MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
29971462	*Lakmee*	*Amila*
29462924	*Alshaikhmubarak*	*Sarah*

* Please include the names of all other group members.

Unit name and code	FIT5137 Advanced database technology	
Title of assignment	Assignment 1 - MongoDB and Cassandra	
Lecturer/tutor	Agnes Haryanto/Alina Bhari	
Tutorial day and time	Tuesday 10:00 am	**Campus: Caulfield**
Is this an authorised group assignment?	**Yes** **No**	
Has any part of this assignment been previously submitted as part of another unit/course?	**Yes** **No**	
Due Date 18-09-2019		**Date submitted 18-09-2019**

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) ..... Signature of lecturer/tutor .....

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ..... *Date*.....

* delete (iii) if not applicable

Signature _____Sarah_____Date:_____18-09-201_____ Signature _____Date:_____

Signature _____Lakmee_____Date:_____ 18-09-201_____ Signature _____Date:_____

Signature _____Date:_____ Signature _____Date:_____

Privacy Statement

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Updated: 17 Jun 2014

Contribution Declaration Form
(to be completed by all team members)

Please fill in the form with the contribution from each student towards the assignment.

1 NAME AND CONTRIBUTION DETAILS

Student ID	Student Name	Contribution Percentage
29971462	Amila Lakmee	50%
29462924	Sarah Alshaikhmubarak	50%

List of parts that each student did:

I. Amila:

□ Task C.1 (MongoDB Host table, Cassandra review table)
□ Task C.2 (odd numbers)
□ Task C.3 (odd numbers, two additional queries)
□ Task C.4 (together)

II. Sarah:

□ Task C.1 (MongoDB listing table)
□ Task C.2 (even numbers)
□ Task C.3 (even numbers, 3 additional queries)
□ Task C.4 (together)

2 DECLARATION

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.

3 SIGNATURE

Signatures

Sarah	Amila

Day Month Year

Date 18 / 09 / 2019

FIT5137: Advanced Database Technology

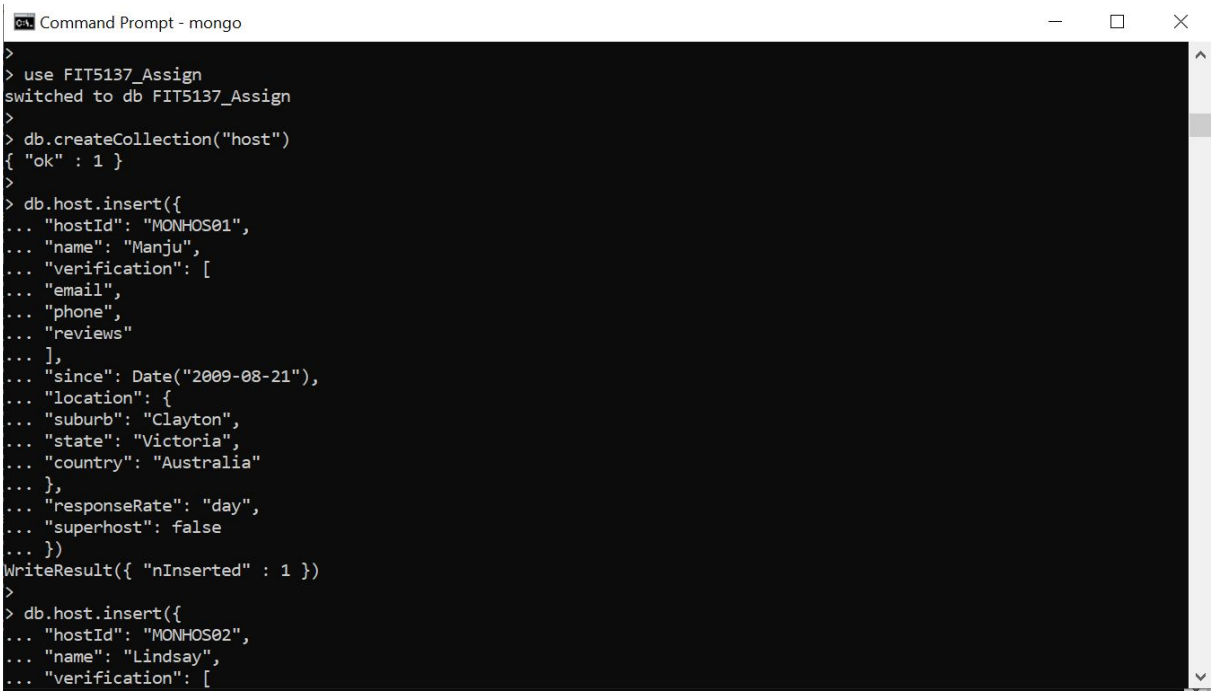
Assignment 1
MongoDB and Cassandra

Group members:

Amila Lakmee - 29971462
Sarah Alshaikhmubarak - 29462924

C.1. Database Design

Accommodation Host Report Table:




```
use FIT5137_Assign

db.createCollection("host")

db.host.insert({
        "hostId": "MONHOS01",
        "name": "Manju",
        "verification": [
                "email",
                "phone",
                "reviews"
        ],
        "since": Date("2009-08-21"),
        "location": {
                "suburb": "Clayton",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within a day",
        "superhost": false
})
```

Reviews Report:

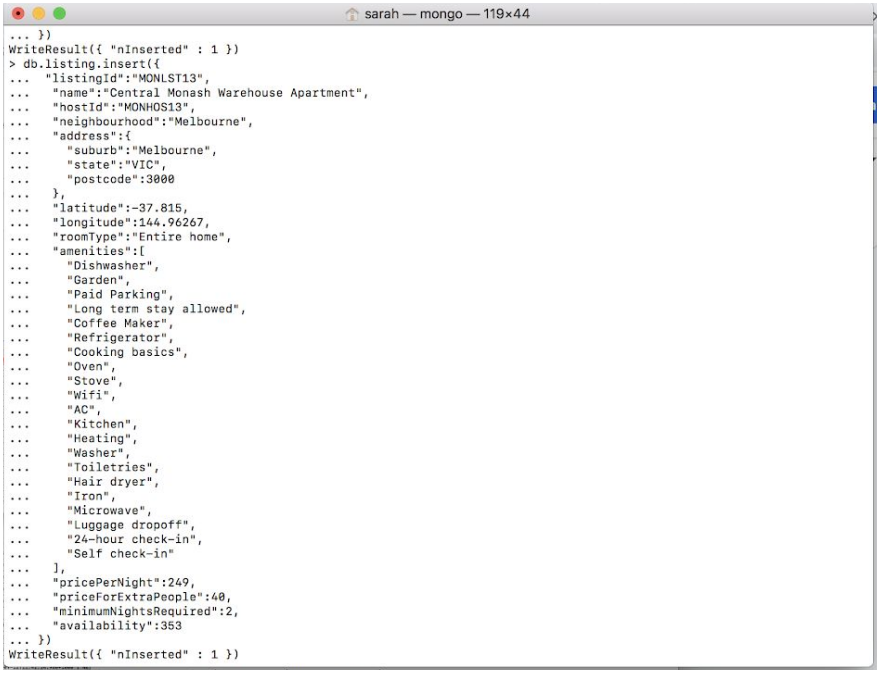
review_id	comment	listing_id	rating	review_timestamp	reviewer_id	reviewer_name	satisfaction_reason
REV03	Nice View and Location	MONLST02	100	2017-03-21 21:37:50.000000+0000	5000031	Camille	{'location', 'view'}
REV19	Good Location	MONLST12	87	2017-03-26 20:37:50.000000+0000	500013	Anouck	{'clean', 'space'}
REV10	Friendly Host	MONLST05	98	2017-03-22 23:37:50.000000+0000	500010	Elisabeth	{'host', 'price'}
REV06	Very Clean House	MONLST03	90	2017-03-22 04:37:50.000000+0000	500006	Greg	{'host', 'view'}
REV13	Very Comfortable	MONLST10	93	2017-03-25 21:02:10.000000+0000	500013	Anouck	{'host', 'view'}
REV15	Beautiful View	MONLST13	98	2017-03-25 21:48:40.000000+0000	500015	Jehan	{'amenities', 'location'}
REV08	Nice Building	MONLST05	96	2017-03-22 07:37:50.000000+0000	500008	Klaus	{'location', 'view'}
REV02	Good Host	MONLST02	90	2017-03-21 22:37:50.000000+0000	500002	Johannes	{'host'}
REV11	Very Clean and comfortable	MONLST09	100	2017-03-23 06:37:50.000000+0000	500011	Derek	{'clean', 'space'}
REV20	Nice Building	MONLST05	96	2017-03-26 21:45:10.000000+0000	500011	Derek	{'host', 'view'}
REV14	Friendly Host	MONLST12	85	2017-03-25 21:49:40.000000+0000	500014	Jerome	{'clean', 'location'}
REV09	Friendly Host	MONLST06	100	2017-03-22 22:37:50.000000+0000	500009	Rox	{'host', 'price'}
REV12	Friendly and Nice Host	MONLST09	92	2017-03-24 21:07:40.000000+0000	500021	Joy	{'clean', 'host'}
REV07	Nice Location	MONLST04	91	2017-03-22 06:37:50.000000+0000	500007	Wolfgang	{'location', 'price'}
REV05	Good Location	MONLST01	93	2017-03-22 02:37:50.000000+0000	500005	Adele	{'location', 'price'}
REV01	Beautiful View	MONLST02	90	2017-03-21 21:37:50.000000+0000	500001	Miriam	{'amenities', 'location'}
REV16	Good Location	MONLST14	97	2017-03-25 21:48:10.000000+0000	500012	Joy	{'amenities', 'view'}
REV18	Bad Service	MONLST10	20	2017-03-25 21:37:10.000000+0000	500002	Johannes	{'amenities', 'view'}
REV04	Excellent Price	MONLST02	95	2017-03-21 23:37:50.000000+0000	500004	Paige	{'price'}
REV17	Bad Location	MONLST14	30	2017-03-25 21:47:40.000000+0000	500001	Jerome	{'price', 'view'}

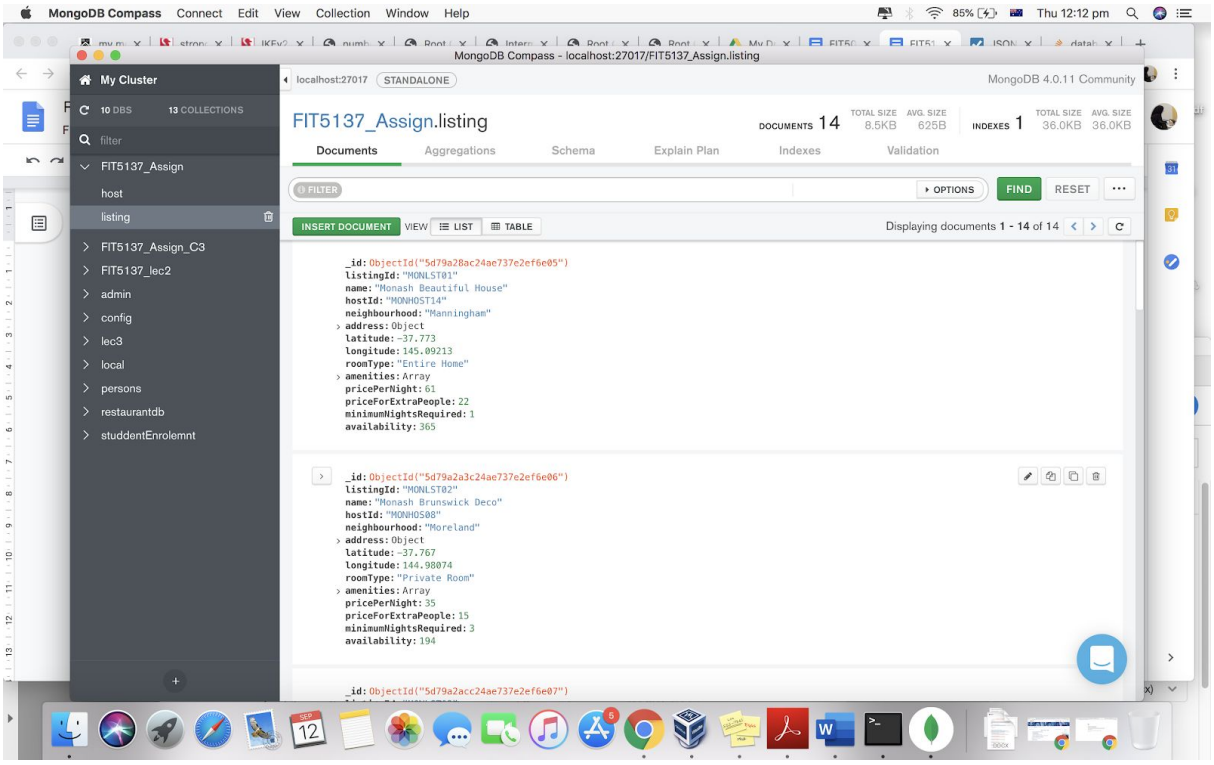
(20 rows)

```
CREATE KEYSPACE fit5137_assign
WITH replication = {'class': 'SimpleStrategy',
'replication_factor' : 1};

use fit5137_assign;

CREATE TABLE review (
        listing_id text,
        review_id text,
        review_timestamp timestamp,
        reviewer_id int,
        reviewer_name text,
        rating int,
        satisfaction_reason set<text>,
        comment text,
        PRIMARY KEY (review_id)
);

INSERT INTO review (
        listing_id,
        review_id,
        review_timestamp,
        reviewer_id,
        reviewer_name,
        rating,
        satisfaction_reason,
        comment)
```

Listing Table:

```
use FIT5137_Assign

db.createCollection("listing")

db.listing.insert({
"listingId ":"MONLST01",
 "name":"Monash Beautiful House",
 "hostId":"MONHOST14",
 "neighbourhood":"Manningham",
 "address":{
  "suburb":"Clayton",
  "state":"VIC",
  "postcode":3800
 },
 "latitude":-37.773,
 "longitude":145.09213,
 "roomType":"Entire Home",
 "amenities":[
  "TV",
  "Wifi",
  "Pets Allowed",
  "Family friendly",
  "24-hour check-in",
  "Self check-in"
 ],
 "pricePerNight":61,
 "priceForExtraPeople":22,
 "minimumNightsRequired":1,
 "availability":365})
```


sarah — mongo — 119×44
... })
WriteResult({ "nInserted" : 1 })
> db.listing.insert({
... "listingId":"MONLST13",
... "name":"Central Monash Warehouse Apartment",
... "hostId":"MONHOS13",
... "neighbourhood":"Melbourne",
... "address":{
... "suburb":"Melbourne",
... "state":"VIC",
... "postcode":3000
... },
... "latitude":-37.815,
... "longitude":144.96267,
... "roomType":"Entire home",
... "amenities":[
... "Dishwasher",
... "Garden",
... "Paid Parking",
... "Long term stay allowed",
... "Coffee Maker",
... "Refrigerator",
... "Cooking basics",
... "Oven",
... "Stove",
... "Wifi",
... "AC",
... "Kitchen",
... "Heating",
... "Washer",
... "Toiletries",
... "Hair dryer",
... "Iron",
... "Microwave",
... "Luggage dropoff",
... "24-hour check-in",
... "Self check-in"
...],
... "pricePerNight":249,
... "priceForExtraPeople":40,
... "minimumNightsRequired":2,
... "availability":353
... })
WriteResult({ "nInserted" : 1 })


MongoDB Compass Connect Edit View Collection Window Help
MongoDB Compass - localhost:27017/FIT5137_Assign.listing
My Cluster
localhost:27017 STANDALONE
MongoDB 4.0.11 Community
FIT5137_Assign.listing
Documents Aggregations Schema Explain Plan Indexes Validation
FIND RESET
INSERT DOCUMENT
Displaying documents 1 - 14 of 14
FIT5137_Assign
host
listing
FIT5137_Assign_C3
FIT5137_lec2
admin
config
lec3
local
persons
restaurantdb
studdentEnrolemnt

C.2. Database Modifications.

1. Updating the verification Adam and adding Facebook to his list of existing verifications.

```
db.host.update(
        { "name": "Adam" },
        { $push: { "verification": "facebook" } }
)
```

```
> db.host.update(
... { "name": "Adam" },
... { $push: { "verification": "facebook" } }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
_id: ObjectId("5d7f3188d319798e394ff557")
hostId: "MONHOS03"
name: "Adam"
verification: Array
  0: "email"
  1: "phone"
  2: "google"
  3: "reviews"
  4: "jumio"
  5: "government id"
  6: "work email"
  7: "facebook"
since: "Mon Sep 16 2019 16:54:00 GMT+1000 (AUS Eastern Standard Time)"
location: Object
responseRate: "hour"
superhost: false
```

2. Insert 5 new hosts with the following details using a single insert command:

```
db.host.insertMany([{
        "hostId": "MONHOS011",
        "name": "Alison",
        "verification": [
                "email",
                "phone",
                "facebook",
                "reviews"],
        "since": Date("2009-01-09"),
        "location": {
```

```
                "suburb": "Caulfield",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within an hour",
        "superhost": false
},
{
        "hostId": "MONHOS012",
        "name": "Mike",
        "verification": [
                "email",
                "phone"],
        "since": Date("2009-01-09"),
        "location": {
                "suburb": "Clayton",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within a day",
        "superhost": true
},{
        "hostId": "MONHOS013",
        "name": "Robyn",
        "Verification":[
                "facebook",
                "reviews"],
        "since": Date("2009-01-09"),
        "location": {
                "suburb": "Berwick",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within an hour",
        "superhost": false
},{
        "hostId": "MONHOS014",
        "name": "Daniel",
        "verification": [
                "email",
                "manual offline"        ,
                "work email"],
```

```
        "since": Date("2009-01-09"),
        "location": {
                "suburb": "Frankston",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within a day",
        "superhost": true
},{
        "hostId": "MONHOS015",
        "name": "Ron",
        "verification": [
                "facebook"
                ],
        "since": Date("2009-01-09"),
        "location": {
                "suburb": "Caulfield",
                "state": "Victoria",
                "country": "Australia"
        },
        "responseRate": "within a day",
        "superhost": false
}])
```

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5d80d540543a8d1557e9a0a7"),
                ObjectId("5d80d540543a8d1557e9a0a8"),
                ObjectId("5d80d540543a8d1557e9a0a9"),
                ObjectId("5d80d540543a8d1557e9a0aa"),
                ObjectId("5d80d540543a8d1557e9a0ab")
        ]
}
```

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

```
db.host.updateMany(
        { "responseRate": "within an hour" },
        { $set: { superhost: true } }
)
```

```
> db.host.updateMany(
... { "responseRate": "within an hour" },
... { $set: { superhost: true } }
... )
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
>
```

```
_id: ObjectId("5d7f424ed319798e394ff57d")
hostId: "MONHOS013"
name: "Robyn"
> Verification: Array
since: "Mon Sep 16 2019 18:05:34 GMT+1000 (AUS Eastern Standard Time)"
> location: Object
responseRate: "within an hour"
superhost: true
```

4. Delete all listings with zero availability, taking into account any future addition of data where availability maybe zero.

db.listing.deleteMany({ "availability":0})

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

5. Using only the neighbourhood information, change the neighbourhood name of "Monash" to "Monash City".

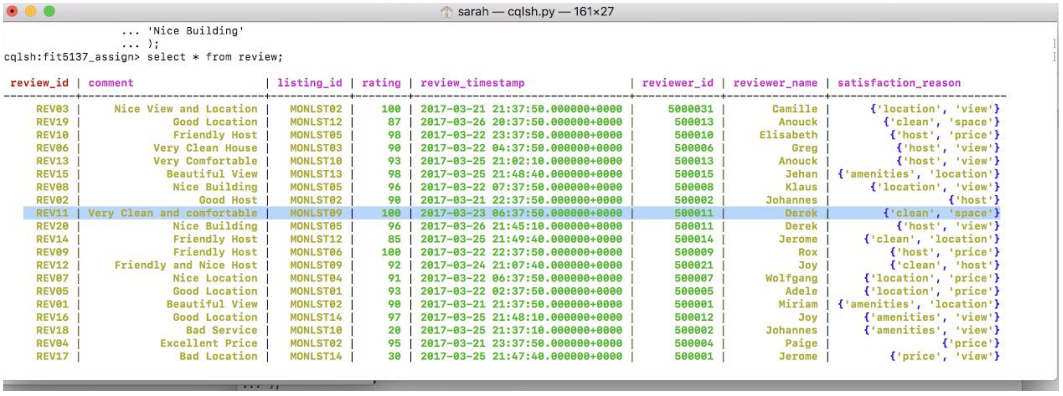
db.listing.updateMany(
{ "neighbourhood": "Monash" },
{ $set: { "neighbourhood": "Monash City" } }
)

```
> db.listing.updateMany(
... { "neighbourhood": "Monash" },
... { $set: { "neighbourhood": "Monash City" } }
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
_id: ObjectId("5d7f330bd319798e394ff575")
listingId: "MONLST09"
name: "Monash JapaneseStyle"
hostId: "MONHOS11"
neighbourhood: "Monash City"
> address: Object
latitude: -37.9
longitude: 145.11447
roomType: "Entire home"
> amenities: Array
pricePerNight: 98
priceForExtraPeople: 0
minimumNightsRequired: 2
availability: 219
```

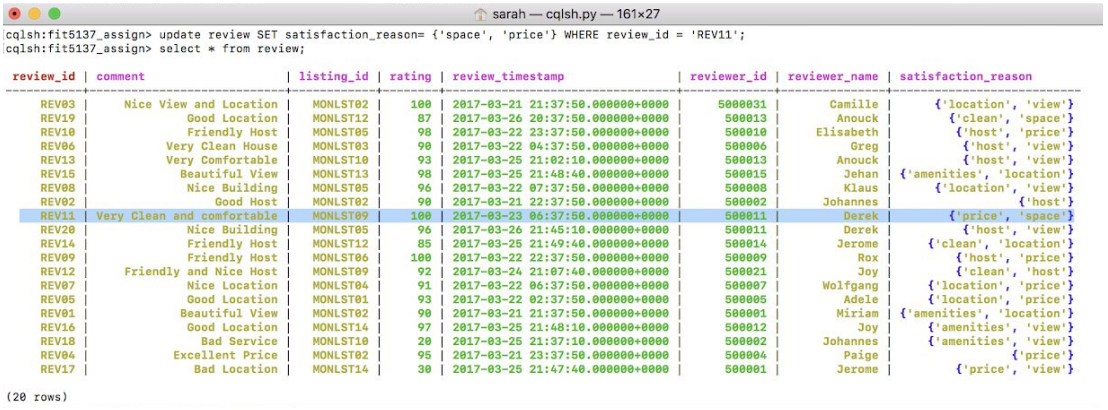
6. Change the satisfied reason to “space” and “price” for review ID “REV11”.

This is the data before the update:



```
sarah — cqlsh.py — 161×27
            ... 'Nice Building'
            ... );
cqlsh:fit5137_assign> select * from review;

 review_id | comment                    | listing_id | rating | review_timestamp                | reviewer_id | reviewer_name | satisfaction_reason
-----------+----------------------------+------------+--------+---------------------------------+-------------+---------------+-----------------------
     REV03 |     Nice View and Location |   MONLST02 |    100 | 2017-03-21 21:37:50.000000+0000 |     5000031 |       Camille |      {'location', 'view'}
     REV19 |              Good Location |   MONLST12 |     87 | 2017-03-26 20:37:50.000000+0000 |      500013 |        Anouck |        {'clean', 'space'}
     REV10 |             Friendly Host  |   MONLST05 |     98 | 2017-03-22 23:37:50.000000+0000 |      500010 |     Elisabeth |         {'host', 'price'}
     REV06 |          Very Clean House  |   MONLST03 |     90 | 2017-03-22 04:37:50.000000+0000 |      500006 |          Greg |          {'host', 'view'}
     REV13 |          Very Comfortable  |   MONLST10 |     93 | 2017-03-25 21:02:10.000000+0000 |      500013 |        Anouck |          {'host', 'view'}
     REV15 |            Beautiful View  |   MONLST13 |     98 | 2017-03-25 21:48:40.000000+0000 |      500015 |         Jehan | {'amenities', 'location'}
     REV08 |             Nice Building  |   MONLST05 |     96 | 2017-03-22 07:37:50.000000+0000 |      500008 |         Klaus |      {'location', 'view'}
     REV02 |                Good Host   |   MONLST02 |     90 | 2017-03-21 22:37:50.000000+0000 |      500002 |      Johannes |                {'host'}
     REV11 | Very Clean and comfortable |   MONLST09 |    100 | 2017-03-23 06:37:50.000000+0000 |      500011 |         Derek |        {'clean', 'space'}
     REV20 |             Nice Building  |   MONLST05 |     96 | 2017-03-26 21:45:10.000000+0000 |      500011 |         Derek |          {'host', 'view'}
     REV14 |             Friendly Host  |   MONLST12 |     85 | 2017-03-25 21:49:40.000000+0000 |      500014 |        Jerome |     {'clean', 'location'}
     REV09 |             Friendly Host  |   MONLST06 |    100 | 2017-03-22 22:37:50.000000+0000 |      500009 |           Rox |         {'host', 'price'}
     REV12 |  Friendly and Nice Host    |   MONLST09 |     92 | 2017-03-24 21:07:40.000000+0000 |      500021 |           Joy |         {'clean', 'host'}
     REV07 |             Nice Location  |   MONLST04 |     91 | 2017-03-22 06:37:50.000000+0000 |      500007 |      Wolfgang |      {'location', 'price'}
     REV05 |             Good Location  |   MONLST01 |     93 | 2017-03-22 02:37:50.000000+0000 |      500005 |         Adele |      {'location', 'price'}
     REV01 |            Beautiful View  |   MONLST02 |     90 | 2017-03-21 21:37:50.000000+0000 |      500001 |        Miriam | {'amenities', 'location'}
     REV16 |             Good Location  |   MONLST14 |     97 | 2017-03-25 21:48:10.000000+0000 |      500012 |           Joy |      {'amenities', 'view'}
     REV18 |               Bad Service  |   MONLST10 |     20 | 2017-03-25 21:37:10.000000+0000 |      500002 |      Johannes |      {'amenities', 'view'}
     REV04 |          Excellent Price   |   MONLST02 |     95 | 2017-03-21 23:37:50.000000+0000 |      500004 |         Paige |                {'price'}
     REV17 |               Bad Location |   MONLST14 |     30 | 2017-03-25 21:47:40.000000+0000 |      500001 |        Jerome |         {'price', 'view'}
```

UPDATE review SET satisfaction_reason= {'space', 'price'} WHERE review_id = 'REV11'.
This is the result after the update:



```
sarah — cqlsh.py — 161×27
cqlsh:fit5137_assign> update review SET satisfaction_reason= {'space', 'price'} WHERE review_id = 'REV11';
cqlsh:fit5137_assign> select * from review;

 review_id | comment                    | listing_id | rating | review_timestamp                | reviewer_id | reviewer_name | satisfaction_reason
-----------+----------------------------+------------+--------+---------------------------------+-------------+---------------+-----------------------
     REV03 |     Nice View and Location |   MONLST02 |    100 | 2017-03-21 21:37:50.000000+0000 |     5000031 |       Camille |      {'location', 'view'}
     REV19 |              Good Location |   MONLST12 |     87 | 2017-03-26 20:37:50.000000+0000 |      500013 |        Anouck |        {'clean', 'space'}
     REV10 |             Friendly Host  |   MONLST05 |     98 | 2017-03-22 23:37:50.000000+0000 |      500010 |     Elisabeth |         {'host', 'price'}
     REV06 |          Very Clean House  |   MONLST03 |     90 | 2017-03-22 04:37:50.000000+0000 |      500006 |          Greg |          {'host', 'view'}
     REV13 |          Very Comfortable  |   MONLST10 |     93 | 2017-03-25 21:02:10.000000+0000 |      500013 |        Anouck |          {'host', 'view'}
     REV15 |            Beautiful View  |   MONLST13 |     98 | 2017-03-25 21:48:40.000000+0000 |      500015 |         Jehan | {'amenities', 'location'}
     REV08 |             Nice Building  |   MONLST05 |     96 | 2017-03-22 07:37:50.000000+0000 |      500008 |         Klaus |      {'location', 'view'}
     REV02 |                Good Host   |   MONLST02 |     90 | 2017-03-21 22:37:50.000000+0000 |      500002 |      Johannes |                {'host'}
     REV11 | Very Clean and comfortable |   MONLST09 |    100 | 2017-03-23 06:37:50.000000+0000 |      500011 |         Derek |         {'price', 'space'}
     REV20 |             Nice Building  |   MONLST05 |     96 | 2017-03-26 21:45:10.000000+0000 |      500011 |         Derek |          {'host', 'view'}
     REV14 |             Friendly Host  |   MONLST12 |     85 | 2017-03-25 21:49:40.000000+0000 |      500014 |        Jerome |     {'clean', 'location'}
     REV09 |             Friendly Host  |   MONLST06 |    100 | 2017-03-22 22:37:50.000000+0000 |      500009 |           Rox |         {'host', 'price'}
     REV12 |  Friendly and Nice Host    |   MONLST09 |     92 | 2017-03-24 21:07:40.000000+0000 |      500021 |           Joy |         {'clean', 'host'}
     REV07 |             Nice Location  |   MONLST04 |     91 | 2017-03-22 06:37:50.000000+0000 |      500007 |      Wolfgang |      {'location', 'price'}
     REV05 |             Good Location  |   MONLST01 |     93 | 2017-03-22 02:37:50.000000+0000 |      500005 |         Adele |      {'location', 'price'}
     REV01 |            Beautiful View  |   MONLST02 |     90 | 2017-03-21 21:37:50.000000+0000 |      500001 |        Miriam | {'amenities', 'location'}
     REV16 |             Good Location  |   MONLST14 |     97 | 2017-03-25 21:48:10.000000+0000 |      500012 |           Joy |      {'amenities', 'view'}
     REV18 |               Bad Service  |   MONLST10 |     20 | 2017-03-25 21:37:10.000000+0000 |      500002 |      Johannes |      {'amenities', 'view'}
     REV04 |          Excellent Price   |   MONLST02 |     95 | 2017-03-21 23:37:50.000000+0000 |      500004 |         Paige |                {'price'}
     REV17 |               Bad Location |   MONLST14 |     30 | 2017-03-25 21:47:40.000000+0000 |      500001 |        Jerome |         {'price', 'view'}

(20 rows)
```

7. Deleting reviews commented by user “500015”

CREATE INDEX ON review (reviewer_id);

SELECT review_id FROM review WHERE reviewer_id=500015;
DELETE FROM review WHERE review_id='REV15';

```
cqlsh:fit5137_assign> Select * FROM review WHERE reviewer_id=500015;

 review_id | comment | listing_id | rating | review_timestamp | reviewer_id | reviewer_name | satisfaction_reason
-----------+---------+------------+--------+------------------+-------------+---------------+---------------------

(0 rows)
```

C.3. Queries.

Importing MongoDB:

mongoimport --db FIT5137_Assign_C3 --collection host --file C:\Users\lakme\Downloads\host.json

mongoimport --db FIT5137_Assign_C3 --collection listing --file C:\Users\lakme\Downloads\listing.json

Creating Embedded Collection

```
db.host.aggregate([
        { $lookup:{ from:"listing", localField:"host_id", foreignField:"host_id", as:"hostList"
} },            {$out: 'host_listing'}
])
```

Importing Cassandra:

CREATE KEYSPACE fit5137_assign_c3
WITH replication = {'class':'SimpleStrategy',
'Replication_factor':1};

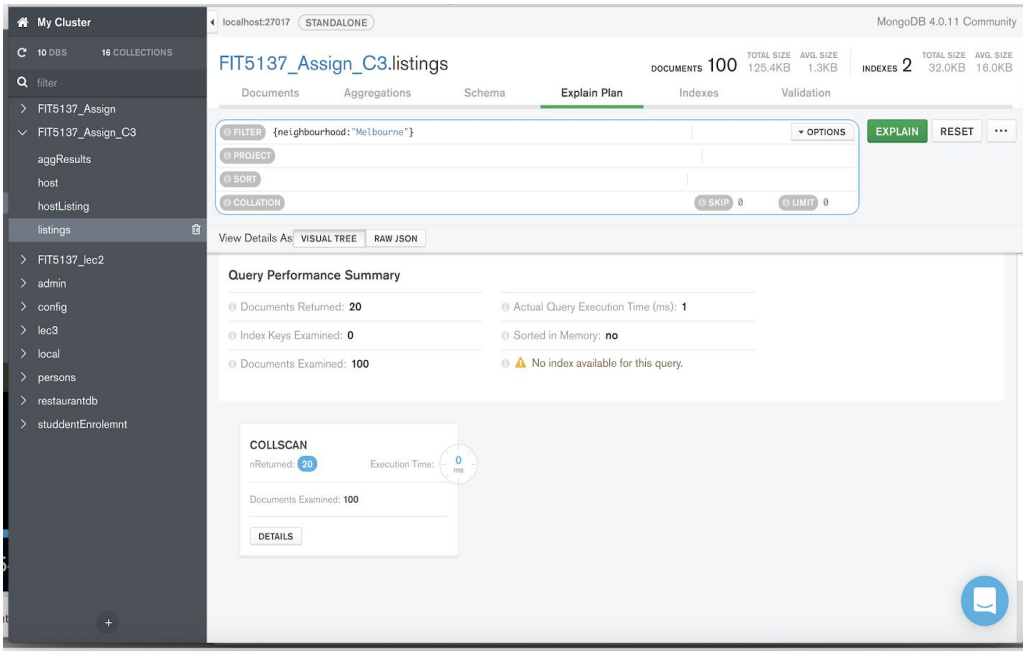
USE fit5137_assign_c3;

CREATE TABLE review(listing_id int, id int, date date, reviewer_id int, reviewer_name text, review_scores_rating int, comments text, PRIMARY KEY (id)) ;

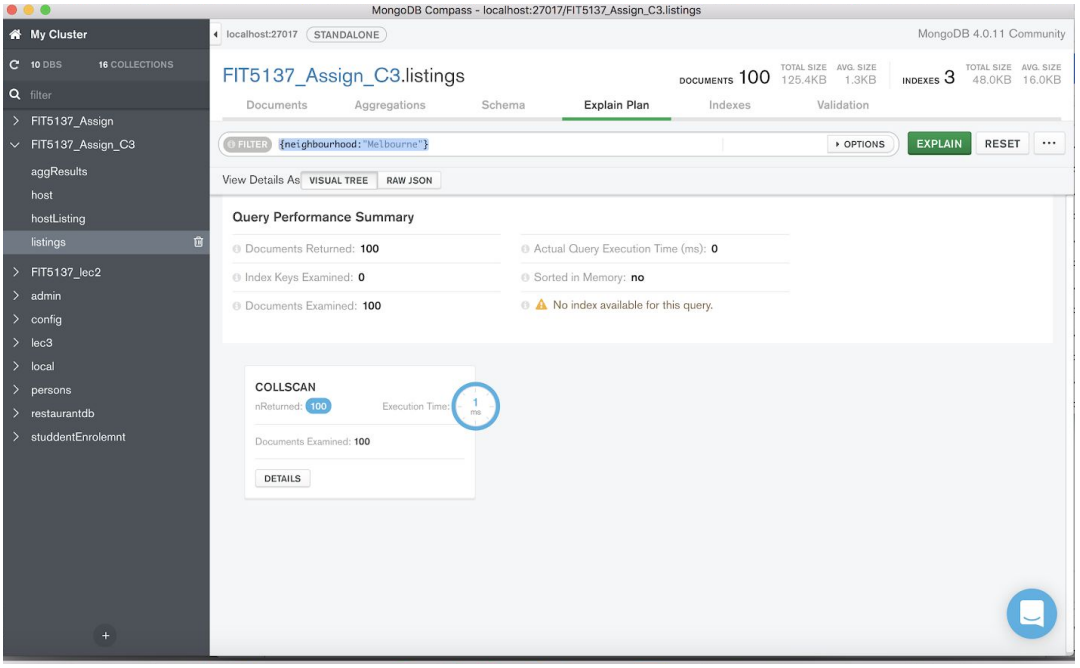
COPY review(listing_id,id,date,reviewer_id,reviewer_name,review_scores_rating,comments) FROM '/Users/sarah/Downloads/assignment_data_3/review.csv' WITH DELIMITER=',' AND HEADER=TRUE;

Creating Indexes:

1. Based on frequently using of neighbourhood to see the listing and locations we have created this index: db.listings.createIndex({"neighbourhood":1})
   Without indexing, the time is 1 ms.



And this is with indexing the time has reduced:



2. For the compound index, we have created an index of price and minimum nights, thus, they can query based on the price and nights to know the total price and how many nights they can book this accommodation: db.listings.createIndex({"price":1,"minimum_nights":1})

3. Based on our queries on the task 3 most of the queries is based on price and neighbourhood, thus, to allow better and sufficient querying we have created this query: db.listings.createIndex({"price":1,"neighbourhood":1})
4. Based on frequently using of name to see the listing we have created this index: db.listings.createIndex({"name":1})

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

Referencing:

```
db.listing.aggregate([
        { $match: {"last_review" : { $gte: ISODate("2018-12-01T00:00:00") }, "last_review"
: { $lte: ISODate("2018-12-31T00:00:00") }} },
        { $count : "totalAccommodations" }
]).pretty()
```

```
> db.listing.aggregate([
... { $match: {"last_review" : { $gte: ISODate("2018-12-01T00:00:00") }, "last_review" : { $lt
e: ISODate("2018-12-31T00:00:00") }} },
... { $count : "totalAccommodations" }
... ]).pretty()
{ "totalAccommodations" : 32 }
```

Embedding:

```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $match: {'hostList.last_review' : { $gte: ISODate("2018-12-01T00:00:00") },
'hostList.last_review' : { $lte: ISODate("2018-12-31T00:00:00") }} },
        { $count : "totalAccommodations" }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $match: {'hostList.last_review' : { $gte: ISODate("2018-12-01T00:00:00") }, 'hostList.la
st_review' : { $lte: ISODate("2018-12-31T00:00:00") }} },
... { $count : "totalAccommodations" }
... ]).pretty()
{ "totalAccommodations" : 32 }
```

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

Referencing:

```
db.listings.aggregate([{ $match : {"neighbourhood": "Port Phillip"}},{ $group:{_id:
"$neighbourhood", avg_price:{$avg:"$price"}}}])
```

```
{ "_id" : "Port Phillip", "avg_price" : 134.5 }
>
```

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"},{ $match : {"hostList.neighbourhood": "Port Phillip"}},{ $group:{_id: "$hostList.neighbourhood", avg_price:{$avg:"$hostList.price"}}}]).pretty()

```
{ "_id" : "Port Phillip", "avg_price" : 134.5 }
>
```

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

Referencing:

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

```
> db.listing.aggregate([
... { $group: { _id: "$neighbourhood", avgReviews: { $avg: "$reviews_per_month" } } },
... { $sort : {avgReviews : -1} },
...  { $limit: 10 }
... ]).pretty()
{ "_id" : "Stonnington", "avgReviews" : 2.1233333333333335 }
{ "_id" : "Brimbank", "avgReviews" : 1.9449999999999998 }
{ "_id" : "Yarra", "avgReviews" : 1.65375 }
{ "_id" : "Monash", "avgReviews" : 1.2850000000000001 }
{ "_id" : "Maribyrnong", "avgReviews" : 1.09 }
{ "_id" : "Melbourne", "avgReviews" : 1.0510526315789475 }
{ "_id" : "Casey", "avgReviews" : 0.905 }
{ "_id" : "Hobsons Bay", "avgReviews" : 0.855 }
{ "_id" : "Kingston", "avgReviews" : 0.8433333333333334 }
{ "_id" : "Port Phillip", "avgReviews" : 0.7835714285714286 }
>
```

Embedding:

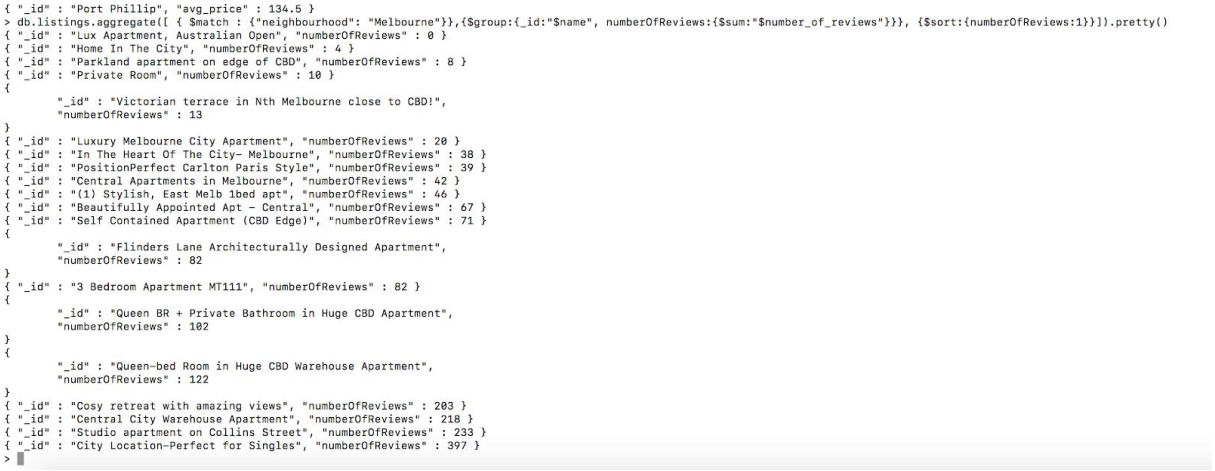
```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $group: { _id: '$hostList.neighbourhood', avgReviews: { $avg:
'$hostList.reviews_per_month' } } },
        { $sort : {avgReviews : -1} },
        { $limit: 10 }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $group: { _id: '$hostList.neighbourhood', avgReviews: { $avg: '$hostList.reviews_per_mon
th' } } },
... { $sort : {avgReviews : -1} },
...  { $limit: 10 }
... ]).pretty()
{ "_id" : "Stonnington", "avgReviews" : 2.1233333333333335 }
{ "_id" : "Brimbank", "avgReviews" : 1.9449999999999998 }
{ "_id" : "Yarra", "avgReviews" : 1.65375 }
{ "_id" : "Monash", "avgReviews" : 1.2850000000000001 }
{ "_id" : "Maribyrnong", "avgReviews" : 1.09 }
{ "_id" : "Melbourne", "avgReviews" : 1.0510526315789475 }
{ "_id" : "Casey", "avgReviews" : 0.905 }
{ "_id" : "Hobsons Bay", "avgReviews" : 0.855 }
{ "_id" : "Kingston", "avgReviews" : 0.8433333333333334 }
{ "_id" : "Port Phillip", "avgReviews" : 0.7835714285714286 }
>
```

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

Referencing:

db.listings.aggregate([{ $match : {"neighbourhood": "Melbourne"}},{$group:{_id:"$name", numberOfReviews:{$sum:"$number_of_reviews"}}}, {$sort:{numberOfReviews:1}}]).pretty()



```
{ "_id" : "Port Phillip", "avg_price" : 134.5 }
> db.listings.aggregate([ { $match : {"neighbourhood": "Melbourne"}},{$group:{_id:"$name", numberOfReviews:{$sum:"$number_of_reviews"}}}, {$sort:{numberOfReviews:1}}]).pretty()
{ "_id" : "Lux Apartment, Australian Open", "numberOfReviews" : 0 }
{ "_id" : "Home In The City", "numberOfReviews" : 4 }
{ "_id" : "Parkland apartment on edge of CBD", "numberOfReviews" : 8 }
{ "_id" : "Private Room", "numberOfReviews" : 10 }
{
        "_id" : "Victorian terrace in Nth Melbourne close to CBD!",
        "numberOfReviews" : 13
}
{ "_id" : "Luxury Melbourne City Apartment", "numberOfReviews" : 20 }
{ "_id" : "In The Heart Of The City- Melbourne", "numberOfReviews" : 38 }
{ "_id" : "PositionPerfect Carlton Paris Style", "numberOfReviews" : 39 }
{ "_id" : "Central Apartments in Melbourne", "numberOfReviews" : 42 }
{ "_id" : "(1) Stylish, East Melb 1bed apt", "numberOfReviews" : 46 }
{ "_id" : "Beautifully Appointed Apt – Central", "numberOfReviews" : 67 }
{ "_id" : "Self Contained Apartment (CBD Edge)", "numberOfReviews" : 71 }
{
        "_id" : "Flinders Lane Architecturally Designed Apartment",
        "numberOfReviews" : 82
}
{ "_id" : "3 Bedroom Apartment MT111", "numberOfReviews" : 82 }
{
        "_id" : "Queen BR + Private Bathroom in Huge CBD Apartment",
        "numberOfReviews" : 102
}
{
        "_id" : "Queen-bed Room in Huge CBD Warehouse Apartment",
        "numberOfReviews" : 122
}
{ "_id" : "Cosy retreat with amazing views", "numberOfReviews" : 203 }
{ "_id" : "Central City Warehouse Apartment", "numberOfReviews" : 218 }
{ "_id" : "Studio apartment on Collins Street", "numberOfReviews" : 233 }
{ "_id" : "City Location-Perfect for Singles", "numberOfReviews" : 397 }
>
```

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"},{ $match : {"hostList.neighbourhood": "Melbourne"}},{ $group:{_id: "$hostList.name", numberOfReviews:{$sum:"$hostList.number_of_reviews"}}},{$sort:{numberOfReviews:1}}]).pretty()

```
> db.hostListing.aggregate([{$unwind:"$hostList"},{ $match : {"hostList.neighbourhood": "Melbourne"}},{ $group:{_id: "$hostLis
t.name", numberOfReviews:{$sum:"$hostList.number_of_reviews"}}},{$sort:{numberOfReviews:1}}]).pretty()
{ "_id" : "Lux Apartment, Australian Open", "numberOfReviews" : 0 }
{ "_id" : "Home In The City", "numberOfReviews" : 4 }
{ "_id" : "Parkland apartment on edge of CBD", "numberOfReviews" : 8 }
{ "_id" : "Private Room", "numberOfReviews" : 10 }
{
        "_id" : "Victorian terrace in Nth Melbourne close to CBD!",
        "numberOfReviews" : 13
}
{ "_id" : "Luxury Melbourne City Apartment", "numberOfReviews" : 20 }
{ "_id" : "In The Heart Of The City- Melbourne", "numberOfReviews" : 38 }
{ "_id" : "PositionPerfect Carlton Paris Style", "numberOfReviews" : 39 }
{ "_id" : "Central Apartments in Melbourne", "numberOfReviews" : 42 }
{ "_id" : "(1) Stylish, East Melb 1bed apt", "numberOfReviews" : 46 }
{ "_id" : "Beautifully Appointed Apt - Central", "numberOfReviews" : 67 }
{ "_id" : "Self Contained Apartment (CBD Edge)", "numberOfReviews" : 71 }
{ "_id" : "3 Bedroom Apartment MT111", "numberOfReviews" : 82 }
{
        "_id" : "Flinders Lane Architecturally Designed Apartment",
        "numberOfReviews" : 82
}
{
        "_id" : "Queen BR + Private Bathroom in Huge CBD Apartment",
        "numberOfReviews" : 102
}
{
        "_id" : "Queen-bed Room in Huge CBD Warehouse Apartment",
        "numberOfReviews" : 122
}
{ "_id" : "Cosy retreat with amazing views", "numberOfReviews" : 203 }
{ "_id" : "Central City Warehouse Apartment", "numberOfReviews" : 218 }
{ "_id" : "Studio apartment on Collins Street", "numberOfReviews" : 233 }
{ "_id" : "City Location-Perfect for Singles", "numberOfReviews" : 397 }
>
```

5. What is the most popular room type?

Referencing:

```
db.listing.aggregate([
        { $group: { _id: "$room_type", totalOccurrence: { $sum: 1 } } },
        { $sort: {totalOccurrence: -1} },
        { $limit: 1 }
]).pretty()
```

```
> db.listing.aggregate([
... { $group: { _id: "$room_type", totalOccurrence: { $sum: 1 } } },
... { $sort: {totalOccurrence: -1} },
... { $limit: 1 }
... ]).pretty()
{ "_id" : "Private room", "totalOccurrence" : 50 }
>
```

Embedding:

```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $group: { _id: '$hostList.room_type', totalOccurrence: { $sum: 1 } } },
        { $sort: {totalOccurrence: -1} },
        { $limit: 1 }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $group: { _id: '$hostList.room_type', totalOccurrence: { $sum: 1 } } },
... { $sort: {totalOccurrence: -1} },
... { $limit: 1 }
... ]).pretty()
{ "_id" : "Private room", "totalOccurrence" : 50 }
```

6. Where are the top 5 most expensive accommodations?

Referencing:

db.listings.aggregate([
{$project:{_id:0,neighbourhood:1,price:1}},{$sort:{"price":-1}},{$limit:5}]).pretty()

```
{ "neighbourhood" : "Melbourne", "price" : 701 }
{ "neighbourhood" : "Yarra Ranges", "price" : 500 }
{ "neighbourhood" : "Glen Eira", "price" : 419 }
{ "neighbourhood" : "Melbourne", "price" : 357 }
{ "neighbourhood" : "Port Phillip", "price" : 330 }
```

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"},{$project:{_id:0,"hostList.neighbourhood"
:1,"hostList.price":1}},{$sort:{"hostList.price":-1}},{$limit:5}]).pretty()

```
{ "hostList" : { "neighbourhood" : "Melbourne", "price" : 701 } }
{ "hostList" : { "neighbourhood" : "Yarra Ranges", "price" : 500 } }
{ "hostList" : { "neighbourhood" : "Glen Eira", "price" : 419 } }
{ "hostList" : { "neighbourhood" : "Melbourne", "price" : 357 } }
{ "hostList" : { "neighbourhood" : "Port Phillip", "price" : 330 } }
>
```

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

Referencing:

db.host.aggregate([
{ $lookup : { from : "listing", localField: "host_id", foreignField:"host_id", as:"hostList"}},
{ $match : { "host_name" : "Eleni" } },
{ $project : {_id : 0, 'hostList.id' : 1, 'hostList.name' : 1, 'hostList.street' : 1, }}
]).pretty()

```
> db.host.aggregate([
... { $lookup : {  from : "listing", localField: "host_id", foreignField:"host_id", as:"hostList"}},
... { $match : { "host_name" : "Eleni" } },
... { $project : {_id : 0, 'hostList.id' : 1, 'hostList.name' : 1, 'hostList.street' : 1,  }}
... ]).pretty()
{
        "hostList" : [
                {
                        "id" : 15246,
                        "name" : "Large private room-close to city",
                        "street" : "Thornbury, VIC, Australia"
                },
                {
                        "id" : 68482,
                        "name" : "Charming house inner Melbourne",
                        "street" : "Thornbury, VIC, Australia"
                }
        ]
}
>
```

Embedding:

```
db.host_listing.aggregate([
        { $match : { "host_name" : "Eleni" } },
        { $unwind : '$hostList' },
        { $project : {_id : 0, 'hostList.id' : 1, 'hostList.name' : 1, 'hostList.street' : 1,  } }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $match : { "host_name" : "Eleni" } },
... { $unwind : '$hostList' },
... { $project : {_id : 0, 'hostList.id' : 1, 'hostList.name' : 1, 'hostList.street' : 1,  } }
... ]).pretty()
{
        "hostList" : {
                "id" : 15246,
                "name" : "Large private room-close to city",
                "street" : "Thornbury, VIC, Australia"
        }
}
{
        "hostList" : {
                "id" : 68482,
                "name" : "Charming house inner Melbourne",
                "street" : "Thornbury, VIC, Australia"
        }
}
>
```

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

Referencing:

db.host.aggregate([{ $lookup : { from : "listings", localField: "host_id", foreignField:"host_id", as:"listings"}}, {$match:{"host_response_time":"within a few hours"}}]).pretty()

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"},{$match:{"host_response_time":"within a few hours"}}]).pretty()


```
{
        "_id" : ObjectId("5d7a2794b68c6a1119a03a91"),
        "host_id" : 189682,
        "host_url" : "https://www.airbnb.com/users/show/189682",
        "host_name" : "Belinda",
        "host_verifications" : [
                "email",
                "phone",
                "facebook",
                "reviews",
                "jumio",
                "offline_government_id",
                "selfie",
                "government_id",
                "identity_manual",
                "work_email"
        ],
        "host_since" : ISODate("2010-08-03T00:00:00Z"),
        "host_location" : "Melbourne, Victoria, Australia",
        "host_response_time" : "within a few hours",
        "host_is_superhost" : false,
        "hostList" : {
                "_id" : ObjectId("5d78325fd2454d9286b305bd"),
                "id" : 43414,
                "name" : "Home In The City",
                "summary" : "",
                "listing_url" : "https://www.airbnb.com/rooms/43414",
                "picture_url" : "https://a0.muscache.com/im/pictures/a50d7057-1523-4389-b6a0-2f20e6dc7733.jpg?aki_policy=large",
                "host_id" : 189682,
                "neighbourhood" : "Melbourne",
                "street" : "East Melbourne, VIC, Australia",
                "zipcode" : 3002,
                "latitude" : -37.81027,
                "longitude" : 144.98592,
                "room_type" : "Private room",
                "amenities" : "{TV,Pool,Kitchen,Gym,Elevator,Heating,Washer,Dryer,\"Smoke detector\",\"First aid kit\",Essentials,Shampoo,Hangers,\"Hair dryer\",Iron,\"Laptop friendly workspace\",Bathtub,
\"Room-darkening shades\",\"Hot water\",\"Bed linens\",\"Extra pillows and blankets\",Microwave,\"Coffee maker\",Refrigerator,Dishwasher,\"Dishes and silverware\",\"Cooking basics\",Oven,Stove,\"Single le
vel home\",\"BBQ grill\",\"Patio or balcony\",\"Garden or backyard\",\"Long term stays allowed\",\"No stairs or steps to enter\",\"Flat path to guest entrance\",\"Well-lit path to entrance\",\"Host greets
 you\",\"Paid parking on premises\"}",
                "price" : 99,
                "extra_people" : 25,
                "minimum_nights" : 15,
                "number_of_reviews" : 4,
                "last_review" : ISODate("2019-01-26T00:00:00Z"),
                "reviews_per_month" : 0.23,
                "calculated_host_listings_count" : 1,
                "availability_365" : 62
        }
}
{
```


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

Referencing:

db.host.aggregate([

{ $lookup : { from : "listing", localField: "host_id", foreignField:"host_id", as:"hostList"}},

{ $match : { $and: [{ "host_name" : "Colin" }, { 'hostList.amenities' : /internet/i }, { 'hostList.amenities' : /gym/i }]}}

]).pretty()

```
> db.host.aggregate([
... { $lookup : {  from : "listing", localField: "host_id", foreignField:"host_id", as:"hostList"}},
... { $match : { $and: [ { "host_name" : "Colin" }, { 'hostList.amenities' : /internet/i }, { 'hostLi
st.amenities' : /gym/i } ]}}
... ]).pretty()
>
```

There is no listing with internet and gym.

Embedding

db.host_listing.aggregate([

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

{ $unwind : '$hostList' },

{ $match : { $and: [{ 'hostList.amenities' : /internet/i }, { 'hostList.amenities' : /gym/i }]}}

]).pretty()

```
> db.host_listing.aggregate([
... { $match : { "host_name" : "Colin" } },
... { $unwind : '$hostList' },
... { $match : { $and: [ { 'hostList.amenities' : /internet/i }, { 'hostList.amenities' : /gym/i } ]}}
... ]).pretty()
>
>
```

There is no listing with internet and gym.

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

Referencing:

db.listings.aggregate([{$match:{neighbourhood:"Clayton",name:{ $regex:/Beautiful/}}}, {$project:{_id:0,price:1,"room_type":1}}]).pretty()

Nothing appears. Cause in and conditions both matches should be true and there is no homes in Clayton.

```
> db.listings.aggregate([{$match:{neighbourhood:"Clayton",name:{ $regex:/Beautiful/}}}, {$project:{_id:0,price:1,"room_type":1}}]).pretty()
>
```

Embedding: Nothing appears as well.

db.host_listing.aggregate([{$unwind:"$hostList"},{$match:{"hostList.neighbourhood":"Clayton","hostList.name":{ $regex:/Beautiful/}}}, {$project:{_id:0,"hostList.price":1,"hostList.room_type":1}}]).pretty()

```
> db.hostListing.aggregate([{$unwind:"$hostList"},{$match:{"hostList.neighbourhood":"Clayton","hostList.name":{ $regex:/Beautiful/}}}, {$project:{_id:0,"hostList.price":1,"hostList.room_type":1}}]).pretty()
>
```

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.

Referencing:

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

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

Embedding:

```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $project: { personName: { $concat: [
                "$hostList.name",
                ", ",
                "$hostList.street",
                 ", ",
                { $toString : "$hostList.zipcode" },
                ", ",
                "$hostList.neighbourhood" ] } } },
        { $project: {_id: 0} },
        { $sort: { personName : 1} }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $project: { personName: { $concat: [
... "$hostList.name",
... ", ",
... "$hostList.street",
...  ", ",
... { $toString : "$hostList.zipcode" },
... ", ",
... "$hostList.neighbourhood" ] } } },
... { $project: {_id: 0} },
... { $sort: { personName : 1} }
... ]).pretty()
{
        "personName" : "(1) Stylish, East Melb 1bed apt, East Melbourne, VIC, Australia, 3002, Melbourne"
}
{
        "personName" : "**just renovated** Bright & Spacious StKildaEast**, St Kilda East, VIC, Australia, 3183, P
ort Phillip"
}
{
        "personName" : "2 bedrooms-ideal for friends/family, Prahran, VIC, Australia, 3181, Stonnington"
}
{
        "personName" : "3 Bedroom Apartment MT111, Southbank, VIC, Australia, 3006, Melbourne"
}
{
        "personName" : "3 Level Loft with Views close to City, 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.

Referencing:

db.listings.aggregate([{$sort:{"availability_365":-1}},
{$project:{_id:0,"availability_365":1,name:1, current : new ISODate()}},{$limit:1}]).pretty()

```
{
        "name" : "Beautiful Room & House",
        "availability_365" : 365,
        "current" : ISODate("2019-09-17T11:23:42.946Z")
}
```

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"},{$sort:{"hostList.availability_365":-1}},
{$project:{_id:0,"hostList.availability_365":1,"hostList.name":1, current : new ISODate()}},{$limit:1}]).pretty()

```
{
        "hostList" : {
                "name" : "Beautiful Room & House",
                "availability_365" : 365
        },
        "current" : ISODate("2019-09-17T11:30:18.979Z")
}
>
```

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.

Referencing:

```
db.listing.aggregate([
        { $match: {'price' : { $gt: 50 } } },
        { $group : {_id : "$neighbourhood", price : {$first : "$price"}, uniqueNeighbourhood : { $push : "$neighbourhood" } } },
        { $sort : {uniqueNeighbourhood : -1} },
        { $project: { uniqueNeighbourhood : 0 } }
]).pretty()
```

```
> db.listing.aggregate([
... { $match: {'price' : { $gte: 50 } } },
... { $group : {_id : "$neighbourhood", price : {$first : "$price"}, uniqueNeighbourhood : { $push : "$neighbourhood" } } },
... { $sort : {uniqueNeighbourhood : -1} },
... { $project: { uniqueNeighbourhood : 0 } }
... ]).pretty()
{ "_id" : "Yarra Ranges", "price" : 81 }
{ "_id" : "Yarra", "price" : 138 }
{ "_id" : "Wyndham", "price" : 71 }
{ "_id" : "Stonnington", "price" : 98 }
{ "_id" : "Port Phillip", "price" : 69 }
{ "_id" : "Moreland", "price" : 89 }
{ "_id" : "Monash", "price" : 98 }
{ "_id" : "Melton", "price" : 72 }
{ "_id" : "Melbourne", "price" : 69 }
{ "_id" : "Maribyrnong", "price" : 65 }
{ "_id" : "Manningham", "price" : 61 }
{ "_id" : "Kingston", "price" : 71 }
{ "_id" : "Hobsons Bay", "price" : 50 }
{ "_id" : "Glen Eira", "price" : 99 }
{ "_id" : "Frankston", "price" : 59 }
{ "_id" : "Darebin", "price" : 50 }
{ "_id" : "Casey", "price" : 99 }
{ "_id" : "Boroondara", "price" : 61 }
{ "_id" : "Bayside", "price" : 299 }
{ "_id" : "Banyule", "price" : 98 }
>
```

Embedding:

```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $match: { 'hostList.price' : { $gt: 50 } } },
        { $group : { _id : "$hostList.neighbourhood", price : { $first : "$hostList.price"}, uniqueNeighbourhood : { $push : "$hostList.neighbourhood" } } },
        { $sort : {uniqueNeighbourhood : -1} },
        { $project: { uniqueNeighbourhood : 0 } }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $match: { 'hostList.price' : { $gt: 50 } } },
... { $group : { _id : "$hostList.neighbourhood", price : { $first : "$hostList.price"}, uniqueNeighb
ourhood : { $push : "$hostList.neighbourhood" } } },
... { $sort : {uniqueNeighbourhood : -1} },
... { $project: { uniqueNeighbourhood : 0 } }
... ]).pretty()
{ "_id" : "Yarra Ranges", "price" : 160 }
{ "_id" : "Yarra", "price" : 138 }
{ "_id" : "Wyndham", "price" : 71 }
{ "_id" : "Stonnington", "price" : 73 }
{ "_id" : "Port Phillip", "price" : 159 }
{ "_id" : "Moreland", "price" : 89 }
{ "_id" : "Monash", "price" : 98 }
{ "_id" : "Melton", "price" : 72 }
{ "_id" : "Melbourne", "price" : 99 }
{ "_id" : "Maribyrnong", "price" : 65 }
{ "_id" : "Manningham", "price" : 61 }
{ "_id" : "Kingston", "price" : 71 }
{ "_id" : "Hobsons Bay", "price" : 159 }
{ "_id" : "Glen Eira", "price" : 99 }
{ "_id" : "Frankston", "price" : 59 }
{ "_id" : "Darebin", "price" : 140 }
{ "_id" : "Casey", "price" : 99 }
{ "_id" : "Boroondara", "price" : 61 }
{ "_id" : "Bayside", "price" : 299 }
{ "_id" : "Banyule", "price" : 98 }
>
```

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.

Referencing:

db.host.aggregate([
{$project:{_id:0,"host_id":1,"host_name":1,"numberOfVerificationMethods":{ $cond: { if: { $isArray:"$host_verifications"}, then: { $size: "$host_verifications"}, else: "NA"} }}},{$sort:{numberOfVerificationMethods:-1}}]).pretty()

```
{
        "host_id" : 189682,
        "host_name" : "Belinda",
        "numberOfVerificationMethods" : 10
}
{
        "host_id" : 189684,
        "host_name" : "Allan",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 246509,
        "host_name" : "Fiona",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 668249,
        "host_name" : "Shirley",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 700065,
        "host_name" : "Marilyn",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 1349266,
        "host_name" : "Adam",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 182833,
        "host_name" : "Diana",
        "numberOfVerificationMethods" : 8
}
```


Embedding:

db.host_listing.aggregate([{$project:{_id:0,"host_id":1,"host_name":1,"numberOfVerificationMethods":{ $cond: { if: { $isArray:"$host_verifications"}, then: { $size: "$host_verifications"}, else: "NA"} }}},{$sort:{numberOfVerificationMethods:-1}}]).pretty()


```
{
        "host_id" : 189682,
        "host_name" : "Belinda",
        "numberOfVerificationMethods" : 10
}
{
        "host_id" : 189684,
        "host_name" : "Allan",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 246509,
        "host_name" : "Fiona",
        "numberOfVerificationMethods" : 9
}
{
        "host_id" : 668249,
        "host_name" : "Shirley",
        "numberOfVerificationMethods" : 9
}
```

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

For this question we have created a separate table :
CREATE TABLE q3tables(listing_id int, id int, date date, reviewer_id int, reviewer_name text, review_scores_rating int, comments text, PRIMARY KEY ((listing_id), date)) ;
And then copy everything to this table.
And we have specified the primary key here as listing_id and partitioning key date.

SELECT * FROM q3tables WHERE listing_id=10803 ORDER BY date DESC LIMIT 1;

listing_id	date	comments	id	review_scores_rating	reviewer_id	reviewer_name
10803	2019-06-19	Lindsay's place is conveniently situated very near a main tram-stop. The tram-ride into the CBD is around half-an-hour and once in the CBD the tram conveniently passes Southern Cross train Station. Close to the accommodation there is a cafe or two and a pub. There is also a flight of stairs up to the accommodation so this could be a bit awkward for some.	472107310	93	45038845	Adele

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

We have tried to create a new table and then used review_scores_rating as a primary key to allow the filter without using ALLOW FILTERING, because the conditions on review_scores_rating, but it didn't work, then we tried to create another table with listing_id as a primary key and use review_scores_rating as an index but it didn't work as well. I think there is a problem with > and < operators.
What I found that using < > operators in Cassandra, you have to use one = operator for different column. So I have created a table.
CREATE TABLE reviews(listing_id int, id int, date date, reviewer_id int, reviewer_name text, review_scores_rating int, comments text, PRIMARY KEY ((listing_id), review_scores_rating)) ;
COPY reviews (listing_id, id, date, reviewer_id, reviewer_name, review_scores_rating, comments) FROM '/Users/sarah/Downloads/assignment_data_3/review.csv' WITH DELIMITER=',' AND HEADER=TRUE;
SELECT * FROM test2 WHERE listing_id=10803 and review_scores_rating>=70 and review_scores_rating<=90;
So I have specified a listing id in this case and it works.

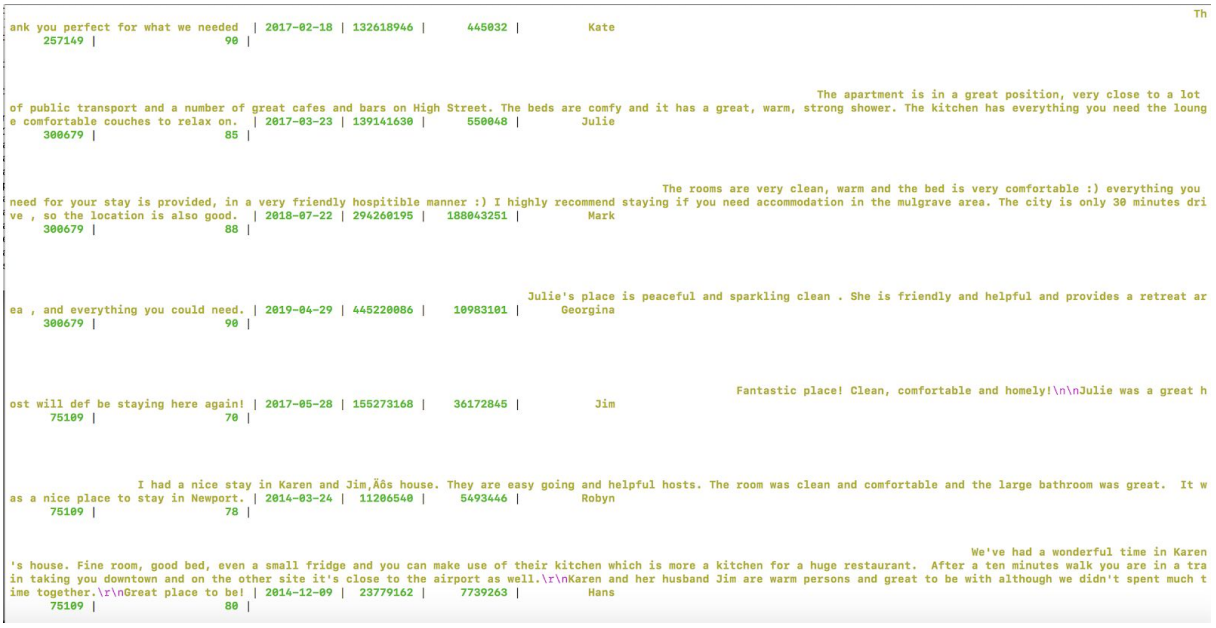
```
cqlsh:fit5137_assign_c3> SELECT * FROM test2 WHERE listing_id=10803 and review_scores_rating>=70 and review_scores_rating<=90;
```

listing_id	review_scores_rating	comments	date	id	reviewer_id	reviewer_name
10803	83	Lindsay is helpful and his house is a bit old but very nice. Tram just in front but considering how big is Melbourne area don't expect to feel close to the center or the beach. It was very convenient for me and I had a good time	2019-04-06	433155314	8156859	Lluv≠s
10803	84	He is so nice and respectful person.\nI really recommend you guys!!	2019-01-13	400698566	180444188	Shuhei
10803	85	The place is great and Lindsay is an amazing host! :) Plus, the location is amazing: the tram is right outside, super easy to get to the city, and the neighborhood is very trendy, full of restaurants and cafes. Definitely recommended!	2019-03-19	425758347	182480724	Ryota
10803	86	Very quirky apartment with the team stop conviently located just outside the door making traveling to and from the city easy	2018-09-08	319989971	163523483	Ping-Song
10803	87	Very good place with a really big bed and nice location closed to 96 tram stop. Lindsay is kind and helpful who has a good knowledge about China and mandarin. I would recommend the pizza restaurant downstairs if you wanna get something cheap and delicious for dinner.	2017-08-18	183632566	50768079	Kajia
10803	88	Sehr relaxte Unterkunft	2018-11-26	352879230	129236612	Roman
10803	89	Lindsay,Äôs place is comfortable and cosy. The tram is right outside and getting to the city is an easy 20 minutes. He is a kind and easygoing host - id recommend his place for people who want to experience melb as locals do.	2019-02-12	411558554	12991093	Kyle
10803	90	excellent host; the location is quite convenient as the place is directly within steps of the tram; yeah, value for money.	2018-10-23	340135544	117421771	Miho

(8 rows)

However, I think this should be for all review according to the requirement so i did this with allow filtering.

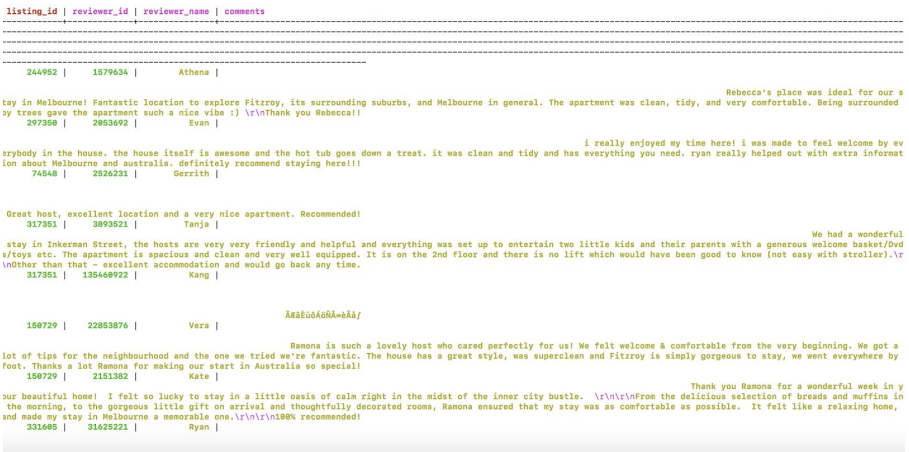
SELECT * FROM review WHERE review_scores_rating>=70 and review_scores_rating<=90 ALLOW FILTERING ;



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

This has the same issue as the previous one because with < > operations is not working even when creating a new table with new primary keys and partitioning key and even when specifying the secondary index

SELECT listing_id,reviewer_id,reviewer_name,comments FROM review WHERE review_scores_rating<50 ALLOW FILTERING ;



18. How many reviews left in 2015?

This question is the same as the previous two, so we used allow filtering.

select count(*) from review where date >= '2015-01-01' And date<='2015-12-31' Allow FILTERING;

```
 count
-------
  1072

(1 rows)
```

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

We have created a secondary index in this question:

CREATE INDEX ON review(date);

And then we use this query:

select reviewer_id, reviewer_name,Max(review_scores_rating) from review where date='2017-03-26';

```
 reviewer_id | reviewer_name | system.max(review_scores_rating)
-------------+---------------+----------------------------------
    69689025 |          Kira |                              100

(1 rows)
```

Then i create another index for reviewer_id and user id from previous step to get the review

CREATE INDEX on review (reviewer_id);

select * from review where reviewer_id=69689025;

```
 review_id | comments | date | listing_id | review_scores_rating | reviewer_id | reviewer_name
-----------+----------+------+------------+----------------------+-------------+--------------
 139908366 | My sisters and I highly recommend this apartment. It is everything you see in the pictures and read in the other reviews. The one thing you may not get a sense of until you get there is the awesomeness of its location, which is basically perfect for everything (especially if you are a Brunswick St fan!). We were able to walk to almost all of the locations on our must see/do /eat list and a quick tram ride to things slightly further afield. Light filled and airy due to the super cute balcony, which was perfect to sit out on any time of day. Rebecca is lovely, with fantastic communication. Book now and start getting excited about your stay! We managed to fit so much into our 3 days because our accommodation was so perfect and fantastically located. Thank you Rebecca. | 2017-03-26 | 244952 | 100 | 69689025 | Kira
```

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

select listing_id, reviewer_name, Max(review_scores_rating) as max_review from review ;

```
 listing_id | reviewer_name | max_review
------------+---------------+------------
      70328 |           Ali |        100
```

CREATE INDEX on review (review_scores_rating);

select listing_id,reviewer_name,review_scores_rating from review where review_scores_rating IN (100);

listing_id	reviewer_name	review_scores_rating
307630	Frances	100
150729	Doug	100
284210	Polly	100
74548	Julianti	100
51592	Gary Mart	100
72576	Robert	100
357740	Barbara	100
283257	Julio	100
232812	Lim	100
247140	Philip	100
157427	Angela	100
307630	Alice	100
75109	Aaron	100
43429	Eugene	100
150729	Daniel	100
257149	Leena	100
240410	Michelle	100
38271	Taryn	100
78143	Anthony	100
268849	Kevin	100
76867	Ashley	100
72576	Kelly	100

Five additional queries

1. Find the exact location (latitude and longitude) and street address of 10 listings that is available than 300 days and room type is private. Sort the results from highest availability to lowest availability.

Referencing:

```
db.listing.aggregate([
        { $match : { $and: [ {availability_365 : { $gt : 300 } }, {room_type : "Private room"} ] } },
        { $limit: 10 },
        { $project: { _id : 0, name : 1, latitude : 1, longitude : 1, availability_365 : 1 } },
        { $sort : {availability_365 : -1} }
]).pretty()
```

```
> db.listing.aggregate([
... { $match : { $and: [ {availability_365 : { $gt : 300 } }, {room_type : "Private room"} ] } },
... { $limit: 10 },
... { $project: { _id : 0, name : 1, latitude : 1, longitude : 1, availability_365 : 1 } },
... { $sort : {availability_365 : -1} }
... ]).pretty()
{
        "name" : "Beautiful Room & House",
        "latitude" : -37.77268,
        "longitude" : 145.09213,
        "availability_365" : 365
}
{
        "name" : "A Room Near the Park",
        "latitude" : -37.92817,
        "longitude" : 145.02518,
        "availability_365" : 365
}
{
        "name" : "Lux Apartment, Australian Open",
        "latitude" : -37.84499,
        "longitude" : 144.97929,
        "availability_365" : 365
}
{
        "name" : "Cool spot near chapel st!",
        "latitude" : -37.85077,
        "longitude" : 145.01013,
        "availability_365" : 365
}
{
        "name" : "Double Guest Room Ensuited",
        "latitude" : -37.83739,
        "longitude" : 144.96637,
        "availability_365" : 360
}
{
        "name" : "Kew Tranquility, Melbourne",
        "latitude" : -37.8037,
```

Embedding:

```
db.host_listing.aggregate([
        { $unwind : '$hostList' },
        { $match : { $and: [
                { 'hostList.availability_365' : { $gt : 300 } },
                { 'hostList.room_type' : "Private room"}
        ] } },
        { $limit: 10 },
                { $project: { _id : 0, "hostList.name" : 1, "hostList.latitude" : 1,
"hostList.longitude" : 1, "hostList.availability_365" : 1 } },
                { $sort : {"hostList.availability_365" : -1} }
]).pretty()
```

```
> db.host_listing.aggregate([
... { $unwind : '$hostList' },
... { $match : { $and: [
... { 'hostList.availability_365' : { $gt : 300 } },
... { 'hostList.room_type' : "Private room"}
... ] } },
... { $limit: 10 },
... { $project: { _id : 0, "hostList.name" : 1, "hostList.latitude" : 1, "hostList.longitude" : 1, "h
ostList.availability_365" : 1 } },
... { $sort : {"hostList.availability_365" : -1} }
... ]).pretty()
{
        "hostList" : {
                "name" : "Beautiful Room & House",
                "latitude" : -37.77268,
                "longitude" : 145.09213,
                "availability_365" : 365
        }
}
{
        "hostList" : {
                "name" : "A Room Near the Park",
                "latitude" : -37.92817,
                "longitude" : 145.02518,
                "availability_365" : 365
        }
}
{
        "hostList" : {
                "name" : "Lux Apartment, Australian Open",
                "latitude" : -37.84499,
                "longitude" : 144.97929,
                "availability_365" : 365
        }
}
{
        "hostList" : {
                "name" : "Cool spot near chapel st!",
                "latitude" : -37.85077,
```

2. Find the listings that it is either in Frankston or Port Phillip and the minimum nights is less than 4 nights, and show the name, the neighbourhood and the number of nights.

Referencing:

db.listings.aggregate([{$match:{minimum_nights:{$lt:4},$or:[{neighbourhood:"Frankston"}, {neighbourhood:"Port Phillip"}]}},{$project:{_id:0,name:1, neighbourhood:1 ,minimum_nights:1}}]).pretty()

```
{
        "name" : "Queen Room in Beautiful House",
        "neighbourhood" : "Frankston",
        "minimum_nights" : 2
}
{
        "name" : "St Kilda 1BR+BEACHSIDE+BALCONY+GARAGE+WIFI+AC",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 2
}
{
        "name" : "Melbourne BnB near City & Sports",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 1
}
{
        "name" : "St Kilda CENTRAL LUXE 2BR+PRIVATE COURTYARDS+WIFI",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 2
}
{
        "name" : "Blissful Beachside Port Melbourne Warehouse",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 2
}
{
        "name" : "City's edge Penthouse – private bath, views!",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 2
}
{
        "name" : "Elwood CHIC 1BR+WALK TO VILLAGE+PARKING+WIFI",
        "neighbourhood" : "Port Phillip",
        "minimum_nights" : 2
}
{
        "name" : "Large room in quiet Victorian home",
```

Embedding:

```
db.host_listing.aggregate([{$unwind:"$hostList"},
{$match:{"hostList.minimum_nights":{$lt:4} ,$or:[ {"hostList.neighbourhood":"Frankston"},
{"hostList.neighbourhood":"Port
Phillip"}]}},{$project:{_id:0,"hostList.name":1,"hostList.neighbourhood":1,"hostList.minimum_
nights":1}}]).pretty()
```

```
{
        "hostList" : {
                "name" : "Queen Room in Beautiful House",
                "neighbourhood" : "Frankston",
                "minimum_nights" : 2
        }
}
{
        "hostList" : {
                "name" : "Melbourne BnB near City & Sports",
                "neighbourhood" : "Port Phillip",
                "minimum_nights" : 1
        }
}
{
        "hostList" : {
                "name" : "Blissful Beachside Port Melbourne Warehouse",
                "neighbourhood" : "Port Phillip",
                "minimum_nights" : 2
        }
}
{
        "hostList" : {
                "name" : "Large room in quiet Victorian home",
                "neighbourhood" : "Port Phillip",
                "minimum_nights" : 3
        }
}
{
        "hostList" : {
                "name" : "Albert Park: Between St Kilda & CBD – Perfect spot",
                "neighbourhood" : "Port Phillip",
                "minimum_nights" : 2
        }
}
```

3. Count the number of accommodations in each neighbourhood and show it from High to Low

Referencing:

```
db.listings.aggregate([ {$group:{_id:{neighbourhood: "$neighbourhood"},
totalAccomedtions:{$sum:1}}},{ $sort: {totalAccomedtions: -1} }]).pretty()
```

```
{ "_id" : { "neighbourhood" : "Melbourne" }, "totalAccomedtions" : 20 }
{ "_id" : { "neighbourhood" : "Yarra" }, "totalAccomedtions" : 16 }
{ "_id" : { "neighbourhood" : "Port Phillip" }, "totalAccomedtions" : 14 }
{ "_id" : { "neighbourhood" : "Darebin" }, "totalAccomedtions" : 7 }
{ "_id" : { "neighbourhood" : "Yarra Ranges" }, "totalAccomedtions" : 5 }
{ "_id" : { "neighbourhood" : "Stonnington" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Moreland" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Boroondara" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Hobsons Bay" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Glen Eira" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Kingston" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Brimbank" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Monash" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Frankston" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Casey" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Banyule" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Bayside" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Wyndham" }, "totalAccomedtions" : 1 }
{ "_id" : { "neighbourhood" : "Whitehorse" }, "totalAccomedtions" : 1 }
{ "_id" : { "neighbourhood" : "Maribyrnong" }, "totalAccomedtions" : 1 }
Type "it" for more
```

Embedding:

db.host_listing.aggregate([{$unwind:"$hostList"}, {$group:{ _id:{"neighbourhood":"$hostList.neighbourhood"}, totalAccomedtions:{$sum:1}}},{ $sort: {totalAccomedtions: -1} }]).pretty()

```
{ "_id" : { "neighbourhood" : "Melbourne" }, "totalAccomedtions" : 20 }
{ "_id" : { "neighbourhood" : "Yarra" }, "totalAccomedtions" : 16 }
{ "_id" : { "neighbourhood" : "Port Phillip" }, "totalAccomedtions" : 14 }
{ "_id" : { "neighbourhood" : "Darebin" }, "totalAccomedtions" : 7 }
{ "_id" : { "neighbourhood" : "Yarra Ranges" }, "totalAccomedtions" : 5 }
{ "_id" : { "neighbourhood" : "Stonnington" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Boroondara" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Moreland" }, "totalAccomedtions" : 4 }
{ "_id" : { "neighbourhood" : "Hobsons Bay" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Kingston" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Glen Eira" }, "totalAccomedtions" : 3 }
{ "_id" : { "neighbourhood" : "Casey" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Frankston" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Bayside" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Banyule" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Brimbank" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Monash" }, "totalAccomedtions" : 2 }
{ "_id" : { "neighbourhood" : "Wyndham" }, "totalAccomedtions" : 1 }
{ "_id" : { "neighbourhood" : "Whitehorse" }, "totalAccomedtions" : 1 }
{ "_id" : { "neighbourhood" : "Manningham" }, "totalAccomedtions" : 1 }
Type "it" for more
> it
{ "_id" : { "neighbourhood" : "Maribyrnong" }, "totalAccomedtions" : 1 }
{ "_id" : { "neighbourhood" : "Melton" }, "totalAccomedtions" : 1 }
>
```

4. Display all the reviews for Julie.

Create a new table:
CREATE TABLE reviewerName(listing_id int, id int, date date, reviewer_id int, reviewer_name text, review_scores_rating int, comments text, PRIMARY KEY (reviewer_name)) ;

COPY reviewerName(listing_id,id,date,reviewer_id,reviewer_name,review_scores_rating,comments) FROM '/Users/sarah/Downloads/assignment_data_3/review.csv' WITH DELIMITER=',' AND HEADER=TRUE;

select * from reviewerName where reviewer_name='Julie';

```
 reviewer_name | comments

                                          | date       | id        | listing_id | review_scores_rating | reviewer_id
---------------+--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------
------------------------------------------+------------+-----------+------------+----------------------+-------------
         Julie | We learned SO much with Julie!  She really took time to understand where we are in our photography journeys and helped
 focus our skills to move forward.  She opened our eyes to different ways to arrange composition and targeted challenges to help learn
our cameras.  We had such a wonderful time, and I would jump at a chance to spend more time with her.   Any question we had she was up
for answering.  Love love loved the experience!! | 2019-02-07 | 409423192 |     210568 |                  100 |    33656879
```

5. Find reviewer id and reviewer name and review score rating for listing id 9835.

CREATE TABLE listingReview(listing_id int, id int, date date, reviewer_id int, reviewer_name text, review_scores_rating int, comments text, PRIMARY KEY (listing_id)) ;

```
COPY listingReview(listing_id,id,date,reviewer_id,reviewer_name,review_scores_rating,comments)
FROM '/Users/sarah/Downloads/assignment_data_3/review.csv' WITH DELIMITER=',' AND
HEADER=TRUE;
```

```
SELECT reviewer_id, reviewer_name, review_scores_rating FROM listingReview WHERE
listing_id = 9835;
```

reviewer_id	reviewer_name	review_scores_rating
26184717	Rosalind	94

C.4. Database Comparison

According to MonashBnB needs, we have decided to choose Cassandra database. To be able to use the created MongoDB database, we have to convert MongoDB to Cassandra and to do that there is a process to do that. There are several strategies to that, however, we will choose 2 Branch Migration strategy. This strategy allows to migrate the database smoothly, thus, the users of the database won't notice that and there will be no downtime of the system.

Here is the steps of doing that, you have to create a write and read branches:

1. Write a 'writes' Branch: Create a new branch from the main branch, and for the shifting components from the MongoDB you have to create a table in the new branch. Then, you have to copy the writes that you did for components your shifting from MongoDB into Cassandra tables.After that, write a migration script that contains all the existing MongoDB object and write that script to Cassandra. Finally, execute the write branch and this will allow writing data in both databases, and you can check during this time the data being written if it's in a good format and as you wanted.

2. Write a 'reads' Branch: creating a read branch from the write branch. This approach simply replaces the MongoDB read and write for the component you're shifting and then transform it into Cassandra, and it takes longer time than write branch, cause the reading queries may different from two databases. It's recommended to do the read branch before executing the write branch to avoid any mistake, cause the read branch validate the data modelling so if you forget anything in write branch you can go back and add it before executing it. This reduces the chances of data inconsistency to occur.

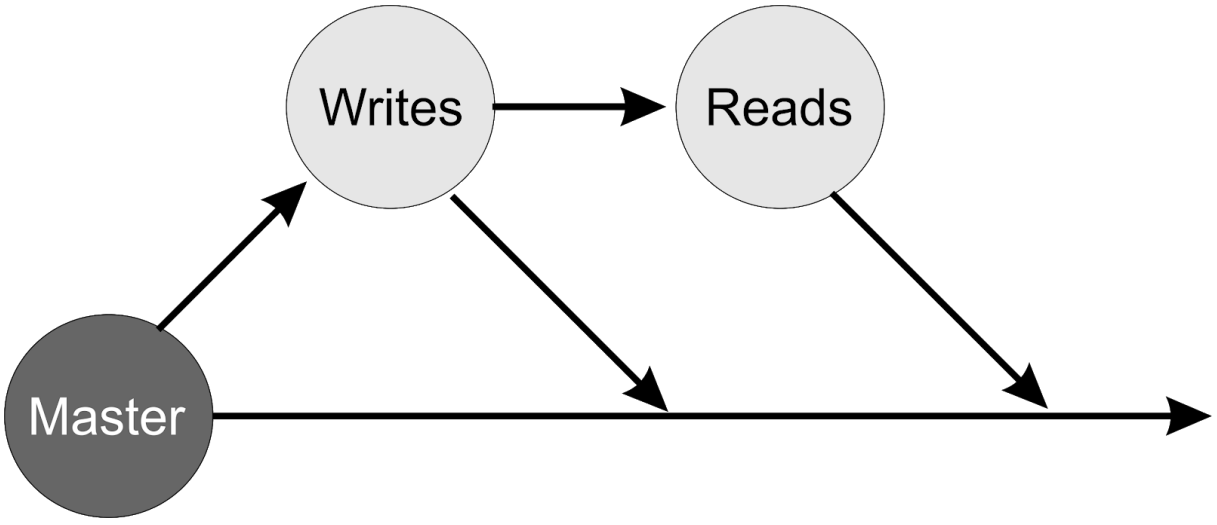


*Figure 1. (*DataStax, 2019)

Comparison of MongoDB and Cassandra

	Cassandra	**MongoDB**
Flexibility of creation	No need to model the column schema in advance as there is not a necessity of having the same set of columns in a row like relational databases.	The schema can be created dynamically without a definitive structure.
Scalability	It can be scaled horizontally in a linear fashion. By adding new machines, read and write throughput can be increased.	The scalability is limited.
Performance	High performance with faster queries and the ability of multi-query execution.	Performance is limited compared to Cassandra. Read operations not that faster and it is quite slow when multi-query execution.
Consistency	The data will eventually be available to all the nodes but in some nodes may contain the old data for a certain period.	Consistency is high as it supports master-slave replication and replica sets.
Availability	As the data is distributed across all nodes of the cluster, the data is available without failure.	To achieve the consistency availability has deprioritized. Writes to a secondary node will be blocked if the primary node is not functional.
Familiarity	Very similar to relational databases. It query language CQL is somewhat similar to SQL.	It's query language is not familiar like Cassandra..

Table 1: Comparison of MongoDB and Cassandra

There are several key reasons behind the selection Cassandra for MonashBnB data. The major reason is that Cassandra offers a somewhat traditional table structure. The data that MonshBnB going to store is more structured and it can clearly identify a natural row-column distribution of these data. Even if required the data can be stored in relational databases with some normalization. Therefore, the data is

much more suitable to store in a column-oriented database such as Cassandra. On the other hand, MongoDB supports rich data. MongoDB is better for rich data specially the objects with properties and nested values. Therefore, Cassandra is naturally supported for MonashBnB data than MongoDB.

In addition, there were other key reasons for selecting Cassandra over MongoDB as described in table 1. Due to the architecture of Cassandra, there will no unavailability of the data stored in Cassandra. Further, to support future demand, MonashBNB can add more machine and improve the scalability quite easily and cost-effectively. Moreover, in terms of speed as well as consistency Cassandra is better than the MongoDB. Further, most of the database developers are familiar with SQL and Cassandra's query language, CQL is similar to SQL. Therefore, developers will find it more confidence in working with Cassandra.

By considering all of the reasons described above, Cassandra can be recommended for storing the MonashBnB data.

References:

DataStax. (2019). MongoDB to Cassandra Migrations. Retrieved from: https://academy.datastax.com/planet-cassandra/mongodb-to-cassandra-migration

EDUCBA. (2019). MongoDB vs Cassandra. Retrieved from: https://www.educba.com/mongodb-vs-cassandra/