor':1};
2. CREATE TABLE IF NOT EXISTS FIT5137_Assign.review (
   listing_id text,
   review_id text,
   review_date_time timestamp,
   reviewer_id int,
   reviewer_name text,
   review_scores int,
   satisfied_reason set<text>,
   comments text,
   PRIMARY KEY (review_id)
   );

```
cqlsh> use FIT5137_Assign;
cqlsh:fit5137_assign> describe keyspace;

CREATE KEYSPACE fit5137_assign WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_wr
ites = true;

CREATE TABLE fit5137_assign.review (
    review_id text PRIMARY KEY,
    comments text,
    listing_id text,
    review_date_time timestamp,
    review_scores int,
    reviewer_id int,
    reviewer_name text,
    satisfied_reason set<text>
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
```

3.

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV01','MONLST02','2017-03-22 10:37:50+1300',500001,'Miriam',90,{'location','amenities'},'Beautiful View');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV02','MONLST02','2017-03-22 11:37:50+1300',500002,'Johannes',90,{'host'},'Good Host');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV03','MONLST02','2017-03-22 11:37:50+1300',500003,'Camille',100,{'location','view'},'Nice View and Location');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV04','MONLST02','2017-03-22
12:37:50+1300',500004,'Paige',95,{'price'},'Excellent Price');

INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV05','MONLST01','2017-03-22
15:37:50+1300',500005,'Adele',93,{'location','price'},'Good Location');

INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV06','MONLST03','2017-03-22
17:37:50+1300',500006,'Greg',87,{'host','view'},'Very Clean House');

INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV07','MONLST04','2017-03-22
19:37:50+1300',500007,'Wolfgang',91,{'location','price'},'Nice Location');

INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV08','MONLST05','2017-03-22
20:37:50+1300',500008,'Klaus',96,{'location','view'},'Nice Building');

INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV09','MONLST06','2017-03-23
11:37:50+1300',500009,'Rox',100,{'host','price'},'Friendly Host');

```
INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV10','MONLST05','2017-03-23
12:37:50+1300',500010,'Elisabeth',98,{'host','price'},'Friendly Host');


INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV11','MONLST09','2017-03-23
19:37:50+1300',500011,'Derek',100,{'space','clean'},'Very Clean and omfortable');


INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV12','MONLST09','2017-03-25
10:07:40+1300',500012,'Joy',92,{'host','clean'},'Friendly and Nice Host');


INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV13','MONLST10','2017-03-26
10:02:10+1300',500013,'Anouck',93,{'host','view'},'Very Comfortable');


INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV14','MONLST12','2017-03-26
10:49:40+1300',500014,'Jerome',85,{'location','clean'},'Friendly Host');


INSERT INTO review
(review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)
```

VALUES ('REV15','MONLST13','2017-03-26 10:48:40+1300',500015,'Jehan',98,{'location','amenities'},'Beautiful View');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV16','MONLST14','2017-03-26 10:48:10+1300',500012,'Joy',97,{'amenities','view'},'Good Location');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV17','MONLST14','2017-03-26 10:47:40+1300',500014,'Jerome',30,{'price','view'},'Bad Location');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV18','MONLST10','2017-03-26 10:47:10+1300',500002,'Johannes',20,{'amenities','view'},'Bad Service');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV19','MONLST12','2017-03-27 09:37:50+1300',500013,'Anouck',87,{'space','clean'},'Good Location');

INSERT INTO review (review_id,listing_id,review_date_time,reviewer_id,reviewer_name,review_scores,satisfied_reason,comments)

VALUES ('REV20','MONLST05','2017-03-27 10:45:10+1300',500011,'Derek',96,{'host','view'},'Nice Building');

```
cqlsh:fit5137_assign> select * from review;

 review_id | comments                 | listing_id | review_date_time                | review_scores | reviewer_id | reviewer_name | satisfied_reason
-----------+--------------------------+------------+---------------------------------+---------------+-------------+---------------+-------------------------
     REV03 |    Nice View and Location |   MONLST02 | 2017-03-21 22:37:50.000000+0000 |           100 |      500003 |       Camille |      {'location', 'view'}
     REV19 |             Good Location |   MONLST12 | 2017-03-26 20:37:50.000000+0000 |            87 |      500013 |        Anouck |       {'clean', 'space'}
     REV10 |             Friendly Host |   MONLST05 | 2017-03-22 23:37:50.000000+0000 |            98 |      500010 |     Elisabeth |        {'host', 'price'}
     REV06 |          Very Clean House |   MONLST03 | 2017-03-22 04:37:50.000000+0000 |            87 |      500006 |          Greg |         {'host', 'view'}
     REV13 |          Very Comfortable |   MONLST10 | 2017-03-25 21:02:10.000000+0000 |            93 |      500013 |        Anouck |         {'host', 'view'}
     REV15 |            Beautiful View |   MONLST13 | 2017-03-25 21:48:40.000000+0000 |            98 |      500015 |         Jehan | {'amenities', 'location'}
     REV08 |            Nice Building  |   MONLST05 | 2017-03-22 07:37:50.000000+0000 |            96 |      500008 |         Klaus |      {'location', 'view'}
     REV02 |                Good Host  |   MONLST02 | 2017-03-21 22:37:50.000000+0000 |            98 |      500002 |      Johannes |                {'host'}
     REV11 | Very Clean and comfortable |   MONLST09 | 2017-03-23 06:37:50.000000+0000 |           100 |      500011 |         Derek |       {'price', 'space'}
     REV20 |            Nice Building  |   MONLST05 | 2017-03-26 21:45:10.000000+0000 |            96 |      500011 |         Derek |         {'host', 'view'}
     REV14 |             Friendly Host |   MONLST12 | 2017-03-25 21:49:40.000000+0000 |            85 |      500014 |       Jerome |      {'clean', 'location'}
     REV09 |             Friendly Host |   MONLST06 | 2017-03-22 22:37:50.000000+0000 |           100 |      500009 |           Rox |        {'host', 'price'}
     REV12 |   Friendly and Nice Host  |   MONLST09 | 2017-03-24 21:07:40.000000+0000 |            92 |      500012 |           Joy |        {'clean', 'host'}
     REV07 |            Nice Location  |   MONLST04 | 2017-03-22 06:37:50.000000+0000 |            91 |      500007 |       Wolfgang |     {'location', 'price'}
     REV05 |             Good Location |   MONLST01 | 2017-03-22 02:37:50.000000+0000 |            93 |      500005 |         Adele |     {'location', 'price'}
     REV01 |            Beautiful View |   MONLST02 | 2017-03-21 21:37:50.000000+0000 |            98 |      500001 |        Miriam | {'amenities', 'location'}
     REV16 |             Good Location |   MONLST14 | 2017-03-25 21:48:10.000000+0000 |            97 |      500012 |           Joy |     {'amenities', 'view'}
     REV18 |               Bad Service |   MONLST10 | 2017-03-25 21:47:10.000000+0000 |            20 |      500002 |      Johannes |     {'amenities', 'view'}
     REV04 |           Excellent Price |   MONLST02 | 2017-03-21 23:37:50.000000+0000 |            95 |      500004 |         Paige |                {'price'}
     REV17 |               Bad Location |   MONLST14 | 2017-03-25 21:47:40.000000+0000 |            30 |      500014 |        Jerome |       {'price', 'view'}
```

C.2. Database Modifications.

1.

db.Host.updateOne(

{"host_name": "Adam"},

{$set:{"host_verifications":["email","phone","google","reviews","jumio","government id","work email","facebook"]},

$currentDate: { lastModified: true }

}

)

```
> db.Host.updateOne(
... {"host_name": "Adam"},
... {$set:{"host_verifications":["email","phone","google","reviews","jumio","government id","work email","facebook"]},
... $currentDate: { lastModified: true }
... }
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> _
```

2.

db.Host.insertMany([

{"host_id":"MONHOS11","host_name":"Alison","host_verifications":["email","phone","facebook","reviews"],"host_since":Date("9/1/2019"),"host_location":{"suburb":"Caulfield","state":"Victoria","Country":"Australia"},"host_response_time":"within an hour","host_is_superhost":false},

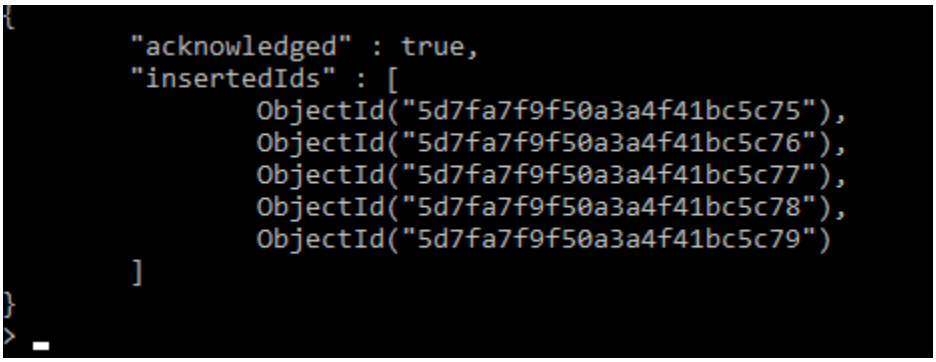
{"host_id":"MONHOS12","host_name":"Mike","host_verifications":["email","phone"],"host_since":Date("9/1/2019"),"host_location":{"suburb":"Clayton","state":"Victoria","Country":"Australia"},"host_response_time":"within a day","host_is_superhost":true},

{"host_id":"MONHOS13","host_name":"Robyn","host_verifications":["facebook","reviews"],"host_since":Date("9/1/2019"),"host_location":{"suburb":"Berwick","state":"Victoria","Country":"Australia"},"host_response_time":"within an hour","host_is_superhost":false},

{"host_id":"MONHOS14","host_name":"Daniel","host_verifications":["email","namual offline","work email"],"host_since":Date("9/1/2019"),"host_location":{"suburb":"Frankston","state":"Victoria","Country":"Australia"},"host_response_time":"within a day","host_is_superhost":true},

{"host_id":"MONHOS15","host_name":"Ron","host_verifications":["facebook"],"host_since":Date("9/1/2019"),"host_location":{"suburb":"Caulfield","state":"Victoria","Country":"Australia"},"host_response_time":"within a day","host_is_superhost":false},

])



3.

db.Host.updateMany(

{"host_response_time": {$eq:"within an hour"}},

{$set:{"host_is_superhost":true},

$currentDate:{lastModified: true}

}

)

```
> db.Host.updateMany(
... {"host_response_time": {$eq:"within an hour"}},
... {$set:{"host_is_superhost":true},
... $currentDate:{lastModified: true}
... }
... )
{ "acknowledged" : true, "matchedCount" : 8, "modifiedCount" : 8 }
>
```

4.

db.Listing.deleteMany(

{"availability_365":0})

```
> db.Listing.deleteMany(
... {"availability_365":0})
{ "acknowledged" : true, "deletedCount" : 2 }
>
```

5.

db.Listing.update(

{"neighbourhood":"Monash"},

{$set:{"neighbourhood":"Monash City"},

$currentDate:{lastModified: true}

}

)

```
> db.Listing.update(
... {"neighbourhood":"Monash"},
... {$set:{"neighbourhood":"Monash City"},
... $currentDate:{lastModified: true}
... }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

6.

UPDATE review

SET satisfied_reason = {'space','price'}

WHERE review_id = 'REV11';

```
cqlsh:fit5137_assign> UPDATE review
                   ... SET satisfied_reason = {'space','price'}
                   ... WHERE review_id = 'REV11';
cqlsh:fit5137_assign> select satisfied_reason from review where review_id='REV11';

 satisfied_reason
--------------------
 {'price', 'space'}
```

7.

DELETE comments FROM review WHERE reviewer_id=500015;

C.3. Queries.

Importing JSON files to mongodb:

mongoimport --db FIT5137_Assign_C3 --collection Host –file C:\Users\User\Desktop\FIT5137\host.json

mongoimport --db FIT5137_Assign_C3 --collection Listing –file C:\Users\User\Desktop\FIT5137\listing.json

Creating Embedding Model:

Creating ***listing_host*** collection and insert data into it:

db.createCollection("listing_host")

db.listing_host.insertMany(

db.Listing.aggregate([

{

$lookup:{

from:"Host",

localField:"host_id",

foreignField:"host_id",

as:"hosts"

}

}

]).pretty().toArray()

)

```
> db.createCollection("listing_host")
{ "ok" : 1 }
> db.listing_host.insertMany(
... db.Listing.aggregate([
... {
... $lookup:{
... from:"Host",
... localField:"host_id",
... foreignField:"host_id",
... as:"hosts"
... }
... }
... ]).pretty().toArray()
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5d7ef395a21b89bacd82456c"),
                ObjectId("5d7ef395a21b89bacd82456d"),
                ObjectId("5d7ef395a21b89bacd82456e"),
                ObjectId("5d7ef395a21b89bacd82456f"),
                ObjectId("5d7ef395a21b89bacd824570"),
                ObjectId("5d7ef395a21b89bacd824571"),
                ObjectId("5d7ef395a21b89bacd824572"),
                ObjectId("5d7ef395a21b89bacd824573"),
                ObjectId("5d7ef395a21b89bacd824574"),
                ObjectId("5d7ef395a21b89bacd824575"),
```

1. How many accommodations were last reviewed in December 2018?

Embedding Model:

db.listing_host.find({"last_review":{$in:[ISODate("2018-12
01T00:00:00Z"),ISODate("2018-12-31T00:00:00Z")]}}).count()

```
> db.listing_host.find({"last_review":{$in:[ISODate("2018-12-01T00:00:00Z"),ISODate("2018-12-31T00:00:00Z")]}}).count()
1
> db.Listing.find({"last_review":{$in:[ISODate("2018-12-01T00:00:00Z"),ISODate("2018-12-31T00:00:00Z")]}}).count()
1
```

Referencing Model:

db.Listing.find({"last_review":{$in:[ISODate("2018-12-01T00:00:00Z"),ISODate("2018-12-31T00:00:00Z")]}}).count()

2. What is the average price for the accommodations in the Port Phillip neighbourhood?

Embedding Model:

db.listing_host.aggregate([

{$match:{neighbourhood:"Port Phillip"}},

{$group:{_id:"$neighbourhood",average_price:{$avg:"$price"}}}

])

```
> db.listing_host.aggregate([
... {$match:{neighbourhood:"Port Phillip"}},
... {$group:{_id:"$neighbourhood",average_price:{$avg:"$price"}}}
... ])
{ "_id" : "Port Phillip", "average_price" : 134.5 }
> db.Listing.aggregate([
... {$match:{"neighbourhood":"Port Phillip"}},
... {$group:{"_id":"$neighbourhood",avgPrice:{$avg:"$price"}}}
... ]).pretty();
{ "_id" : "Port Phillip", "avgPrice" : 134.5 }
>
```

Referencing Model:

db.Listing.aggregate([

{$match:{"neighbourhood":"Port Phillip" }},

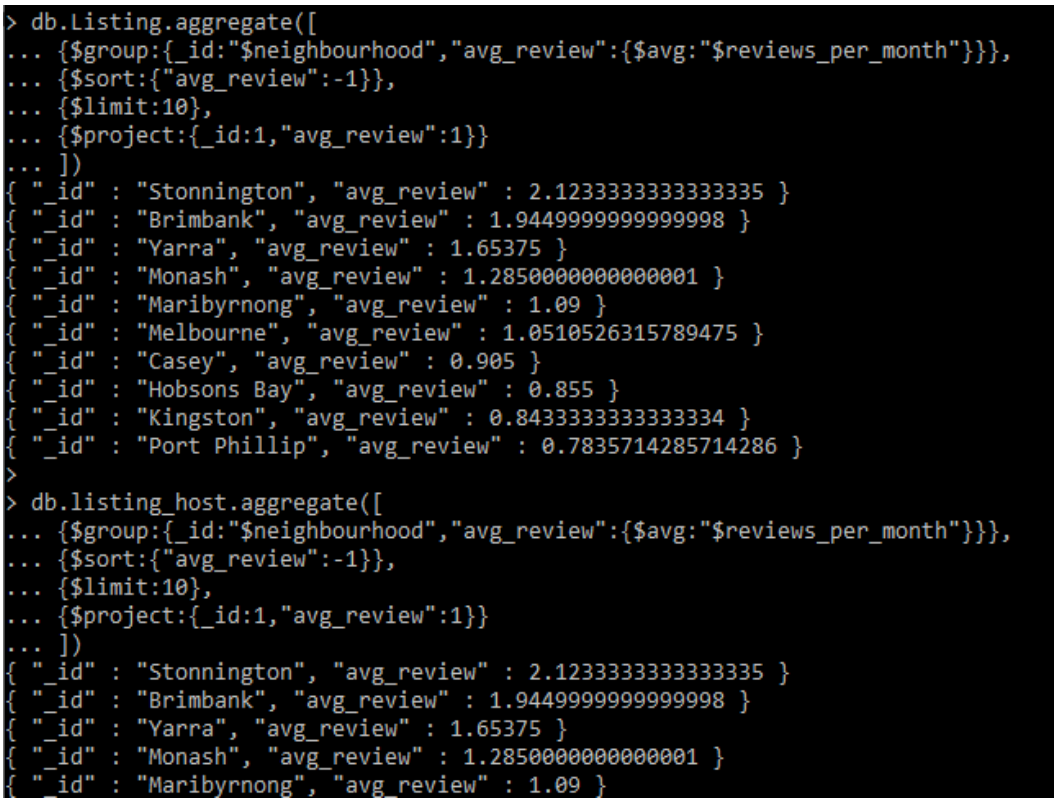
{$group:{"_id":"$neighbourhood",avgPrice:{$avg:"$price"}}}

]).pretty();

3. What are the top 10 most popular neighbourhoods based on the average reviews per month?

Embedding Model:

```
db.listing_host.aggregate([
{$group:{_id:"$neighbourhood","avg_review":{$avg:"$reviews_per_month"}}},
{$sort:{"avg_review":-1}},
{$limit:10},
{$project:{_id:1,"avg_review":1}}
])
```



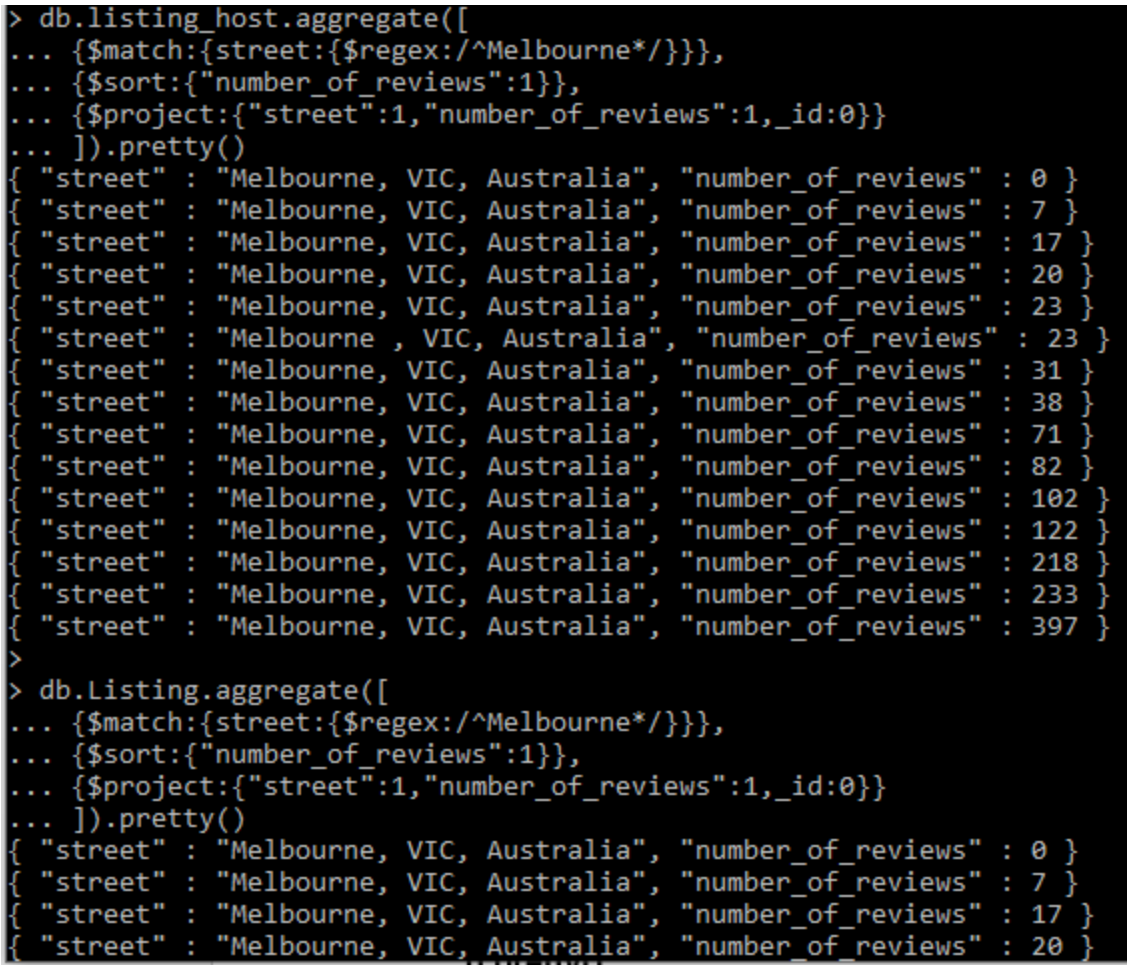
Referencing Model:

```
db.Listing.aggregate([
{$group:{_id:"$neighbourhood","avg_review":{$avg:"$reviews_per_month"}}},
{$sort:{"avg_review":-1}},
{$limit:10},
{$project:{_id:1,"avg_review":1}}
])
```

4. What is the range of number of accommodation reviewed (the lowest to the highest number) in Melbourne?

Embedding Model:

```
db.listing_host.aggregate([
{$match:{street:{$regex:/^Melbourne*/}}},
{$sort:{"number_of_reviews":1}},
{$project:{"street":1,"number_of_reviews":1,_id:0}}
]).pretty()
```



Referencing Model:

```
db.Listing.aggregate([
```

{$match:{street:{$regex:/^Melbourne*/}}},

{$sort:{"number_of_reviews":1}},

{$project:{"street":1,"number_of_reviews":1,_id:0}}

]).pretty()

5. What is the most popular room type?

Embedding Model:

db.listing_host.aggregate([

{$group:{_id:"$room_type",room_type:{$sum:1}}},

{$sort:{"room_type":-1}},

{$limit:1},

{$project:{_id:1 }}

])

```
> db.listing_host.aggregate([
... {$group:{_id:"$room_type",room_type:{$sum:1}}},
... {$sort:{"room_type":-1}},
... {$limit:1},
... {$project:{_id:1}}
... ])
{ "_id" : "Private room" }
>
> db.Listing.aggregate([
... ...  {$group:{_id:"$room_type",myCount:{$sum:1}}},
... ... {$sort:{myCount:-1}},
... ... {$limit:1},
... ... {$project:{_id:1}}
... ... ])
{ "_id" : "Private room" }
```

Referencing Model:

db.Listing.aggregate([

{$group:{_id:"$room_type",myCount:{$sum:1}}},

{$sort:{myCount:-1}},

{$limit:1},

{$project:{_id:1}}

])

6. Where are the top 5 most expensive accommodations?

Embedding Model:

```
db.listing_host.aggregate([
{$project:{_id:0,"street":1,"price":1}},
{$sort:{"price":-1}},
{$limit:5}
])
```

```
> db.listing_host.aggregate([
... {$project:{_id:0,"street":1,"price":1}},
... {$sort:{"price":-1}},
... {$limit:5}
... ])
{ "street" : "Southbank, VIC, Australia", "price" : 701 }
{ "street" : "Belgrave, VIC, Australia", "price" : 500 }
{ "street" : "Caulfield North, VIC, Australia", "price" : 419 }
{ "street" : "Southbank, VIC, Australia", "price" : 357 }
{ "street" : "Balaclava, VIC, Australia", "price" : 330 }
>
> db.Listing.aggregate([
... {$sort:{"price":-1}},
... {$limit:5},
... {$project:{street:1,_id:0,price:1}}
... ])
{ "street" : "Southbank, VIC, Australia", "price" : 701 }
{ "street" : "Belgrave, VIC, Australia", "price" : 500 }
{ "street" : "Caulfield North, VIC, Australia", "price" : 419 }
{ "street" : "Southbank, VIC, Australia", "price" : 357 }
{ "street" : "Balaclava, VIC, Australia", "price" : 330 }
>
```

Referencing Model:

```
db.Listing.aggregate([
{$sort:{"price":-1}},
{$limit:5},
{$project:{street:1,_id:0,price:1}}
])
```

7. Display all listings with host name “Eleni”.

Embedding Model:

db.listing_host.aggregate([

{$match:{"hosts.host_name":"Eleni"}}

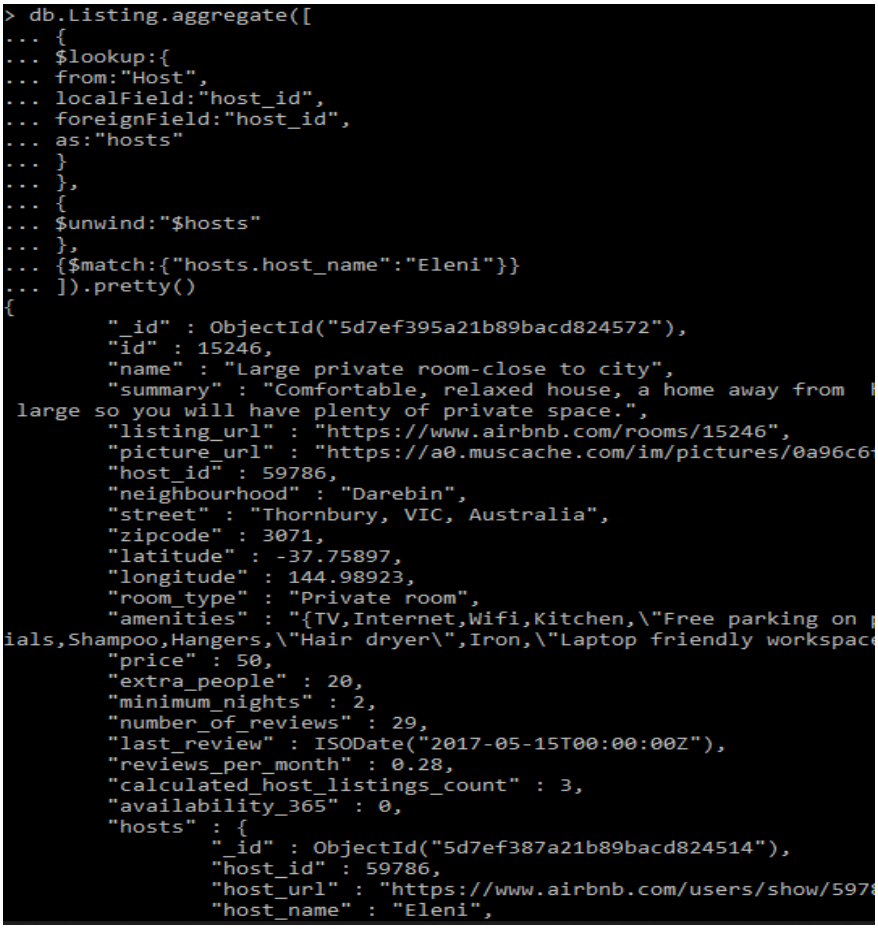
]).pretty()

Referencing Model:

db.Listing.aggregate([

{

```
        $lookup:{
                        from:"Host",
                        localField:"host_id",
                        foreignField:"host_id",
                        as:"hosts"
                        }
},
{
        $unwind:"$hosts"
},
{$match:{"hosts.host_name":"Eleni"}}
]).pretty()
```



8. Display the entire homes that the host can respond within a few hours.

Embedding Model:

```
db.listing_host.aggregate([
{$match:{"hosts.host_response_time":"within a few hours"}}
]).pretty()
```

```
> db.listing_host.aggregate([
... {$match:{"hosts.host_response_time":"within a few hours"}}
... ]).pretty()
{
        "_id" : ObjectId("5d7ef395a21b89bacd824573"),
        "id" : 67211,
        "name" : "Kew Tranquility, Melbourne",
        "summary" : "",
        "listing_url" : "https://www.airbnb.com/rooms/67211",
        "picture_url" : "https://a0.muscache.com/im/pictures/7b6d8c6d-
        "host_id" : 326880,
        "neighbourhood" : "Boroondara",
        "street" : "Kew, VIC, Australia",
        "zipcode" : 3101,
        "latitude" : -37.8037,
        "longitude" : 145.03396,
        "room_type" : "Private room",
        "amenities" : "{TV,Wifi,Kitchen,\"Paid parking off premises\",
tials,Hangers,\"Hair dryer\",Iron,\"Laptop friendly workspace\",\"Self
ator,\"Dishes and silverware\",\"Cooking basics\",Oven,Stove,\"Patio o
        "price" : 45,
        "extra_people" : 0,
        "minimum_nights" : 2,
        "number_of_reviews" : 160,
        "last_review" : ISODate("2019-06-27T00:00:00Z"),
        "reviews_per_month" : 1.56,
```

Referencing Model:

```
db.Listing.aggregate([
{
        $lookup:{
                        from:"Host",
                        localField:"host_id",
                        foreignField:"host_id",
                        as:"hosts"
                        }
```

},

{

$unwind:"$hosts"

},

{$match:{"hosts.host_response_time":"within an hour"}}

]).pretty()

```
> db.Listing.aggregate([
... {
... $lookup:{
... from:"Host",
... localField:"host_id",
... foreignField:"host_id",
... as:"hosts"
... }
... },
... {
... $unwind:"$hosts"
... },
... {$match:{"hosts.host_response_time":"within an hour"}}
... ]).pretty()
{
        "_id" : ObjectId("5d7ef395a21b89bacd82456d"),
        "id" : 10803,
        "name" : "Room in Cool Deco Apartment in Brunswick Eas
        "summary" : "A large air conditioned room with queen s
y. This area is known for its arts culture, live music, cafes
        "listing_url" : "https://www.airbnb.com/rooms/10803",
        "picture_url" : "https://a0.muscache.com/im/pictures/3
        "host_id" : 38901,
        "neighbourhood" : "Moreland",
        "street" : "Brunswick East, VIC, Australia",
        "zipcode" : 3057,
        "latitude" : -37.76651,
        "longitude" : 144.98074,
        "room_type" : "Private room",
        "amenities" : "{TV,Internet,Wifi,\"Air conditioning\",
Essentials,Shampoo,\"Lock on bedroom door\",\"24-hour check-in
```

9. Display the listing belongs to “Colin” with internet and gym access.

Embedding Model:

db.listing_host.aggregate([

{$unwind:"$hosts"},

{$match:{"hosts.host_name":"Colin"}},

{$match:{$or:[{amenities:{$regex:/Internet*/i}},{amenities:{$regex:/Gym*/i}}]}}]).pretty()

```
> db.listing_host.aggregate([
... {$unwind:"$hosts"},
... {$match:{"hosts.host_name":"Colin"}},
... {$match:{$or:[{amenities:{$regex:/Internet*/i}},{amenities:{$regex:/Gym*/i}}]}}]).pretty()
{
        "_id" : ObjectId("5d7ef395a21b89bacd82456f"),
        "id" : 16760,
        "name" : "Melbourne BnB near City & Sports",
        "summary" : "",
        "listing_url" : "https://www.airbnb.com/rooms/16760",
        "picture_url" : "https://a0.muscache.com/im/pictures/509974/5a9cf3d5_original.jpg?aki_polic
        "host_id" : 65090,
```

Referencing Model:

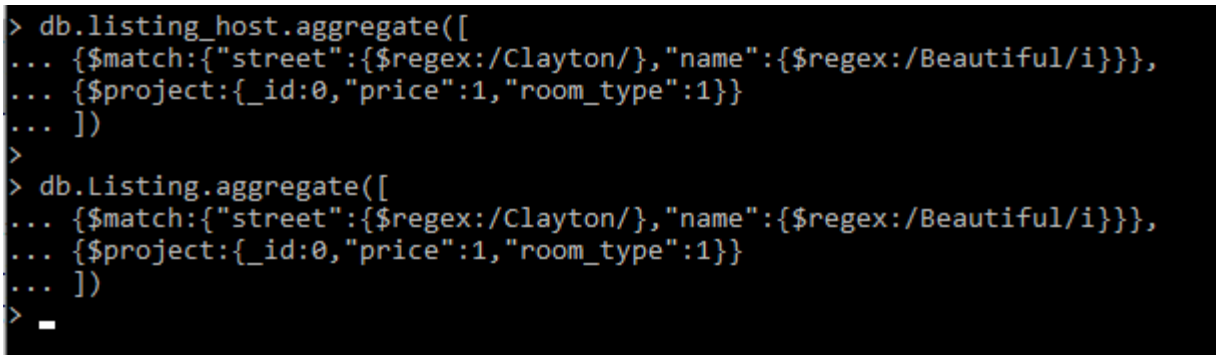
```
db.Listing.aggregate([
 {$lookup:{

        from:"Host",
        localField:"host_id",
        foreignField:"host_id",
        as:"hosts"
        }},
{$match:{"hosts.host_name":"Colin"}},
{$match:{$or:[{amenities:{$regex:/Internet*/i}},{amenities:{$regex:/Gym*/i}}]}}
]).pretty()
```

```
> db.Listing.aggregate([
...  {$lookup:{
...
... from:"Host",
... localField:"host_id",
... foreignField:"host_id",
... as:"hosts"
... }},
... {$match:{"hosts.host_name":"Colin"}},
... {$match:{$or:[{amenities:{$regex:/Internet*/i}},{amenities:{$regex:/Gym*/i}}]}}
... ]).pretty()
{
        "_id" : ObjectId("5d7ef395a21b89bacd82456f"),
        "id" : 16760,
        "name" : "Melbourne BnB near City & Sports",
        "summary" : "",
        "listing_url" : "https://www.airbnb.com/rooms/16760",
        "picture_url" : "https://a0.muscache.com/im/pictures/509974/5a9cf3d5_original
        "host_id" : 65090,
        "neighbourhood" : "Port Phillip",
        "street" : "St Kilda East, VIC, Australia",
        "zipcode" : 3183,
        "latitude" : -37.86453,
        "longitude" : 144.99224,
        "room_type" : "Private room",
        "amenities" : "{Internet,Wifi,Heating,Washer,\"Smoke detector\",\"Carbon monc
blankets\",\"Luggage dropoff allowed\"}",
```

10. What is the price and room type for the listings in Clayton and the name contains “Beautiful”?

Embedding Model:

```
db.listing_host.aggregate([
{$match:{"street":{$regex:/Clayton/},"name":{$regex:/Beautiful/i}}},
{$project:{_id:0,"price":1,"room_type":1}}
])
```



Referencing Model:

```
db.Listing.aggregate([
{$match:{"street":{$regex:/Clayton/},"name":{$regex:/Beautiful/i}}},
{$project:{_id:0,"price":1,"room_type":1}}
])
```

11. For each listing, display the listing’s name, address and neighbourhood in the following format: “Monash Beautiful House, Clayton, VIC, 3800, Clayton” and sort the list in alphabetical order.

Embedding Model:

```
db.listing_host.aggregate([
{$project:{_id:1,address:{$concat:["$name",", ","$street",", ",{$toString:"$zipcode"},",","$neighbourhood"]}}},
{$project:{_id:0}},
{$sort:{address:1}}
]).pretty()
```

```
> db.listing_host.aggregate([
... {$project:{_id:1,address:{$concat:["$name",", ","$street",", ",{$toString:"$zipcode"}," ,","$neighbourhood"]}}},
... {$project:{_id:0}},
... {$sort:{address:1}}
... ]).pretty()
{
        "address" : "(1) Stylish, East Melb 1bed apt, East Melbourne, VIC, Australia, 3002 ,Melbourne"
}
{
        "address" : "**just renovated** Bright & Spacious StKildaEast**, St Kilda East, VIC, Australia, 3183 ,Port Phill
ip"
}
{
        "address" : "2 bedrooms-ideal for friends/family, Prahran, VIC, Australia, 3181 ,Stonnington"
}
{
        "address" : "3 Bedroom Apartment MT111, Southbank, VIC, Australia, 3006 ,Melbourne"
}
{
        "address" : "3 Level Loft with Views close to City, Richmond, VIC, Australia, 3121 ,Yarra"
}
{
        "address" : "5 mins Melbourne CBD in Richmond., Richmond, VIC, Australia, 3121 ,Yarra"
```

Referencing Model:

db.Listing.aggregate([

{$project:{_id:1,address:{$concat:["$name",",","$street",", ",{$toString:"$zipcode"},"
,","$neighbourhood"]}}},

{$project:{_id:0}},

{$sort:{address:1}}

]).pretty()

```
> db.Listing.aggregate([
... {$project:{_id:1,address:{$concat:["$name",",","$street",", ",{$toString:"$zipcode"}," ,","$neighbourhood"]}}},
... {$project:{_id:0}},
... {$sort:{address:1}}
... ]).pretty()
{
        "address" : "(1) Stylish, East Melb 1bed apt,East Melbourne, VIC, Australia, 3002 ,Melbourne"
}
{
        "address" : "**just renovated** Bright & Spacious StKildaEast**,St Kilda East, VIC, Australia, 3183 ,Port Phi
p"
}
{
        "address" : "2 bedrooms-ideal for friends/family,Prahran, VIC, Australia, 3181 ,Stonnington"
}
{
        "address" : "3 Bedroom Apartment MT111,Southbank, VIC, Australia, 3006 ,Melbourne"
}
{
        "address" : "3 Level Loft with Views close to City,Richmond, VIC, Australia, 3121 ,Yarra"
}
{
        "address" : "5 mins Melbourne CBD in Richmond.,Richmond, VIC, Australia, 3121 ,Yarra"
```

12. Which listing is available for the longest number of days? Add the attribute for report generation time and store the current time for when your output is generated.

Embedding Model:

```
db.listing_host.aggregate([
{$group:{_id:"$name",maximum_period:{$max:"$availability_365"}}},
{$sort:{"maximum_period":-1}},
{$limit:13},
{$project:{"Executed_at":new Date(),_id:1,maximum_period:1}}
]).pretty()
```

```
> db.listing_host.aggregate([
... {$group:{_id:"$name",maximum_period:{$max:"$availability_365"}}},
... {$sort:{"maximum_period":-1}},
... {$limit:13},
... {$project:{"Executed_at":new Date(),_id:1,maximum_period:1}}
... ]).pretty()
{
        "_id" : "Cool spot near chapel st!",
        "maximum_period" : 365,
        "Executed_at" : ISODate("2019-09-18T06:49:54.426Z")
}
{
        "_id" : "Lux Apartment, Australian Open",
        "maximum_period" : 365,
        "Executed_at" : ISODate("2019-09-18T06:49:54.426Z")
}
{
        "_id" : "A Room Near the Park",
        "maximum_period" : 365,
        "Executed_at" : ISODate("2019-09-18T06:49:54.426Z")
}
{
        "_id" : "LARGE RM ENSUITE NEAR CTY & AIRPORT",
        "maximum_period" : 365,
        "Executed_at" : ISODate("2019-09-18T06:49:54.426Z")
}
{
```

Referencing Model:

```
db.Listing.aggregate([
{$group:{_id:"$name",maximum_period:{$max:"$availability_365"}}},
{$sort:{"maximum_period":-1}},
{$limit:13},
{$project:{" Executed_at":new Date(),_id:1,maximum_period:1}}
]).pretty()
```

```
> db.Listing.aggregate([
... {$group:{_id:"$name",maximum_period:{$max:"$availability_365"}}},
... {$sort:{"maximum_period":-1}},
... {$limit:13},
... {$project:{" Executed_at":new Date(),_id:1,maximum_period:1}}
... ]).pretty()
{
        "_id" : "Cool spot near chapel st!",
        "maximum_period" : 365,
        " Executed_at" : ISODate("2019-09-18T06:50:46.038Z")
}
{
        "_id" : "Lux Apartment, Australian Open",
        "maximum_period" : 365,
        " Executed_at" : ISODate("2019-09-18T06:50:46.038Z")
}
{
        "_id" : "A Room Near the Park",
        "maximum_period" : 365,
        " Executed_at" : ISODate("2019-09-18T06:50:46.038Z")
}
{
        "_id" : "LARGE RM ENSUITE NEAR CTY & AIRPORT",
        "maximum_period" : 365,
        " Executed_at" : ISODate("2019-09-18T06:50:46.038Z")
}
{
```

13. Using a single query, list the unique neighbourhoods with the price per night greater than $50. Your list should display only the neighbourhoods and prices, and be sorted in reverse alphabetical order of neighbourhood names.

Embedding Model:

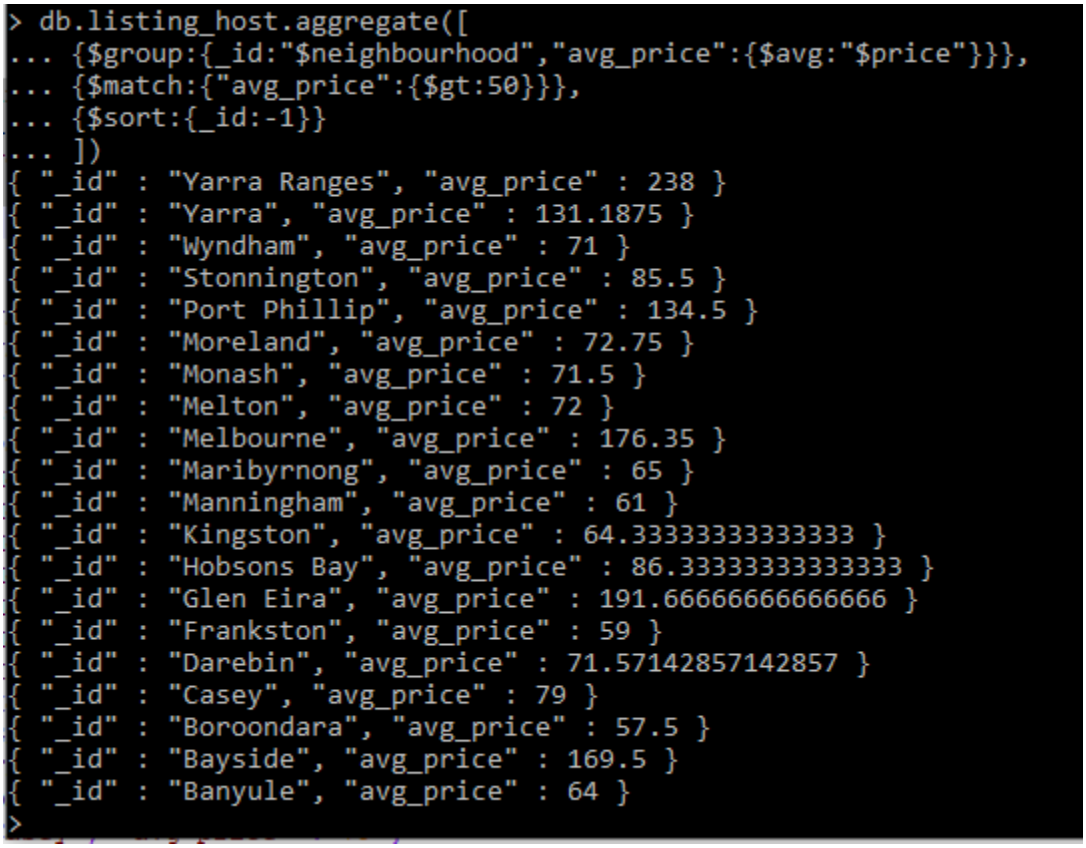
db.listing_host.aggregate([

{$group:{_id:"$neighbourhood","avg_price":{$avg:"$price"}}},

{$match:{"avg_price":{$gt:50}}},

{$sort:{_id:-1}}

])



Referencing Model:

db.Listing.aggregate([

{$group:{_id:"$neighbourhood","avg_price":{$avg:"$price"}}},

{$match:{"avg_price":{$gt:50}}},

{$sort:{_id:-1}}

])

```
> db.Listing.aggregate([
... {$group:{_id:"$neighbourhood","avg_price":{$avg:"$price"}}},
... {$match:{"avg_price":{$gt:50}}},
... {$sort:{_id:-1}}
... ])
{ "_id" : "Yarra Ranges", "avg_price" : 238 }
{ "_id" : "Yarra", "avg_price" : 131.1875 }
{ "_id" : "Wyndham", "avg_price" : 71 }
{ "_id" : "Stonnington", "avg_price" : 85.5 }
{ "_id" : "Port Phillip", "avg_price" : 134.5 }
{ "_id" : "Moreland", "avg_price" : 72.75 }
{ "_id" : "Monash", "avg_price" : 71.5 }
{ "_id" : "Melton", "avg_price" : 72 }
{ "_id" : "Melbourne", "avg_price" : 176.35 }
{ "_id" : "Maribyrnong", "avg_price" : 65 }
{ "_id" : "Manningham", "avg_price" : 61 }
{ "_id" : "Kingston", "avg_price" : 64.33333333333333 }
{ "_id" : "Hobsons Bay", "avg_price" : 86.33333333333333 }
{ "_id" : "Glen Eira", "avg_price" : 191.66666666666666 }
{ "_id" : "Frankston", "avg_price" : 59 }
{ "_id" : "Darebin", "avg_price" : 71.57142857142857 }
{ "_id" : "Casey", "avg_price" : 79 }
{ "_id" : "Boroondara", "avg_price" : 57.5 }
{ "_id" : "Bayside", "avg_price" : 169.5 }
{ "_id" : "Banyule", "avg_price" : 64 }
>
```

14. For each host, find the total number of verification methods and store the results as “number of verification methods”. The output should be sorted according to descending order for number of verification method and it should show only host id, host name and their number of verification methods.

Embedding Model:

```
db.listing_host.aggregate([{$project:{hosts:1}},

{$unwind:"$hosts"},

{$unwind:"$hosts.host_verifications"},

{$group:{_id:{"hid":"$hosts.host_id","hname":"$hosts.host_name"},number_of_verification_methods:{$sum:1}}},

{$project:{_id:1,"number_of_verification_methods":1}},

{$sort:{"number_of_verification_methods":-1}}

]).pretty()
```

```
> db.listing_host.aggregate([{$project:{hosts:1}},
... {$unwind:"$hosts"},
... {$unwind:"$hosts.host_verifications"},
... {$group:{_id:{"hid":"$hosts.host_id","hname":"$hosts.host_name"},number_of_verification_methods:{$sum:1}}},
... {$project:{_id:1,"number_of_verification_methods":1}},
... {$sort:{"number_of_verification_methods":-1}}
... ]).pretty()
{
        "_id" : {
                "hid" : 50121,
                "hname" : "The A2C Team"
        },
        "number_of_verification_methods" : 49
}
{
        "_id" : {
                "hid" : 569413,
                "hname" : "Dina"
        },
        "number_of_verification_methods" : 21
}
{
        "_id" : {
                "hid" : 182833,
                "hname" : "Diana"
        },
        "number_of_verification_methods" : 16
```

Referencing Model:

db.Host.aggregate([

{$unwind:"$host_verifications"},

{$group:{_id:{"hid":"$host_id","hname":"$host_name"},number_of_verification_methods:{$sum:1}}},

{$project:{_id:1,"number_of_verification_methods":1}},

{$sort:{"number_of_verification_methods":-1}}

]).pretty()

```
> db.Host.aggregate([
... {$unwind:"$host_verifications"},
... {$group:{_id:{"hid":"$host_id","hname":"$host_name"},number_of_verification_methods:{$sum:1}}},
... {$project:{_id:1,"number_of_verification_methods":1}},
... {$sort:{"number_of_verification_methods":-1}}
... ]).pretty()
{
        "_id" : {
                "hid" : 189682,
                "hname" : "Belinda"
        },
        "number_of_verification_methods" : 10
}
{
        "_id" : {
                "hid" : 1349266,
                "hname" : "Adam"
        },
        "number_of_verification_methods" : 9
}
{
        "_id" : {
                "hid" : 700065,
                "hname" : "Marilyn"
        },
        "number_of_verification_methods" : 9
```

15. What is the most recent review about listing number 10803?

Embedding Model:

db.listing_host.aggregate([

{$match:{id:10803}},

{$sort:{last_review:-1}},

{$limit:1}

]).pretty()

```
> db.listing_host.aggregate([
... {$match:{id:10803}},
... {$sort:{last_review:-1}},
... {$limit:1}
... ]).pretty()
{
        "_id" : ObjectId("5d7ef395a21b89bacd82456d"),
        "id" : 10803,
        "name" : "Room in Cool Deco Apartment in Brunswick East",
        "summary" : "A large air conditioned room with queen spring mattress bed in a
y. This area is known for its arts culture, live music, cafes and international food.
        "listing_url" : "https://www.airbnb.com/rooms/10803",
        "picture_url" : "https://a0.muscache.com/im/pictures/31323790/90b6ac18_origin
        "host_id" : 38901,
        "neighbourhood" : "Moreland",
        "street" : "Brunswick East, VIC, Australia",
        "zipcode" : 3057,
        "latitude" : -37.76651,
        "longitude" : 144.98074,
        "room_type" : "Private room",
        "amenities" : "{TV,Internet,Wifi,\"Air conditioning\",Kitchen,Heating,\"Famil
Essentials,Shampoo,\"Lock on bedroom door\",\"24-hour check-in\",\"Hair dryer\",Iron,
ate entrance\",\"Hot water\",Microwave,\"Coffee maker\",Refrigerator,\"Dishes and sil
lowed\",\"Long term stays allowed\",Other}",
        "price" : 35,
        "extra_people" : 15,
        "minimum_nights" : 3,
        "number_of_reviews" : 126,
        "last_review" : ISODate("2019-06-19T00:00:00Z"),
        "reviews_per_month" : 1.59,
        "calculated_host_listings_count" : 1,
        "availability_365" : 194,
```

Referencing Model:

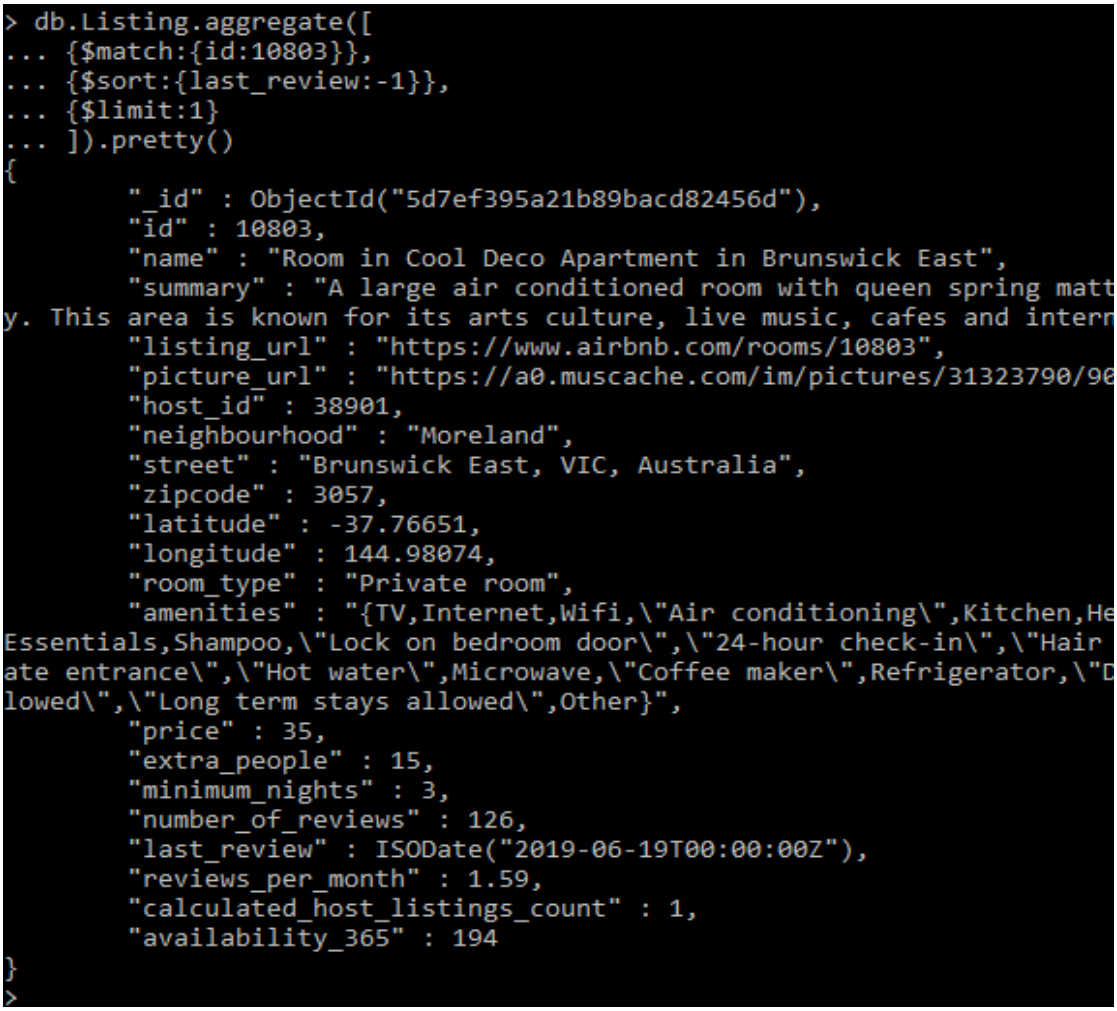
db.Listing.aggregate([

{$match:{id:10803}},

{$sort:{last_review:-1}},

{$limit:1}

]).pretty()



Cassandra:

- create keyspace FIT5137_Assign_C3 with replication={'class':'SimpleStrategy','replication_factor':1};
- use FIT5137_Assign_C3;
- CREATE TABLE review1 (listing_id int,

  id int,

  rdate date,

  reviewer_id int,

  reviewer_name text,

  review_scores_rating int,

  comments text,

PRIMARY KEY ((review_scores_rating),listing_id,rdate,reviewer_id)

) ;

```
copy FIT5137_Assign_C3.review1
(listing_id,id,rdate,reviewer_id,reviewer_name,review_scores_rating,comments) from
'C:\Users\User\Desktop\FIT5137\A\review4.txt' WITH DELIMITER='|' AND
HEADER=TRUE;
```

- CREATE TABLE review2 (listing_id int,

  id int,

  rdate date,

  reviewer_id int,

  reviewer_name text,

  review_scores_rating int,

  comments text,

  PRIMARY KEY ((rdate),listing_id,reviewer_id,review_scores_rating)

) ;

```
copy FIT5137_Assign_C3.review2
(listing_id,id,rdate,reviewer_id,reviewer_name,review_scores_rating,comments) from
'C:\Users\User\Desktop\FIT5137\A\review4.txt' WITH DELIMITER='|' AND
HEADER=TRUE;
```

- CREATE TABLE review3 (listing_id int,

  id int,

  rdate date,

  reviewer_id int,

  reviewer_name text,

  review_scores_rating int,

  comments text,

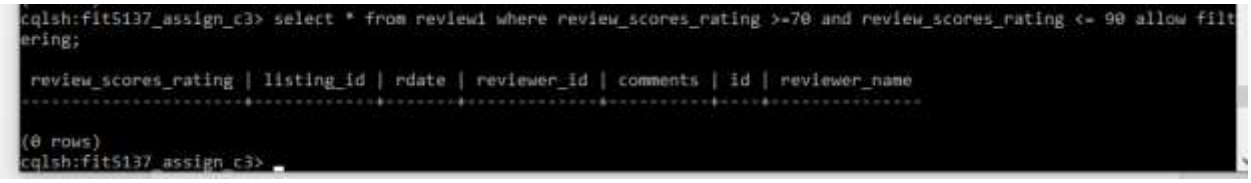
  PRIMARY KEY ((listing_id,reviewer_name),review_scores_rating)

) ;

copy FIT5137_Assign_C3.review3 (listing_id,id,rdate,reviewer_id,reviewer_name,review_scores_rating,comments) from 'C:\Users\User\Desktop\FIT5137\A\review4.txt' WITH DELIMITER='|' AND HEADER=TRUE;

16. Display all reviews which rating is between 70 and 90.

select * from review1 where review_scores_rating >=70 and review_scores_rating <= 90 allow filtering;



note: this query need to search by **review_scores_rating** field**.**Therefore, we have created a column family called **review1** and used **review_scores_rating** as partition key and listing_id,rdate,reviewer_id as clustering keys to support searching.

17. Display the listing, reviewer, and comment that the reviewer rating is below 50.

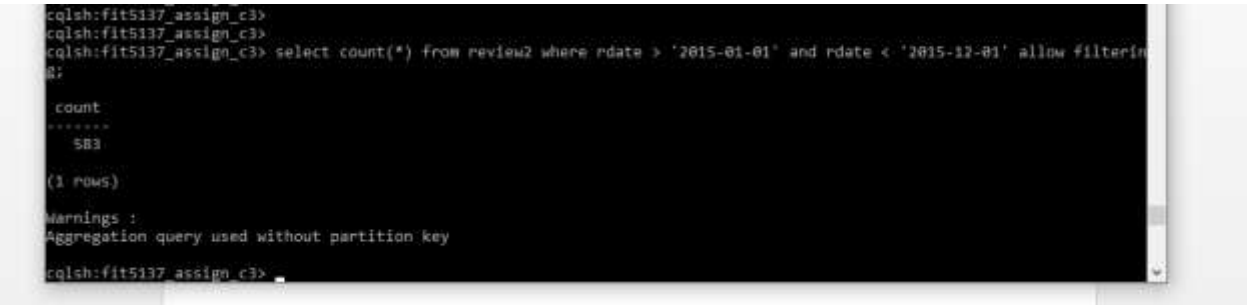
select listing_id,reviewer_name,comments from review1 where review_scores_rating < 50 allow filtering;

note: this query need to search by **review_scores_rating** field**.**Therefore, we have created a column family called **review1** and used **review_scores_rating** as partition key and listing_id,rdate,reviewer_id as clustering keys to support searching.

18. How many reviews left in 2015?

select count(*) from review2 where rdate > '2015-01-01' and rdate < '2015-12-01' allow filtering;

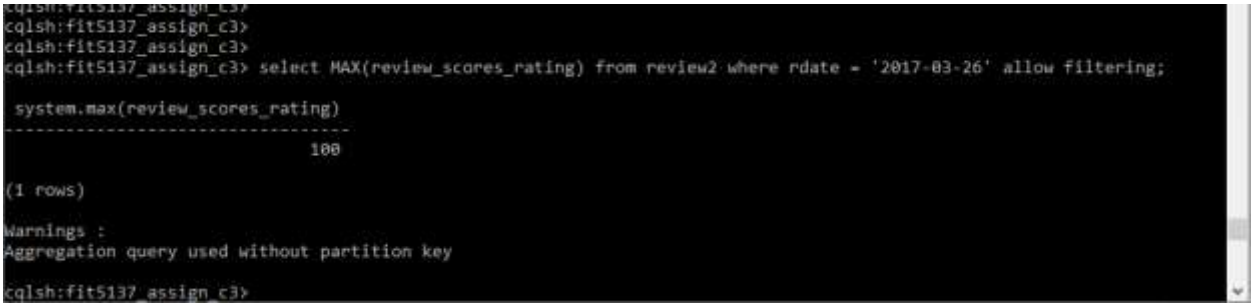
note: this query need to search by **rdate** field**.**Therefore, we have created a column family called **review2** and used **rdate** as partition key and listing_id,reviewer_id,review_scores_rating as clustering keys to support searching.



19. Display the review with the highest rating on 26th March 2017.

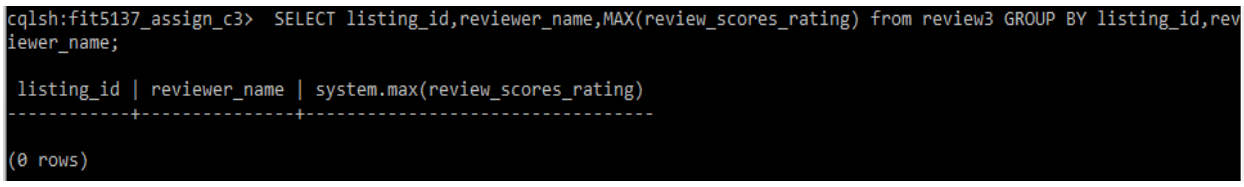
select MAX(review_scores_rating) from review2 where rdate = '2017-03-26' allow filtering;

note: this query need to search by **rdate** field**.**Therefore, we have created a column family called **review2** and used **rdate** as partition key and listing_id,reviewer_id,review_scores_rating as clustering keys to support searching.



20. Display the listing ID, reviewer name, and its highest rating.

SELECT listing_id,reviewer_name,MAX(review_scores_rating) from review3 GROUP BY listing_id,reviewer_name;



note: this query need to search by **listing_id,reviewer_name** field**.**Therefore, we have created a column family called **review3** and used **listing_id,reviewer_name** as partition key and review_scores_rating as clustering key to support searching.

Indices:

1. db.listing_host.createIndex({street:1, price:1})

2. db.listing_host.createIndex({neighbourhood:1})

We selected the neighbourhood because we can filter our search based on neighbourhood.

And for compound index we selected street and price so that we can find listings with correspondence to street address and price.

Five Additional Queries:

1. **What listings provide extra people 20 to 30 and have email verification method?**

```
db.listing_host.aggregate([
{$unwind:"$hosts"},
{$match:{"extra_people" :{$gt:20,$lt :30},"hosts.host_verifications":"email"}}
]).pretty()
```

2. **Which suburbs have more than one listing and sort them by list count?**

```
db.listing_host.aggregate([
{$group:{_id:"$street",list_count:{$sum:1}}},
{$match:{"list_count":{$gt:1}}},
{$sort:{list_count:-1}}
]).pretty()
```

3. **Display listings that have host who responds within an hour?**

```
db.listing_host.aggregate([
```

{$unwind:"$hosts"},

{$match:{"hosts.host_response_time":"within an hour"}},

{$project:{name:1,_id:0}}

]).pretty()

4. Select reviewer id and review score for listing id 279854 which were done before 1 June, 2012.

select reviewer_id,review_scores_rating from review2 where rdate < '2012-06-01' and listing_id = 279854 allow filtering;

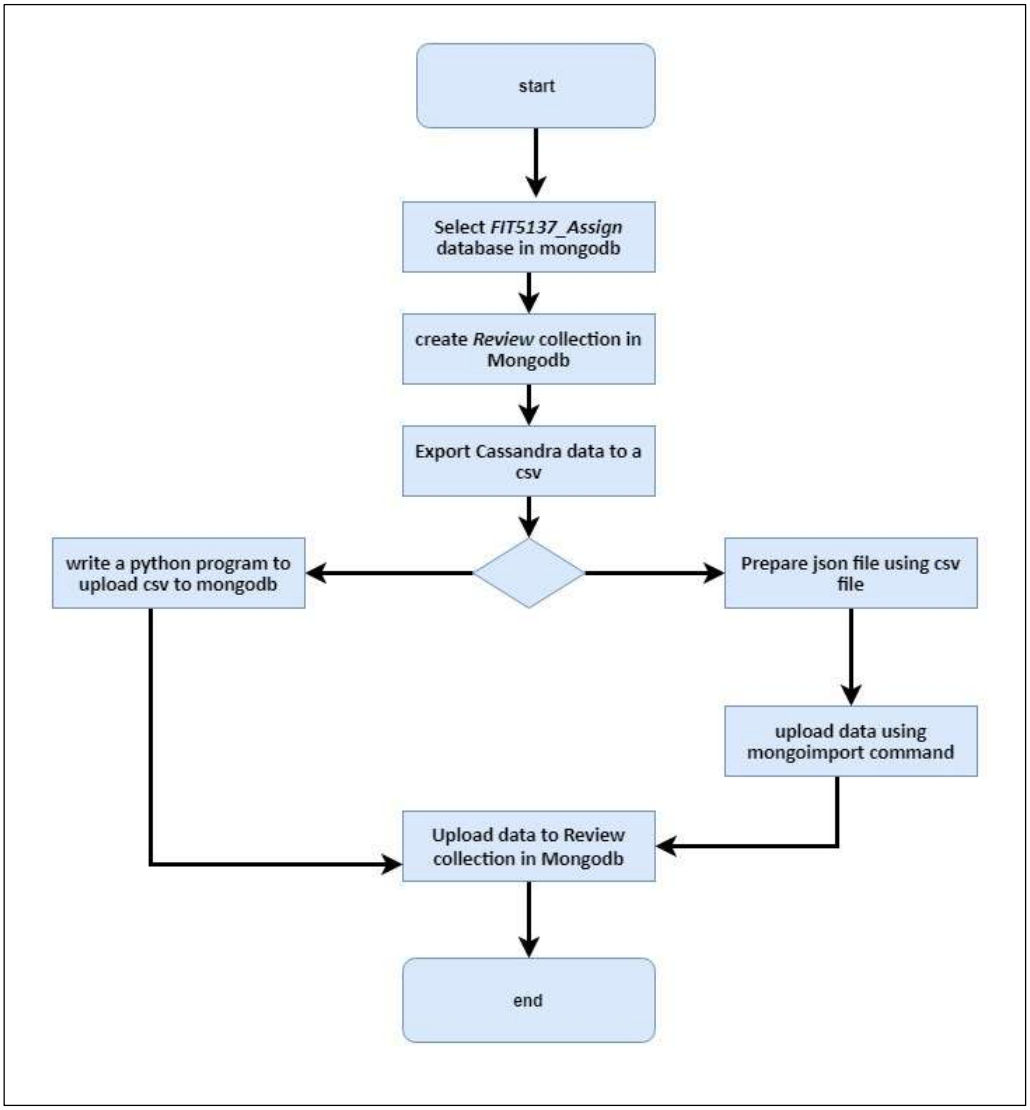
5. Display reviewer id whose review scores are greater than 60 and the date when the scores were given.

select reviewer_id,rdate from review1 where review_scores_rating > 60 allow filtering;

C.4. Database Comparison.

After analyzing data in two databases, I suggest you to use Mongodb to handle all the information related to MonashBNB.

Steps to migrate data from Cassandra to mongo db.



The initial assumption is that there are two collections for **Listing** and **Host** in mongo db database called **FIT5137_Assign** that we created for **c1 part** of this analysis. Inside **Listing** table **host_id is** used as a reference to the Host collection**.** Therefore, it is assumed that the current collections in mongo db are following referencing model.

Step1

In order to migrate data in Cassandra, first user has to select above database (FIT5137_Assign) with following command. Use FIT5137_Assign.

Step2

create a new collection called review in mongodb to store data migrated from Cassandra.

db.createCollection("Review")

step3

export Cassandra data into csv file using export command

copy review
(listing_id,id,date,reviewer_id,reviewer_name,review_scores_rating,comments)

to review.csv';

step4

this step is used to transfer data retrieved from Cassandra as review.csv file. This can be done in two ways. By using Microsoft excel or other software like Notepad++ , prepare a Jason file with correct key value pairs using data in **review**.csv file. Then using mongoimport command, data in json file can be inserted to the monmgo db collection.

Alternatively, step4 can be performed using a python program which can import csv or pandas library to read csv file and then the data can be inserted to the newly created mongodb collection.

Step5

After completing step4, data migration from Cassandra to mongodb in is completed and the company can only use mongodb for its operations.

Mongo db and Cassandra strengths and weaknesses comparison

Cassandra	Mongodb
Handling very large amount of data by scaling them into multiple nodes easily.	Can have 1000 clusters, but adding a new cluster note has more work than Cassandra.
There is no master nodes in the cluster and it consists of peer to peer architecture between similar nodes.	There is only one master node in a cluster.

Data is stored as column families which have predefined columns	Data is stored as documents using BSON format in collections and they do not have a predefined structure.
Cassandra has quick writing and reading	Has lower writing and reading speed compared to Cassandra.
Secondary indexes are limited to single column and equality comparisons.	Easy to index any property and use them in queries.

Reason to select Mongodb

Firstly, By analyzing MonashBNB data it is evident that Host data, Listing data, review data is now growing so fast. Therefore, a software like Casandra which is a very powerful software to handle large amount of data is not beneficial to handle MonashBNB data.

Secondly, MonashBnB provides accommodation only around Melbourne. Therefore, the system doesn't need much scalability to split over multiple sites and it can use mongo db to process the data.

Querying in Cassandra in for non-key column is harder unless creating an index. But if these data stores in mongo db ,it is easy to query them using any attribute.

Because of above reasons, I would like to suggest to use mongo db as the databse in MonashBnB.