Submitted by

Manali Choudhary

Rohan Nishchal

**FIT5137, Advanced Database Technology**

**Group Assignment - Sem 2/2019**

**GROUP ASSIGNMENT COVER SHEET**

|  |  |  |
| --- | --- | --- |
| **Student ID Number** | **Surname** | **Given Names** |
| 30151198 | Choudhary | **Manali Prakash** |
| 30383633 | Nischal | **Rohan** |
|  |  |  |
|  |  |  |
| **\*** Please include the names of all other group members. | | |
| **Unit name and code** | **FIT5137, Advanced Database Technology** | |
| **Title of assignment** | **Group Assignment - Sem 2/2019** | |
| **Lecturer/tutor** | Chaluka | |
| **Tutorial day and time** | 18th September, 2019 11.55 pm | **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**18th September, 2019 | | **Date submitted**18th September, 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 The material used can be from any source (staff, students or the internet, published and unpublished works).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](http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-policy.html) and [Procedures](http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-managing-plagiarism-collusion-procedures.html). * 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:  1. provide to another member of faculty and any external marker; and/or 2. submit it to a text matching software; and/or 3. 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 ....................Manali Choudhary, Rohan Nischal.......................................................... Date………*18th September, 2019 *……………………………*  \* delete (iii) if not applicable |

Signature \_\_ *Manali Choudhary* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date:\_\_\_\_ 18th September, 2019 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Rohan Nischal* \_\_\_\_\_\_\_\_\_\_\_\_Date:\_\_\_\_ 18th September, 2019 \_\_\_\_

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 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](mailto:privacyofficer@adm.monash.edu.au) |

**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 |
| 30151198 | Manali Choudhary | 50% |
| 30383633 | Rohan Nischal | 50% |
|  |  |  |
|  |  |  |

**2 DECLARATION**

**Percentage of contribution:**

**I. Name: Manali, ID:** 30151198**, Contribution: 50%**

**II. Name: Rohan, ID:** 30383633**, Contribution: 50%**

**List of parts that each student did:**

**I. Manali:**

**□ Task C.1 (MongoDB create collections and insert for Host, Cassandra insert)**

**□ Task C.2(Q1-3, Q6)**

**□ Task C.3(Q1-7 queries of Mongodb,Q16-18,indexing)**

**□ Task C.4(comparison and analysis)**

**II. Rohan:**

**□ Task C.1 (MongoDB create collections and insert for Listing, Cassandra create table, Cassandra insert)**

**□ Task C.2(Q4-5, Q7)**

**□ Task C.3(Q8-15 queries of Mongodb,Q18-20,extra queries)**

**□ Task C.4(flowchart and analysis)**

**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**

Manali Choudhary

**Signatures**

Rohan Nischal

DayMonth Year

18 / 09 / 2019

**Date**

**--C.1. Database Design.--**

**INSERT for MongoDB**

use FIT5137\_Assign;

db.createCollection("listing");

db.createCollection("host");

**For Host:**

db.host.insertOne({

host\_id : "MONHOS01",

host\_name : "Manju",

host\_verifications : ["email","phone","reviews"],

host\_since : new Date("2009-08-21"),

host\_location : { suburb : "Clayton", state : "Victoria", country : "Australia" },

host\_response\_rate : "within a day",

is\_superhost : false

});

**For Listing:**

db.listing.insertOne({

listing\_id : "MONLST01",

name : "Monash Beautiful House",

host\_id : "MONHOS14",

neighbourhood : "Manningham",

address : { suburb : "Clayton", state : "VIC", postcode : "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\_per\_night : 61,

price\_for\_extra\_people : 22,

min\_nights\_for\_booking : 1,

availability : 365

}

);

**Note: The other insert queries are similar and are mentioned in the file FIT5137\_Assign\_C1\_MongoDB.txt**



**INSERT for Cassandra**

CREATE KEYSPACE FIT5137\_Assign

WITH replication = {'class': 'SimpleStrategy',

'replication\_factor' : 1};

use FIT5137\_Assign;

CREATE TABLE review ( listing\_id text,

review\_id text,

review\_date date,

sequence text,

reviewer\_id int,

reviewer\_name text,

review\_scores\_rating int,

satisfied\_reason set<text>,

comments text,

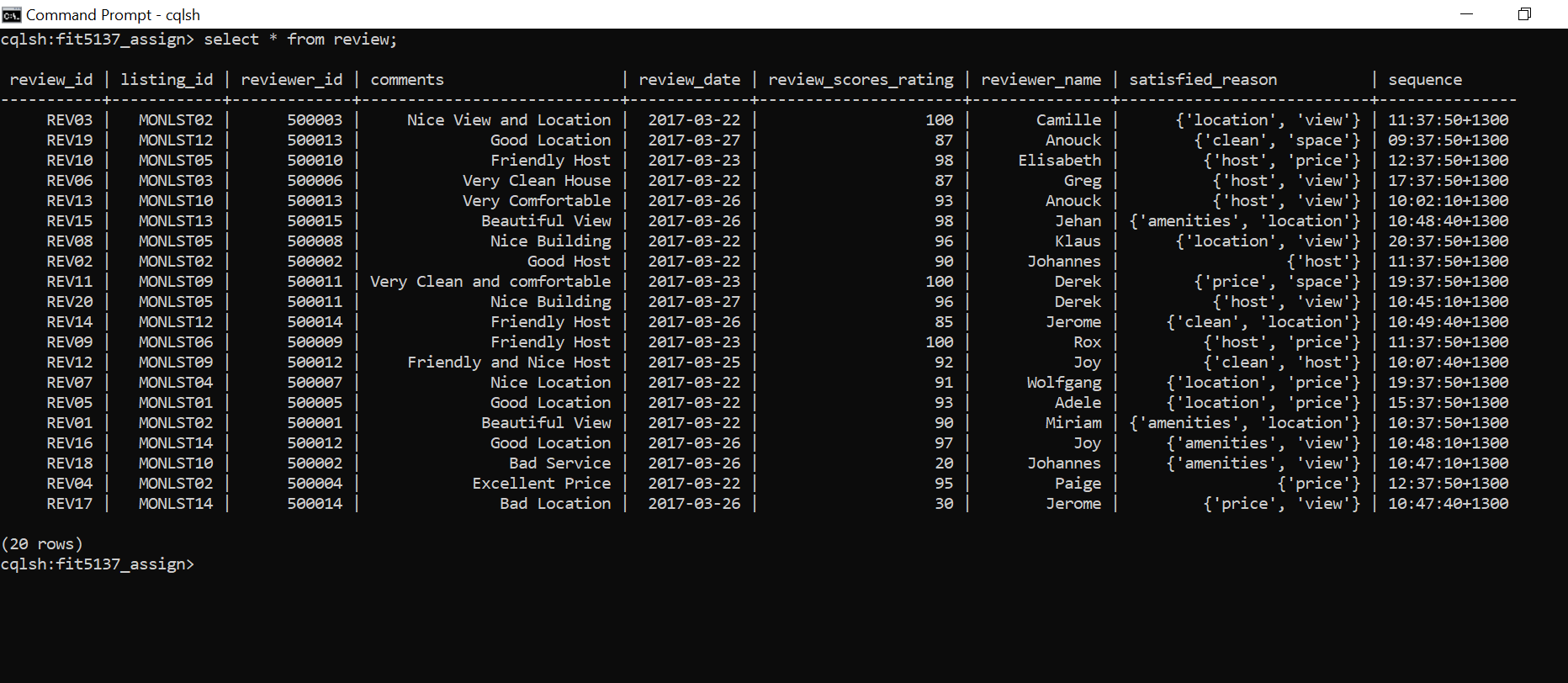
PRIMARY KEY ((review\_id),listing\_id,reviewer\_id)

) ;

INSERT INTO review (listing\_id, review\_id,review\_date,sequence,reviewer\_id,reviewer\_name,review\_scores\_rating,satisfied\_reason,comments)

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

**Note: The other insert queries are similar and are mentioned in the file FIT5137\_Assign\_C1\_Cassandra.txt**



**--C.2. Database Modifications.--**

**Q1**

db.host.updateOne(

{ host\_name : "Adam" },

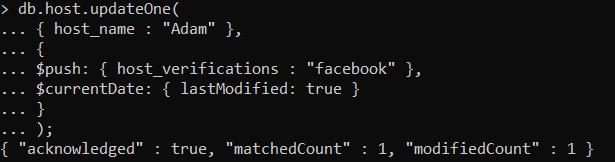
{

$push: { host\_verifications : "facebook" },

$currentDate: { lastModified: true }

}

);



**Q2**

db.host.insertMany([

{

host\_id : "MONHOS11",

host\_name : "Alison",

host\_verifications : ["email","phone","facebook","reviews"],

host\_since : new Date("2019-01-09"),

host\_location : { suburb : "Caulfield", state : "Victoria", country : "Australia" },

host\_response\_rate : "within an hour",

is\_superhost : false

},

{

host\_id : "MONHOS12",

host\_name : "Mike",

host\_verifications : ["email","phone"],

host\_since : new Date("2019-01-09"),

host\_location : { suburb : "Clayton", state : "Victoria", country : "Australia" },

host\_response\_rate : "within a day",

is\_superhost : true

},

{

host\_id : "MONHOS13",

host\_name : "Robyn",

host\_verifications : ["facebook","reviews"],

host\_since : new Date("2019-01-09"),

host\_location : { suburb : "Berwick", state : "Victoria", country : "Australia" },

host\_response\_rate : "within an hour",

is\_superhost : false

},

{

host\_id : "MONHOS14",

host\_name : "Daniel",

host\_verifications : ["email","manual offline","work email"],

host\_since : new Date("2019-01-09"),

host\_location : { suburb : "Frankston", state : "Victoria", country : "Australia" },

host\_response\_rate : "within a day",

is\_superhost : true

},

{

host\_id : "MONHOS15",

host\_name : "Ron",

host\_verifications : ["facebook"],

host\_since : new Date("2019-01-09"),

host\_location : { suburb : "Caulfield", state : "Victoria", country : "Australia" },

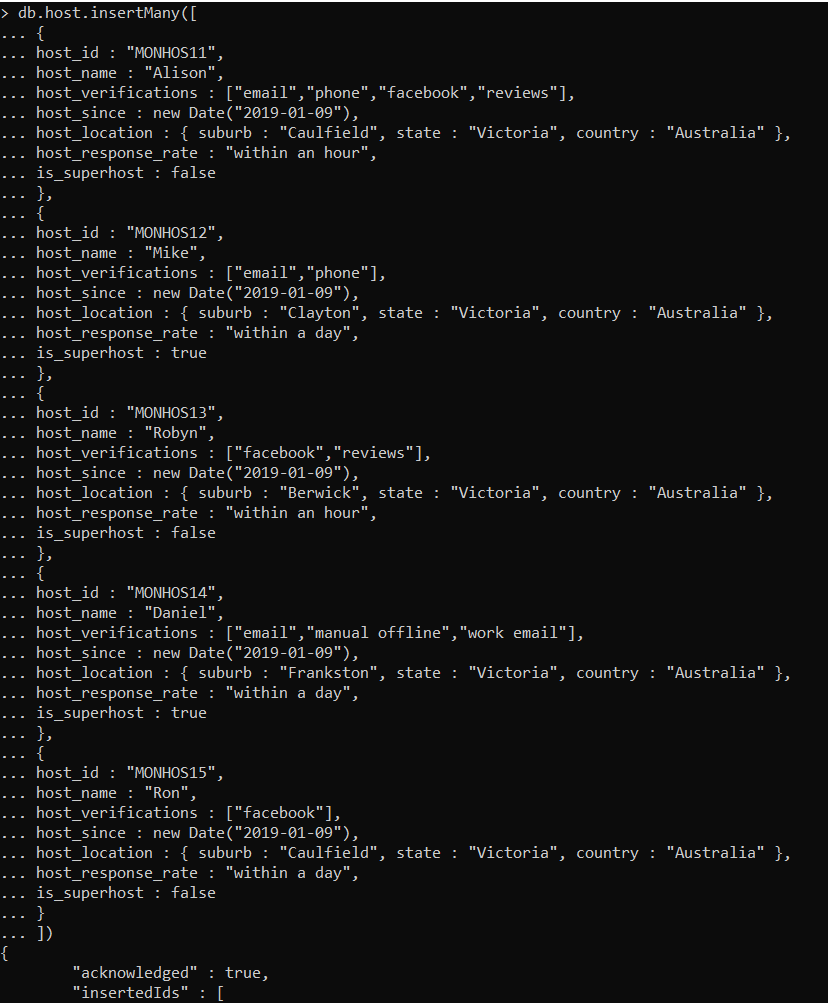
host\_response\_rate : "within a day",

is\_superhost : false

}

])

;



**Q3**

db.host.updateOne(

{host\_response\_rate:"within an hour",is\_superhost:false},

{

$set: { is\_superhost : true },

$currentDate: { lastModified: true }

}

);

**Q4**

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

**Q5**

db.listing.updateMany(

{neighbourhood:"Monash"},

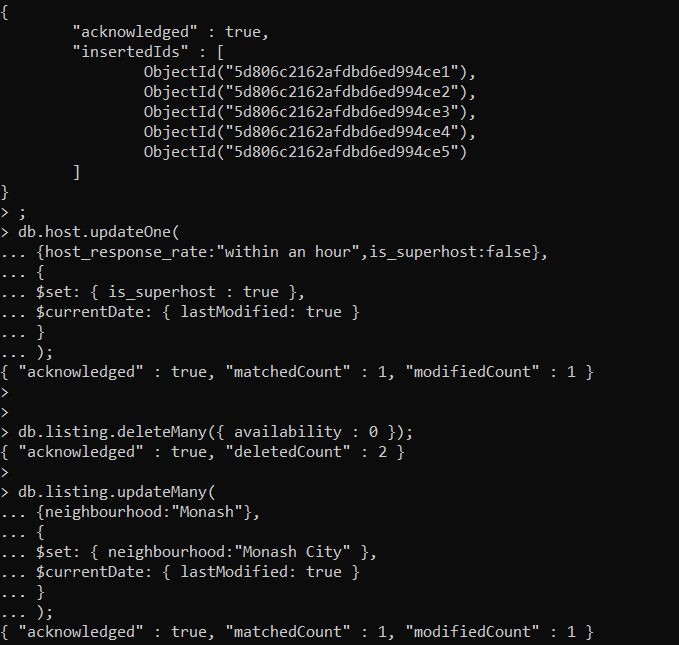
{

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

$currentDate: { lastModified: true }

}

);



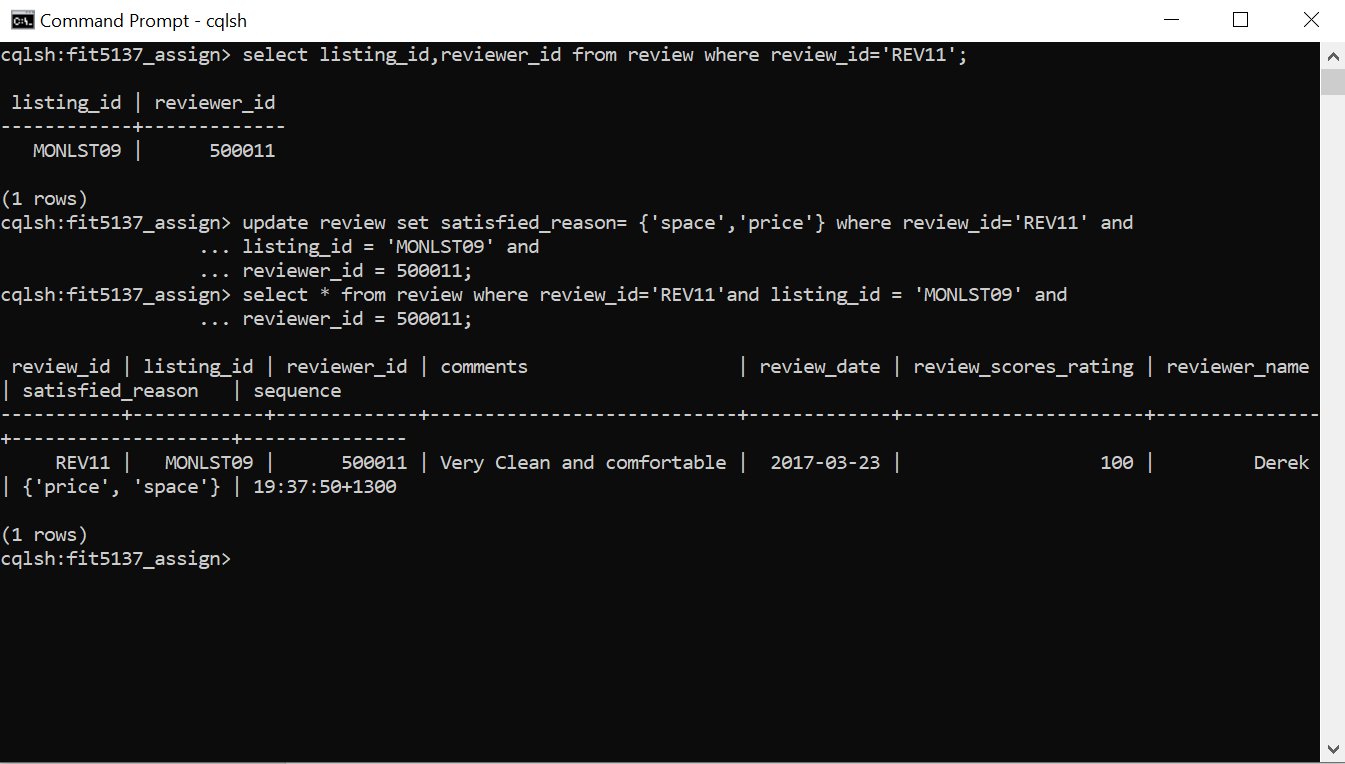
**Q6**

select listing\_id,reviewer\_id from review where review\_id='REV11';

update review set satisfied\_reason= {'space','price'} where review\_id='REV11' and

listing\_id = 'MONLST09' and

reviewer\_id = 500011;

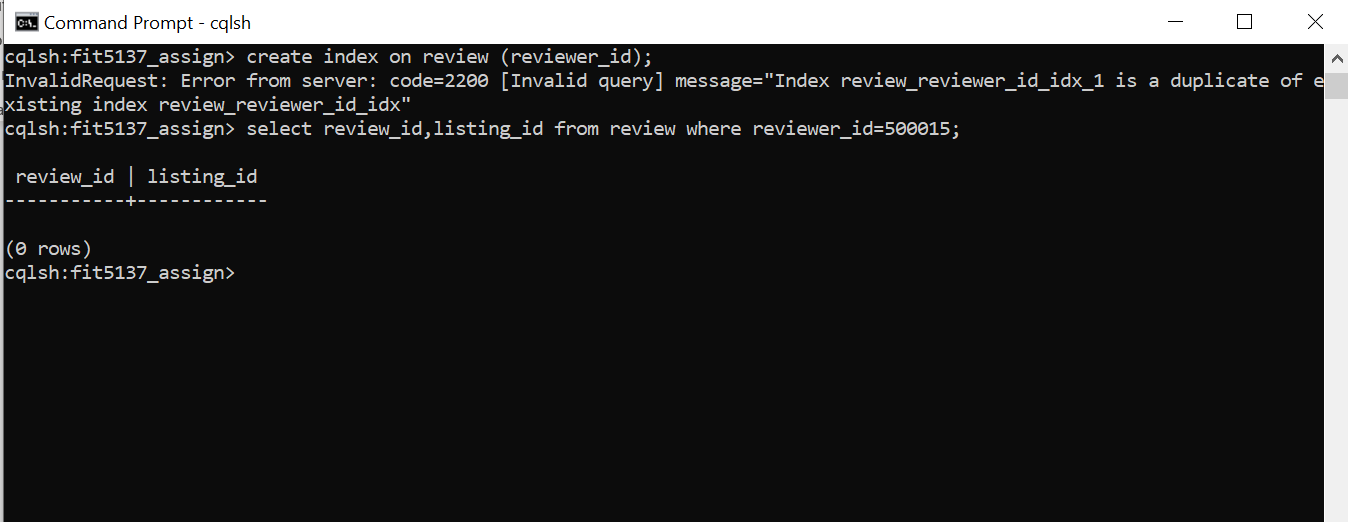


**Q7**

create index on review (reviewer\_id);

select review\_id,listing\_id from review where reviewer\_id=500015;

delete from review where reviewer\_id = 500015 and review\_id= 'REV15' and listing\_id= 'MONLST13';



**--C.3. Queries.--**

**Importing JSON files for MongoDB**

mongoimport --db FIT5137\_Assign\_C3 --collection host --file D:\assignment\_data\host.json

mongoimport --db FIT5137\_Assign\_C3 --collection listing --file D:\assignment\_data\listing.json

**Importing CSV file for 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 ((listing\_id),id,reviewer\_id)

) ;

COPY FIT5137\_Assign\_C3.review (listing\_id,id,date,reviewer\_id,reviewer\_name,review\_scores\_rating,comments)

FROM 'C:\Users\Manali Choudhary\Documents\Sem2\adb\assignment\_data\assignment\_data\review.csv' WITH HEADER=TRUE;

select count(\*) from review;

**//Windows OS**

**Create embedding model**

db.listing.aggregate([

{ $lookup : {

from : "host",

localField : "host\_id",

foreignField : "host\_id",

as : "host\_info"

} },

{

$out:"listing\_host"

}

]);

**Q1**

**Reference**

db.listing.aggregate([

{$project: {month: {$month: "$last\_review"},year: {$year: "$last\_review"},id:1}},

{ $match: {month: 12, year:2018 } },

{$count: "no\_of\_accomodations"}

]);

**Embedded**

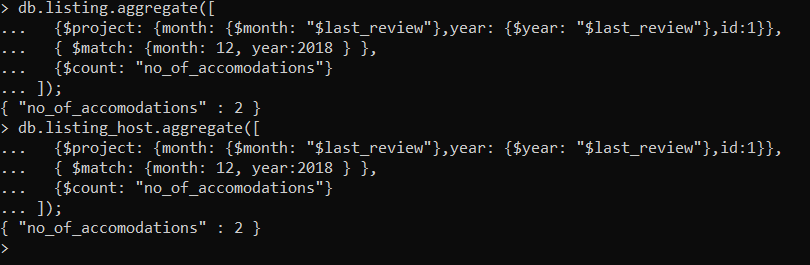
db.listing\_host.aggregate([

{$project: {month: {$month: "$last\_review"},year: {$year: "$last\_review"},id:1}},

{ $match: {month: 12, year:2018 } },

{$count: "no\_of\_accomodations"}

]);



**Q2**

**Reference**

db.listing.aggregate([

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

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

]).pretty();

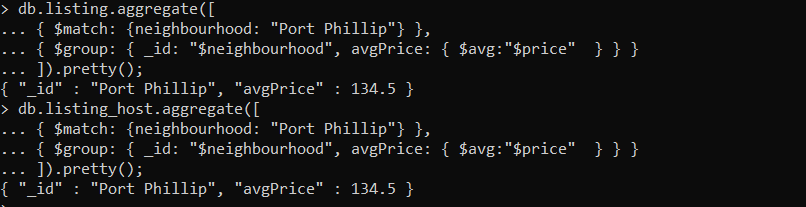
**Embedded**

db.listing\_host.aggregate([

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

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

]).pretty();



**Q3**

**Reference**

db.listing.aggregate([

{ $group: { \_id: "$neighbourhood", reviews\_per\_month: { $avg:"$reviews\_per\_month" } } },

{ $sort : { "reviews\_per\_month" : -1 } },

{ $limit : 10 },

{ $project: {neighbourhood : "$\_id",\_id:0} }

]).pretty();

**Embedded**

db.listing\_host.aggregate([

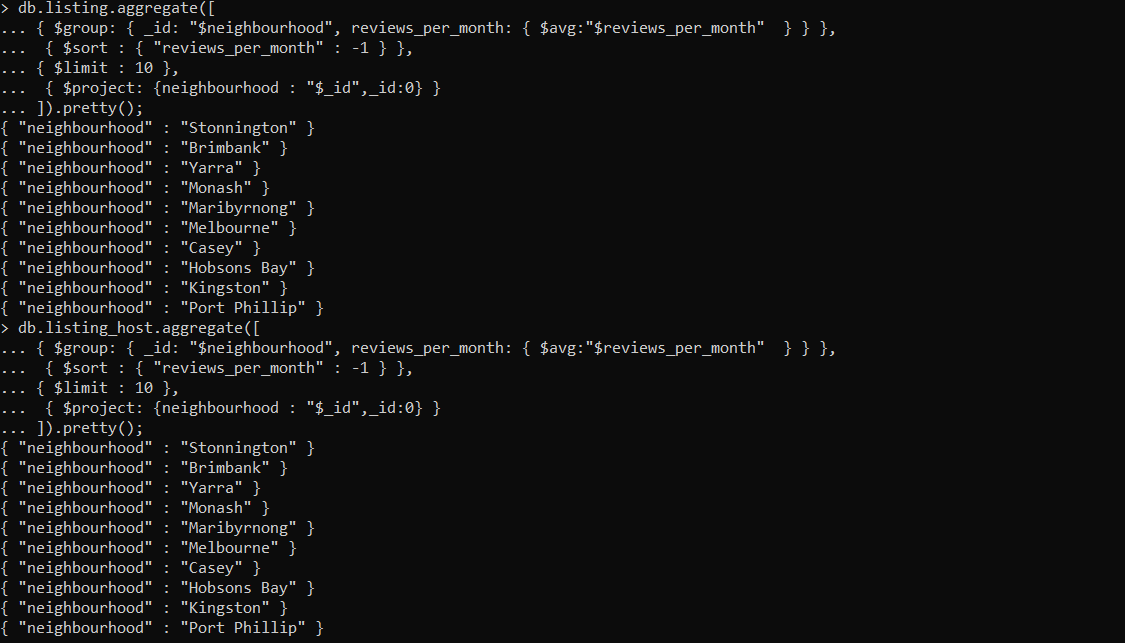
{ $group: { \_id: "$neighbourhood", reviews\_per\_month: { $avg:"$reviews\_per\_month" } } },

{ $sort : { "reviews\_per\_month" : -1 } },

{ $limit : 10 },

{ $project: {neighbourhood : "$\_id",\_id:0} }

]).pretty();



**Q4**

**Reference**

db.listing.aggregate([

{ $group: { \_id: null, maxRange: { $max:"$number\_of\_reviews" }, minRange: { $min:"$number\_of\_reviews" } } },

{ $project: { \_id:0, minNumberOfReviews : "$minRange", maxNumberOfReviews : "$maxRange" }}

]).pretty();

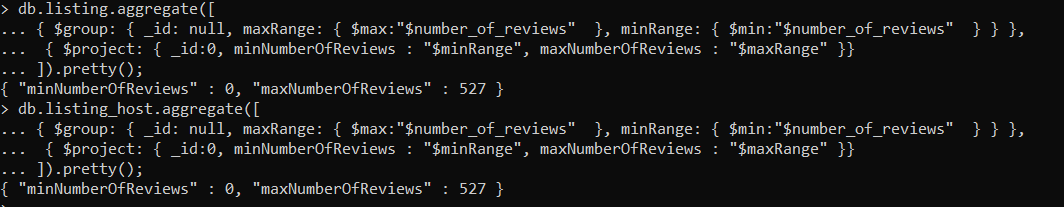
**Embedded**

db.listing\_host.aggregate([

{ $group: { \_id: null, maxRange: { $max:"$number\_of\_reviews" }, minRange: { $min:"$number\_of\_reviews" } } },

{ $project: { \_id:0, minNumberOfReviews : "$minRange", maxNumberOfReviews : "$maxRange" }}

]).pretty();



**Q5**

**Reference**

db.listing.aggregate([

{ $group : { \_id : "$room\_type", noOfOccurence : { $sum: 1 } } },

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

{$limit : 1},

{ $project: {room\_type : "$\_id",\_id:0} }

]).pretty();

**Embedded**

db.listing\_host.aggregate([

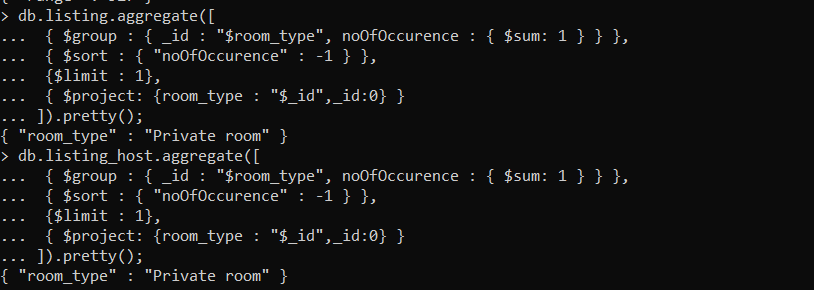
{ $group : { \_id : "$room\_type", noOfOccurence : { $sum: 1 } } },

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

{$limit : 1},

{ $project: {room\_type : "$\_id",\_id:0} }

]).pretty();



**Q6**

**Reference**

db.listing.aggregate([

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

{ $limit : 5 },

{ $project: {neighbourhood:1,\_id:0} }

]).pretty();

**Embedded**

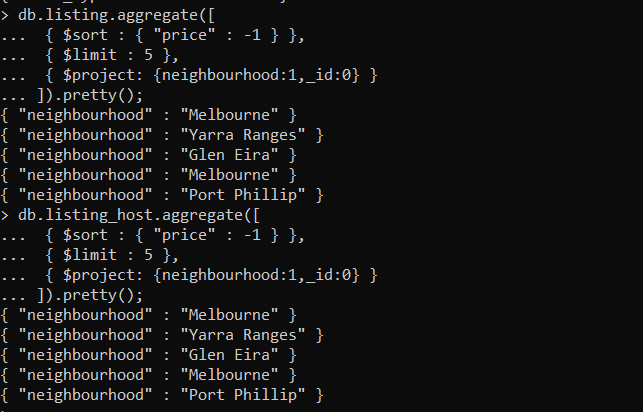
db.listing\_host.aggregate([

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

{ $limit : 5 },

{ $project: {neighbourhood:1,\_id:0} }

]).pretty();



**Q7**

**Reference**

db.host.aggregate([

{ $lookup : {

from : "listing",

localField : "host\_id",

foreignField : "host\_id",

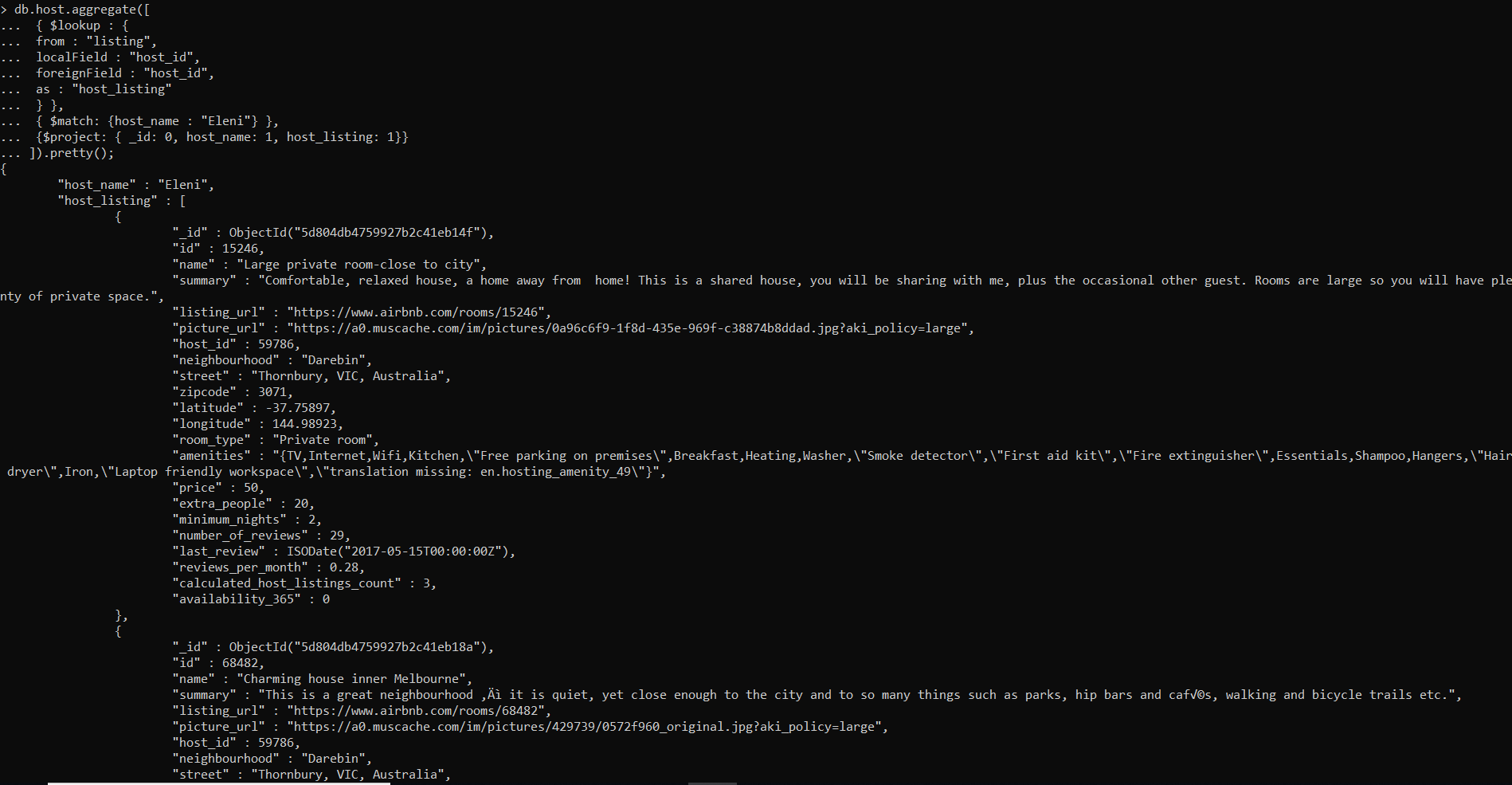
as : "host\_listing"

} },

{ $match: {host\_name : "Eleni"} },

{$project: { \_id: 0, host\_name: 1, host\_listing: 1}}

]).pretty();



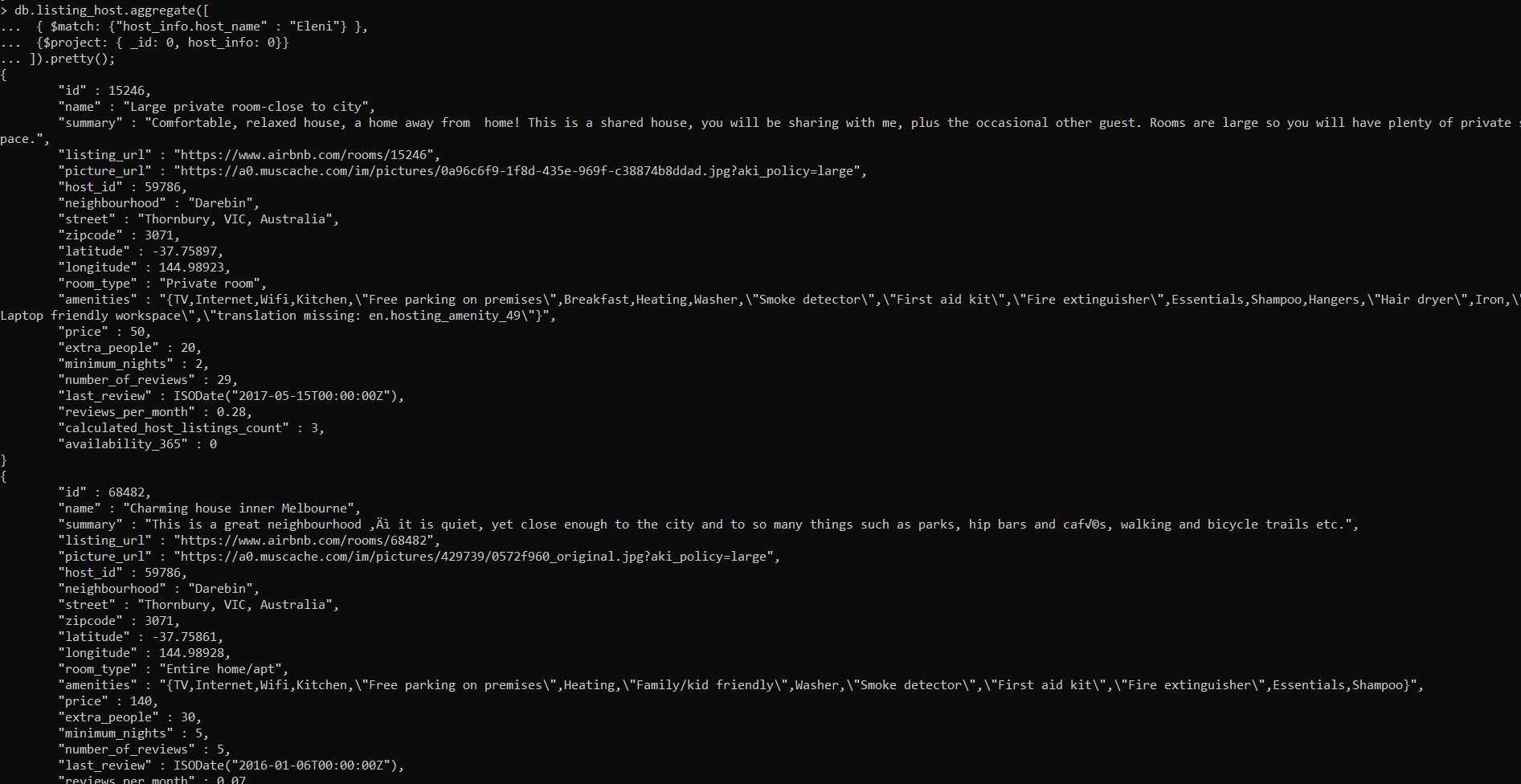
**Embedded**

db.listing\_host.aggregate([

{ $match: {"host\_info.host\_name" : "Eleni"} },

{$project: { \_id: 0, host\_info: 0}}

]).pretty();



**Q8**

**Reference**

db.host.aggregate([

{ $lookup : {

from : "listing",

localField : "host\_id",

foreignField : "host\_id",

as : "host\_listing"

} },

{ $match:{$and:[ {host\_response\_time : "within a few hours","host\_listing.room\_type" : {$regex: "Entire home"}}]} },

{ $project: {host\_listing : 1,\_id:0} }

]).pretty();



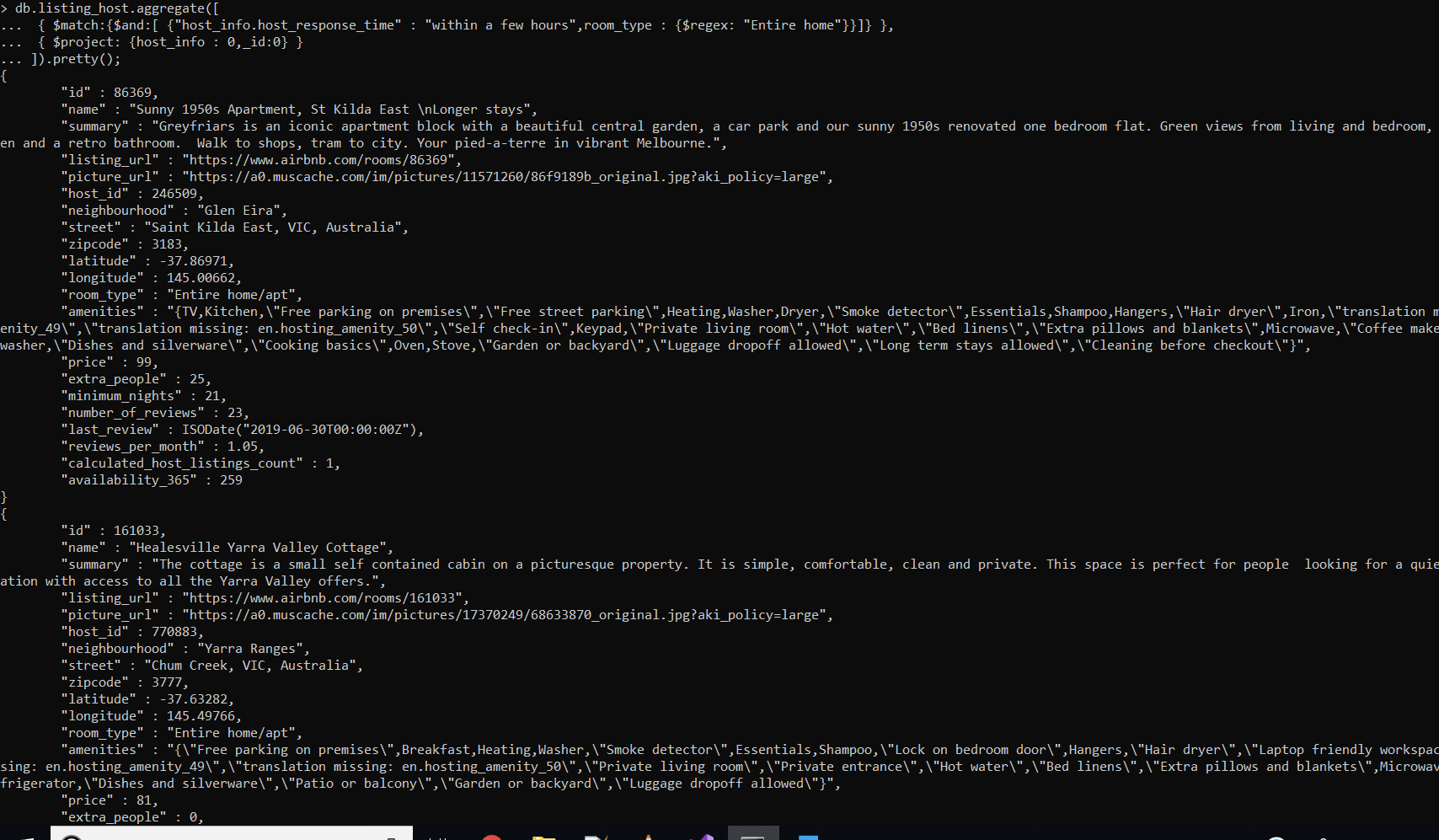
**Embedded**

db.listing\_host.aggregate([

{ $match:{$and:[ {"host\_info.host\_response\_time" : "within a few hours",room\_type : {$regex: "Entire home"}}]} },

{ $project: {host\_info : 0,\_id:0} }

]).pretty();



**Q9**

**Reference**

db.host.aggregate([

{ $lookup : {

from : "listing",

localField : "host\_id",

foreignField : "host\_id",

as : "host\_listing"

} },

{ $match: {$and:[ {host\_name : "Colin", "host\_listing.amenities" : {$regex : "Internet"}, "host\_listing.amenities" : {$regex : "Gym Access"} }] }},

{ $project: {host\_listing : 1,\_id:0} }

]).pretty();

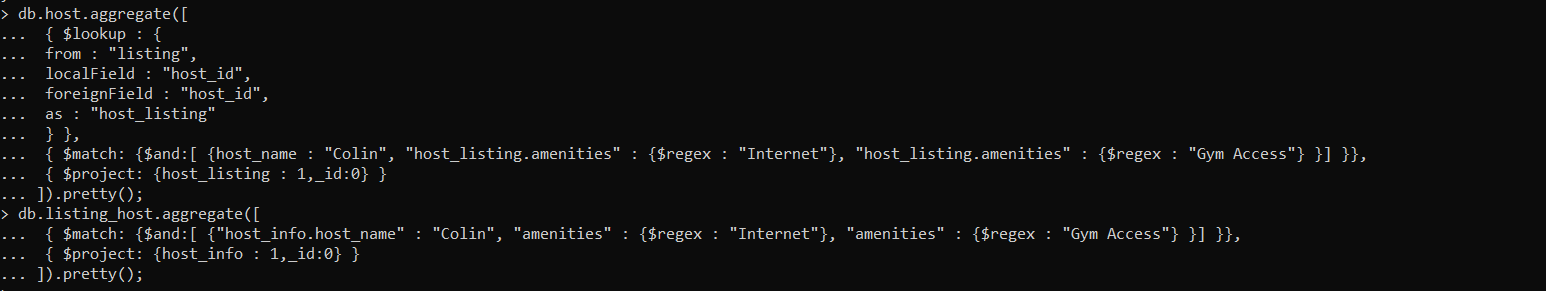
**Embedded**

db.listing\_host.aggregate([

{ $match: {$and:[ {"host\_info.host\_name" : "Colin", "amenities" : {$regex : "Internet"}, "amenities" : {$regex : "Gym Access"} }] }},

{ $project: {host\_info : 1,\_id:0} }

]).pretty();



**Q10**

**Reference**

db.listing.aggregate([

{ $match: {$and:[ {street : {$regex : "Clayton"}, "name" : {$regex : "Beautiful"}}] }},

{ $project: {price : 1,room\_type:1,\_id:0} }

]).pretty();

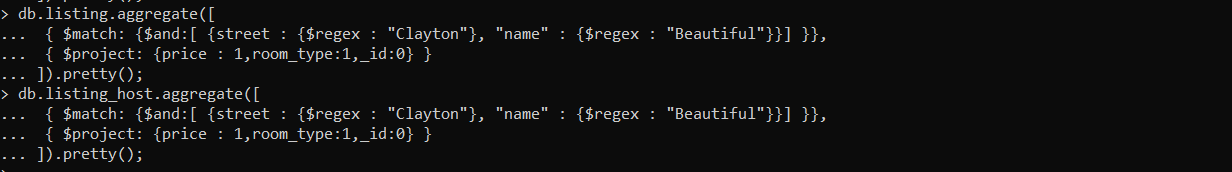
**Embedded**

db.listing\_host.aggregate([

{ $match: {$and:[ {street : {$regex : "Clayton"}, "name" : {$regex : "Beautiful"}}] }},

{ $project: {price : 1,room\_type:1,\_id:0} }

]).pretty();



**Q11**

**Reference**

db.listing.aggregate([

{ $project:

{

\_id:0,

listingDetails: { $concat: ["$name", ", ", { $rtrim: { input: "$street", chars: ", Australia"} }, ", ", {$substr: ["$zipcode", 0, -1]}, ", ", "$neighbourhood"] } } },

{ $sort: { "listingDetails": 1 } }

]).pretty();



**Embedded**

db.listing\_host.aggregate([

{ $project:

{

\_id:0,

listingDetails: { $concat: ["$name", ", ", { $rtrim: { input: "$street", chars: ", Australia"} }, ", ", {$substr: ["$zipcode", 0, -1]}, ", ", "$neighbourhood"] } } },

{ $sort: { "listingDetails": 1 } }

]).pretty();



**Q12**

**Reference**

db.listing.aggregate(

[

{

$sort:

{

"availability\_365" : -1

}

},

{$limit : 1},

{$project: {name:1, \_id:0,"reportGenerationTime": { $add: [new Date()] }

}}

]

).pretty();

**Embedded**

db.listing\_host.aggregate(

[

{

$sort:

{

"availability\_365" : -1

}

},

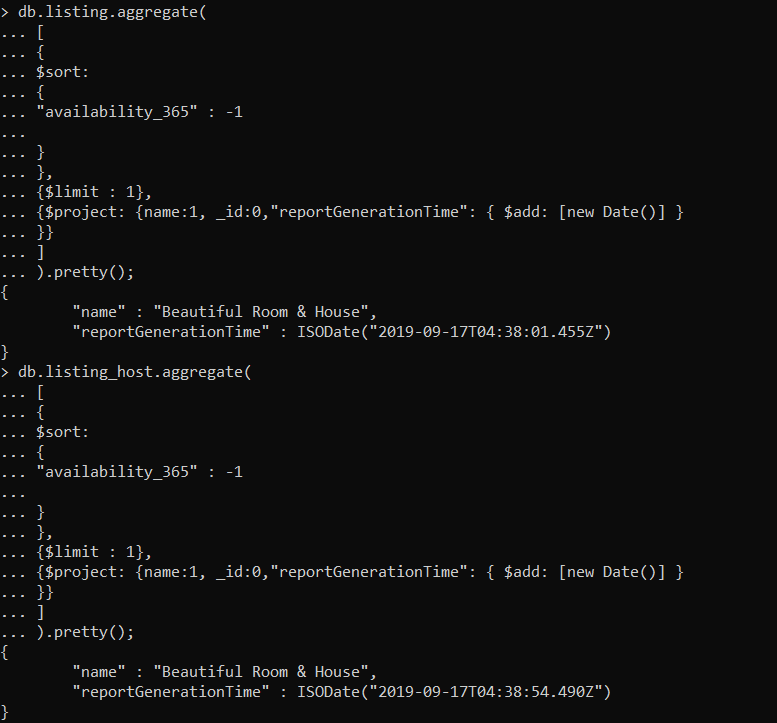
{$limit : 1},

{$project: {name:1, \_id:0,"reportGenerationTime": { $add: [new Date()] }

}}

]

).pretty();



**Q13**

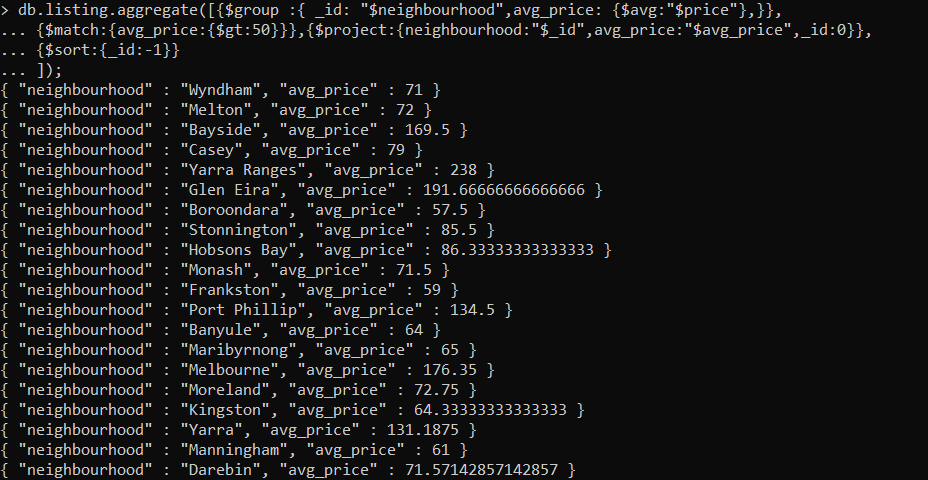
**Reference**

db.listing.aggregate([{$group :{ \_id: "$neighbourhood",avg\_price: {$avg:"$price"},}},

{$match:{avg\_price:{$gt:50}}},{$project:{neighbourhood:"$\_id",avg\_price:"$avg\_price",\_id:0}},

{$sort:{\_id:-1}}

]);



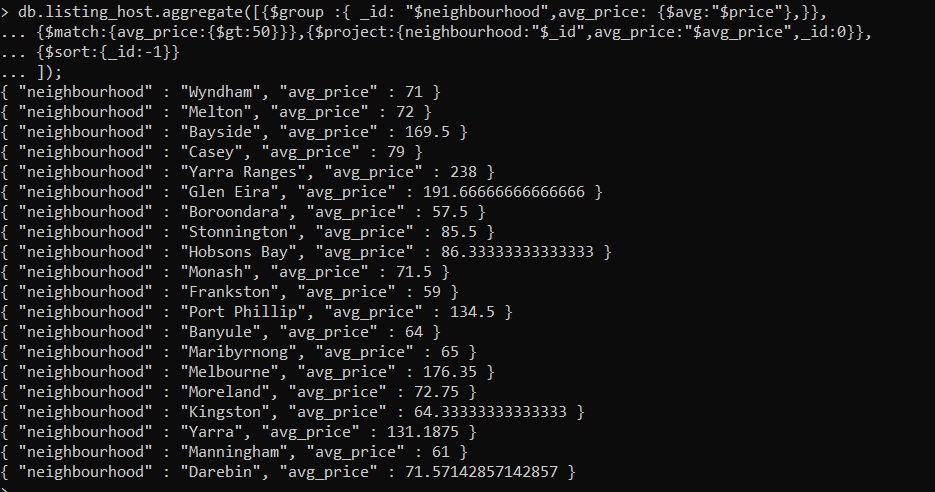
**Embedded**

db.listing\_host.aggregate([{$group :{ \_id: "$neighbourhood",avg\_price: {$avg:"$price"},}},

{$match:{avg\_price:{$gt:50}}},{$project:{neighbourhood:"$\_id",avg\_price:"$avg\_price",\_id:0}},

{$sort:{\_id:-1}}

]);



**Q14**

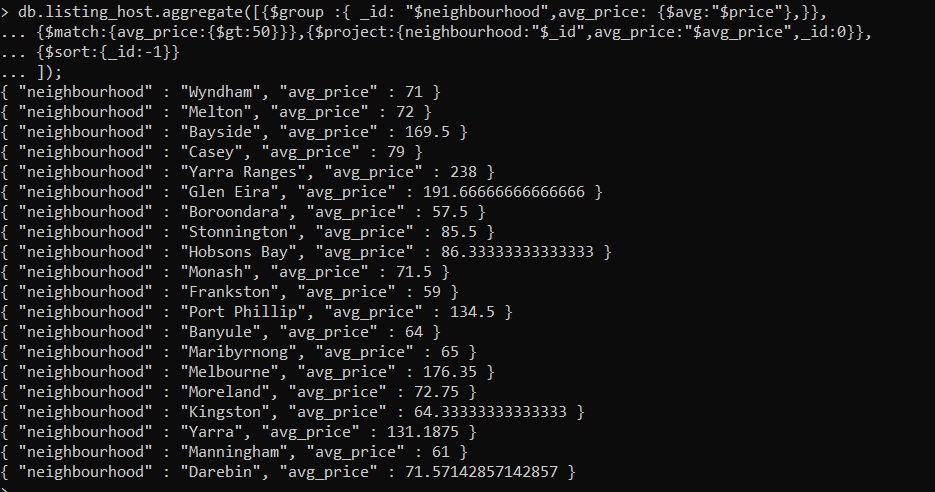
**Reference**

db.host.aggregate([

{$project :{\_id : 0, host\_id: 1, host\_name: 1,"number of verification methods" : { $size: "$host\_verifications" }}},

{$sort:{"number of verification methods":-1}}

]);



**Embedded**

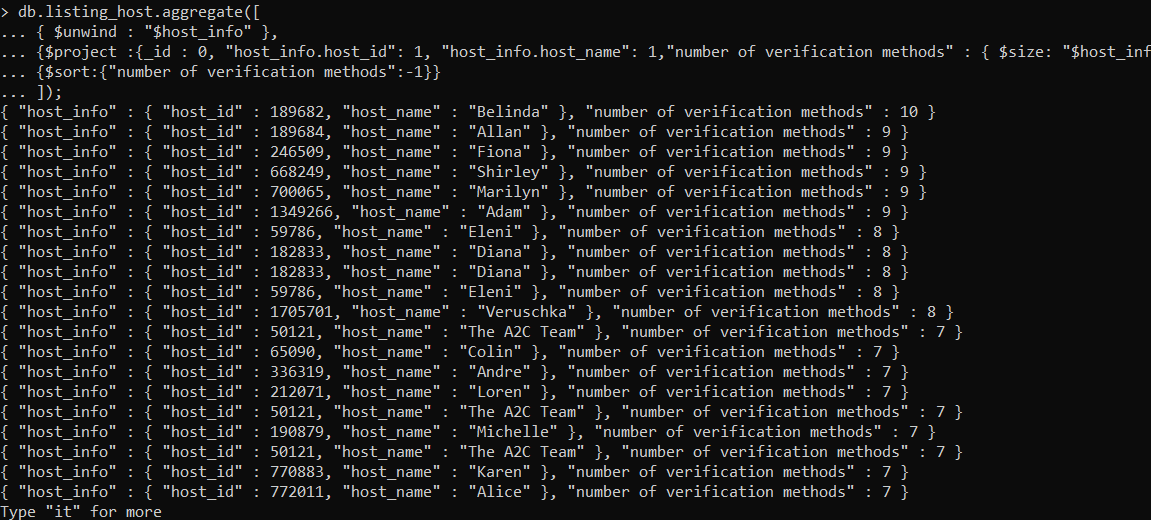
db.listing\_host.aggregate([

{ $unwind : "$host\_info" },

{$project :{\_id : 0, "host\_info.host\_id": 1, "host\_info.host\_name": 1,"number of verification methods" : { $size: "$host\_info.host\_verifications" }}},

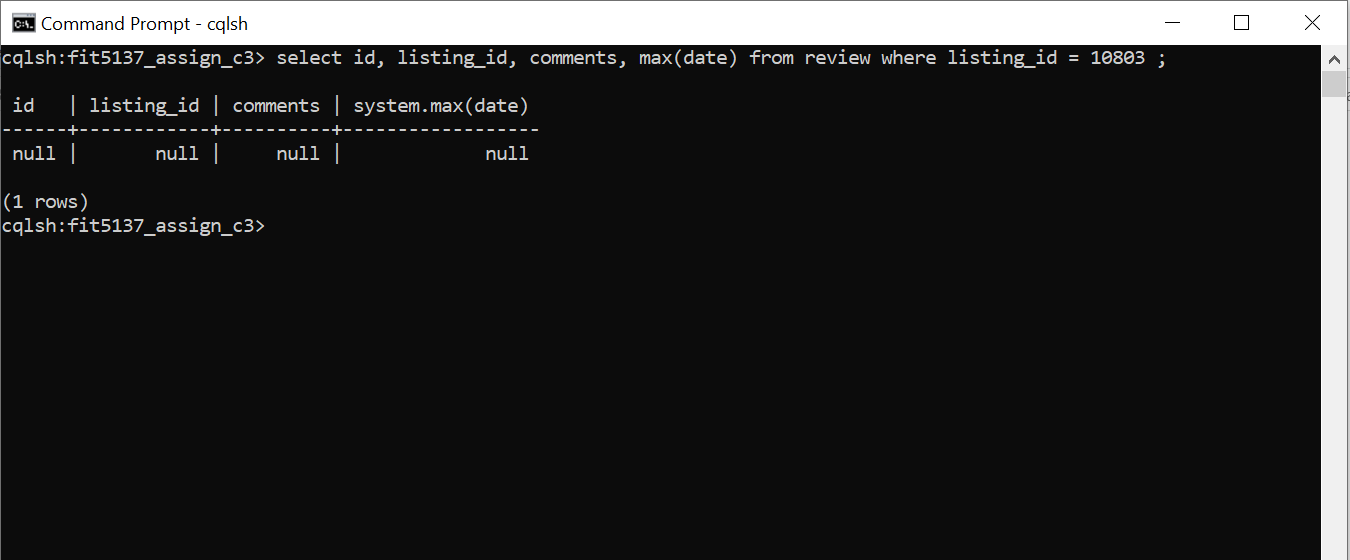
{$sort:{"number of verification methods":-1}}

]);



**Q15**

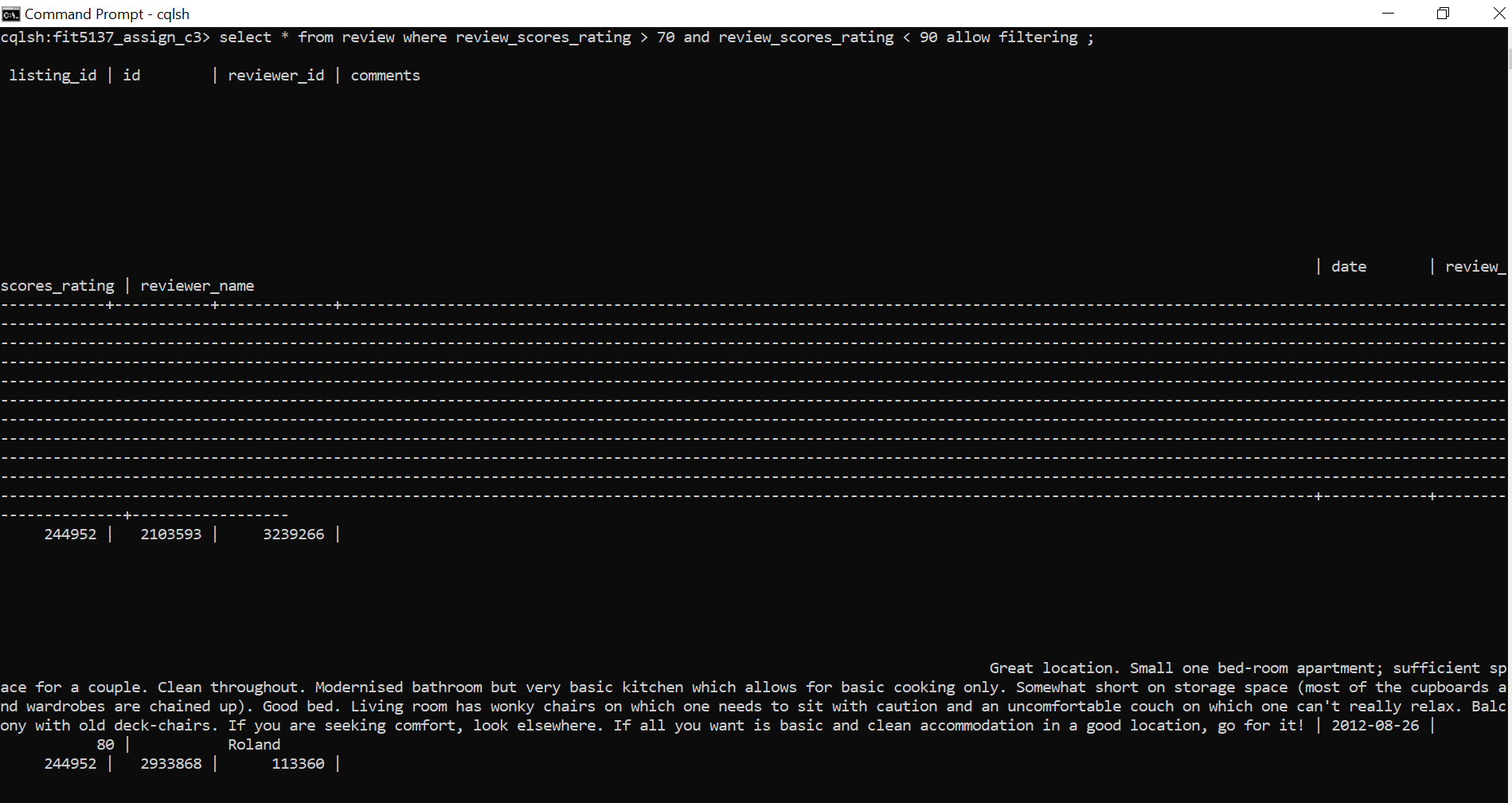
select id, listing\_id, comments, max(date) from review where listing\_id = 10803 ;



**//Windows OS**

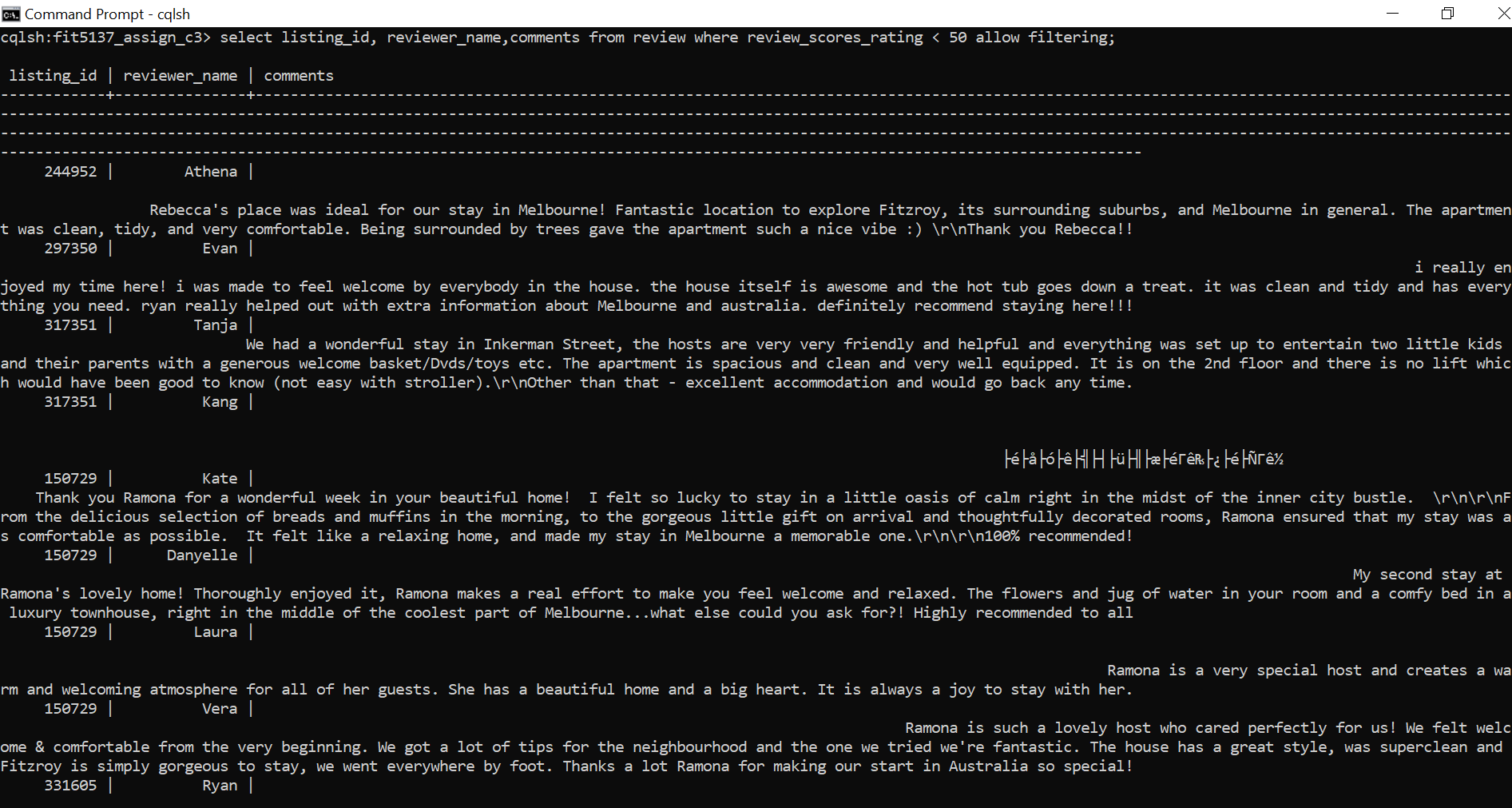
**Q16**

select \* from review where review\_scores\_rating > 70 and review\_scores\_rating < 90 allow filtering ;



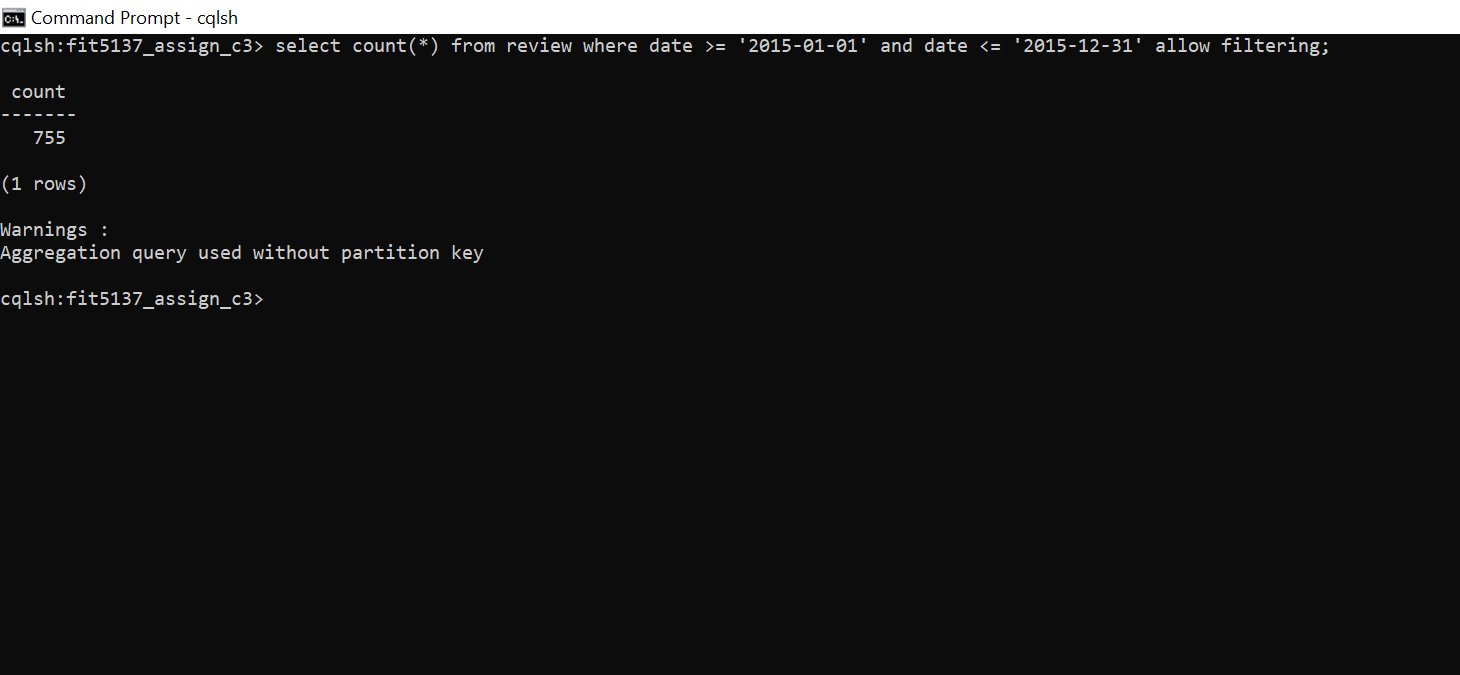
**Q17**

select listing\_id, reviewer\_name,comments from review where review\_scores\_rating < 50 allow filtering;



**Q18**

select count(\*) from review where date >= '2015-01-01' and date <= '2015-12-31' allow filtering;



**Q19**

create index on review(date);

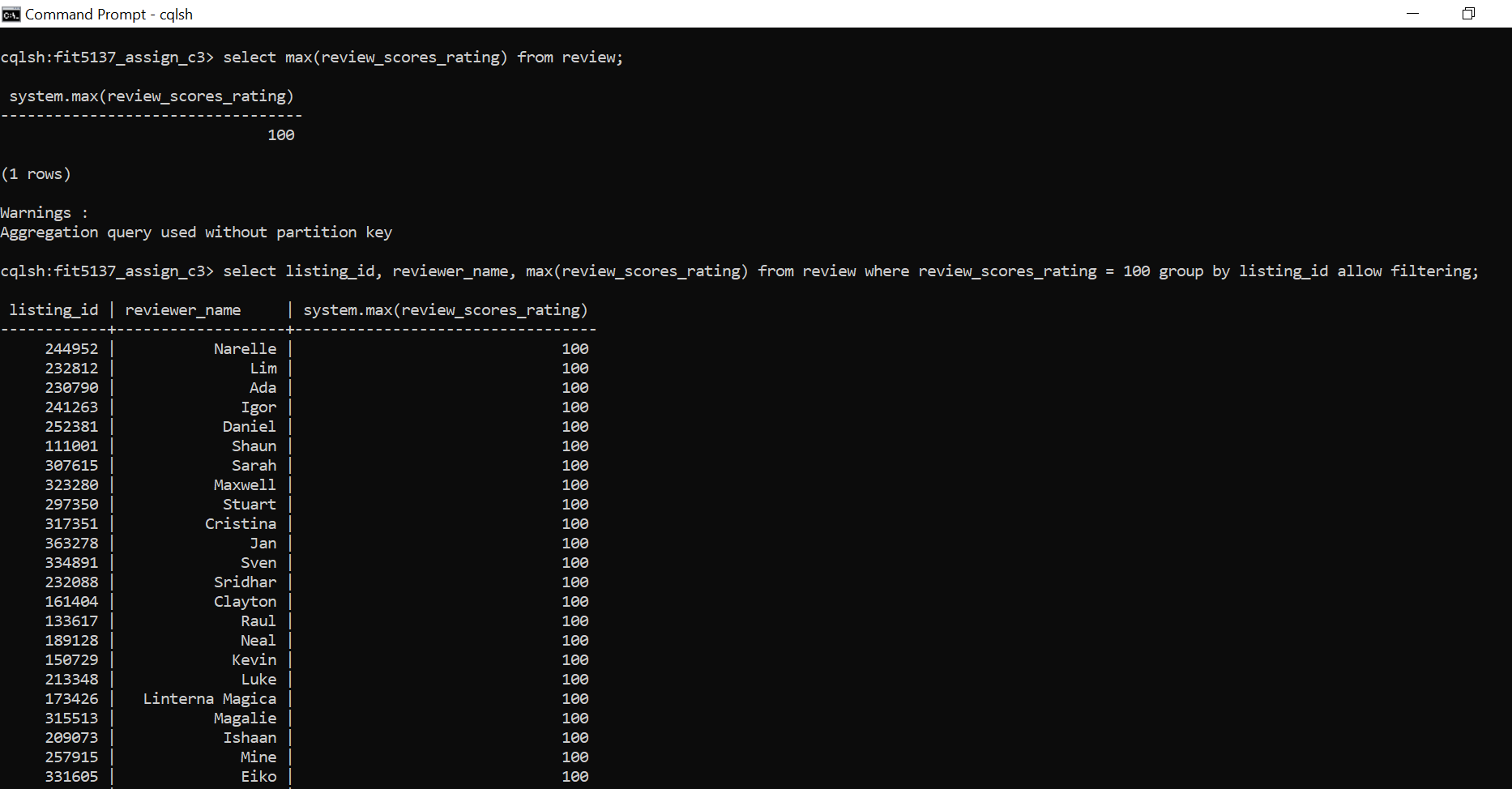
select id, reviewer\_name,comments, max(review\_scores\_rating) from review where date = '2017-03-26';



**Q20**

select max(review\_scores\_rating) from review;

select listing\_id, reviewer\_name, max(review\_scores\_rating) from review where review\_scores\_rating = 100 group by listing\_id allow filtering;



**--EXTRA QUERIES--**

**1.Display the cheapest neighbourhoods based on average price.**

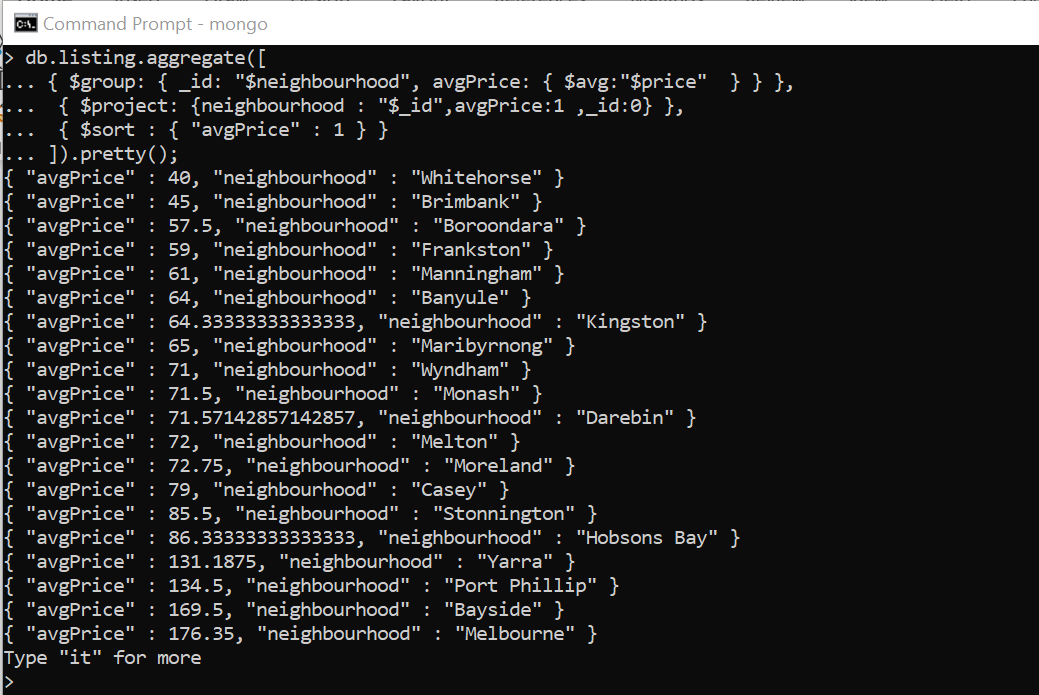
db.listing.aggregate([

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

{ $project: {neighbourhood : "$\_id",avgPrice:1 ,\_id:0} },

{ $sort : { "avgPrice" : 1 } }

]).pretty();



**2.Display the average number of reviews for a host.**

db.listing\_host.aggregate([

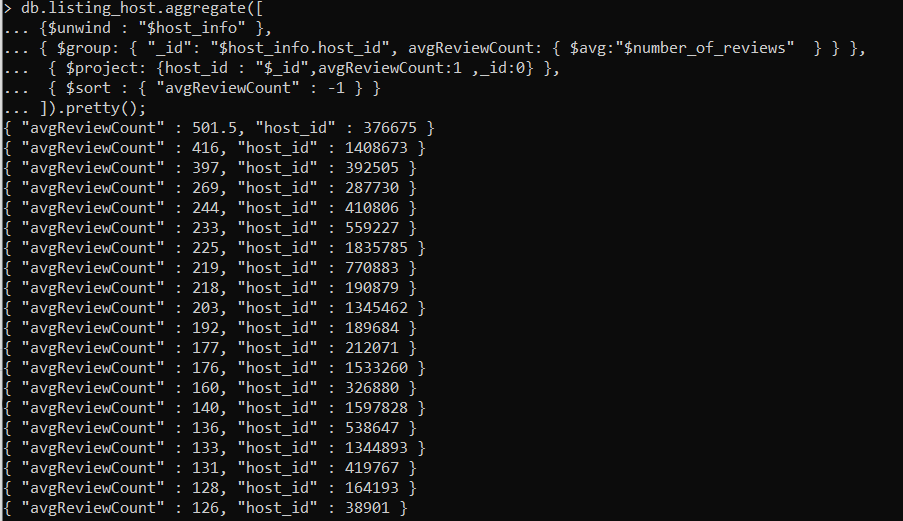
{$unwind : "$host\_info" },

{ $group: { "\_id": "$host\_info.host\_id", avgReviewCount: { $avg:"$number\_of\_reviews" } } },

{ $project: {host\_id : "$\_id",avgReviewCount:1 ,\_id:0} },

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

]).pretty();



**3.Display the most popular listings (based on availability\_365).**

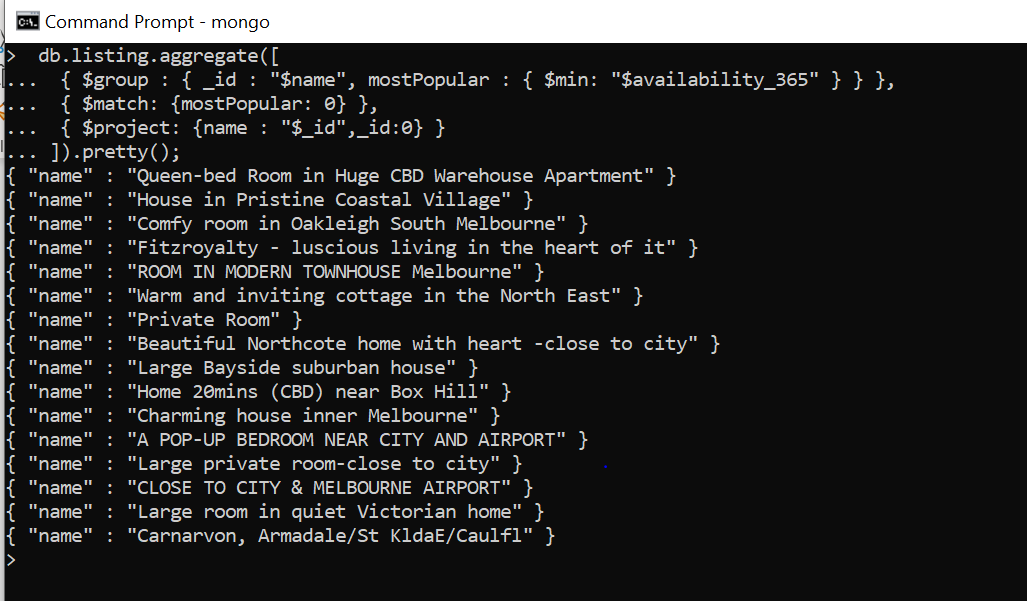
db.listing.aggregate([

{ $group : { \_id : "$name", mostPopular : { $min: "$availability\_365" } } },

{ $match: {mostPopular: 0} },

{ $project: {name : "$\_id",\_id:0} }

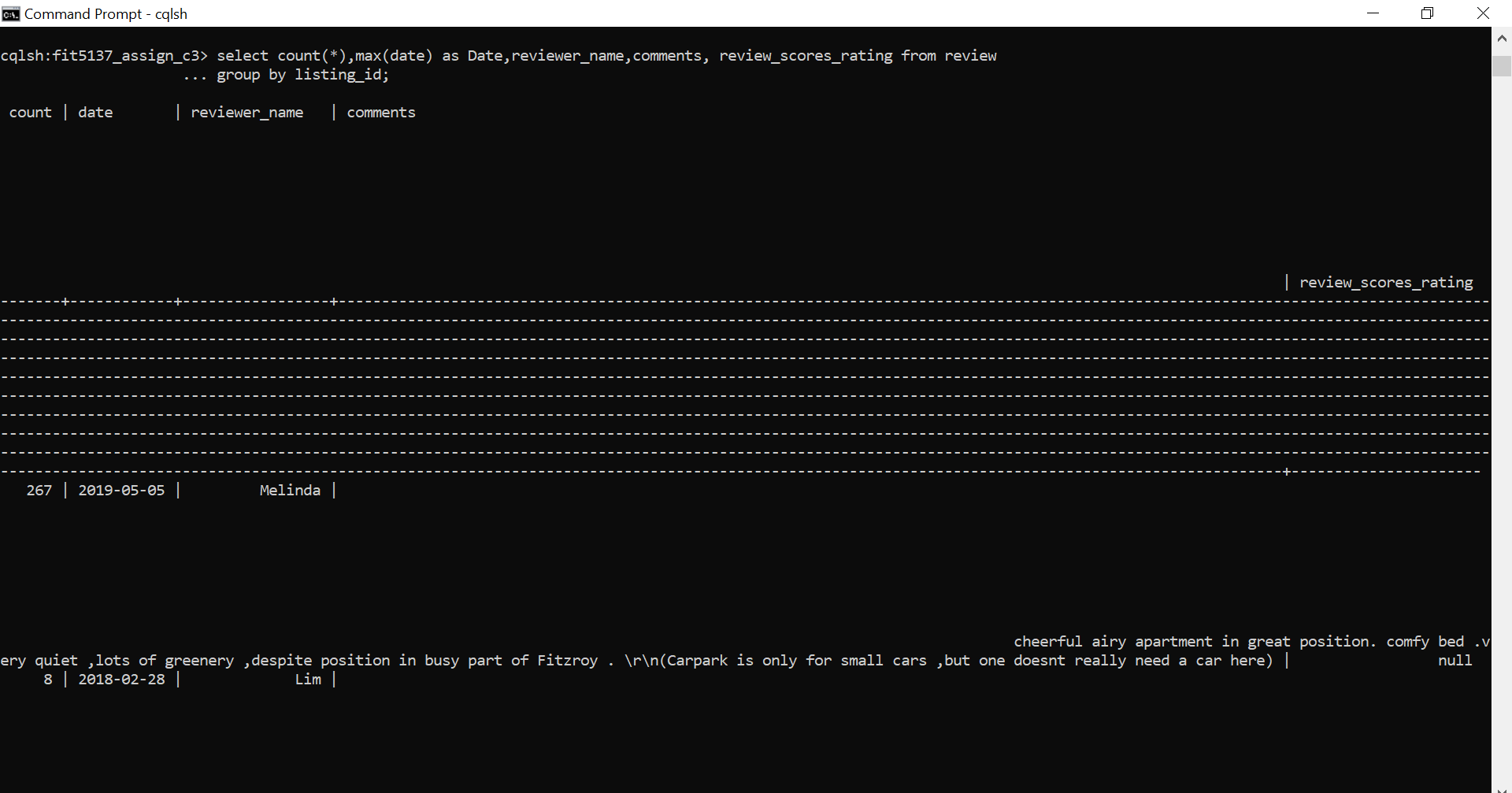
]).pretty();



**4. Display each listing with the count of total reviews, the most recent date of the review, reviewer name and its rating.**

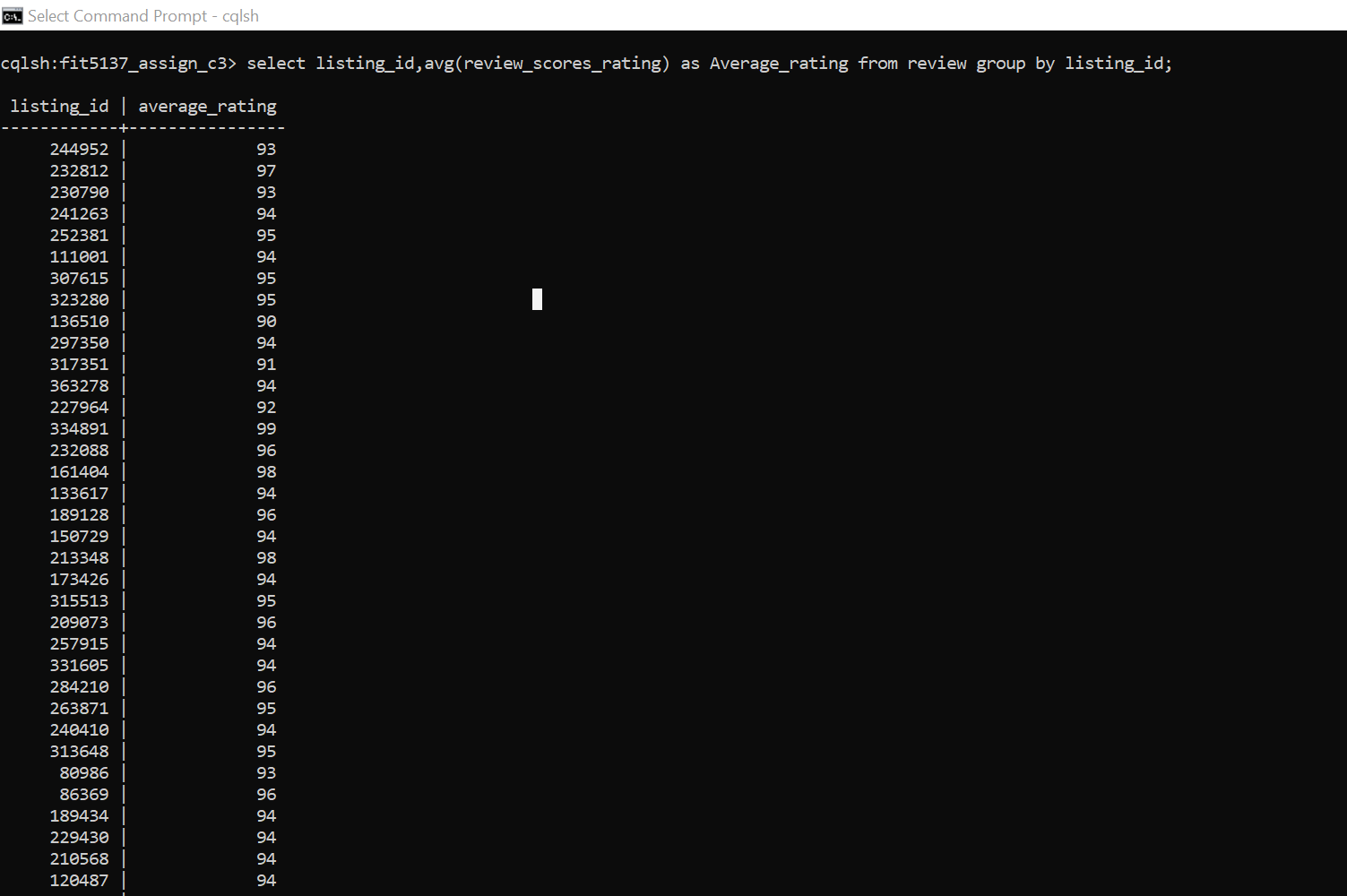
select count(\*),max(date) as Date,reviewer\_name,comments, review\_scores\_rating from review

group by listing\_id;



**5. Display the average rating of the each listing.**

select listing\_id,avg(review\_scores\_rating) as Average\_rating from review group by listing\_id;



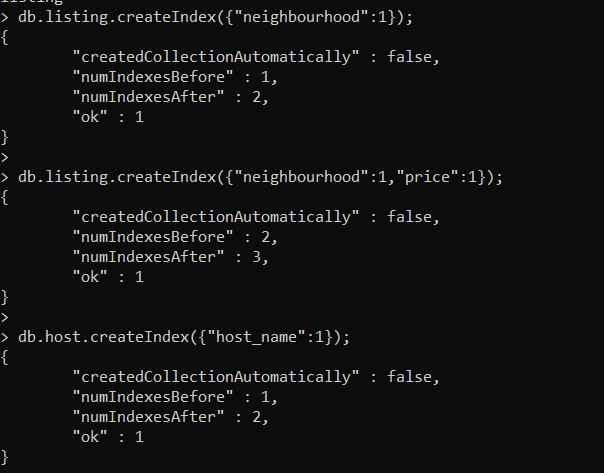
**--Indexing--**

**Create index for referencing model**

db.listing.createIndex({"neighbourhood":1});

db.listing.createIndex({"neighbourhood":1,"price":1});

db.host.createIndex({"host\_name":1});

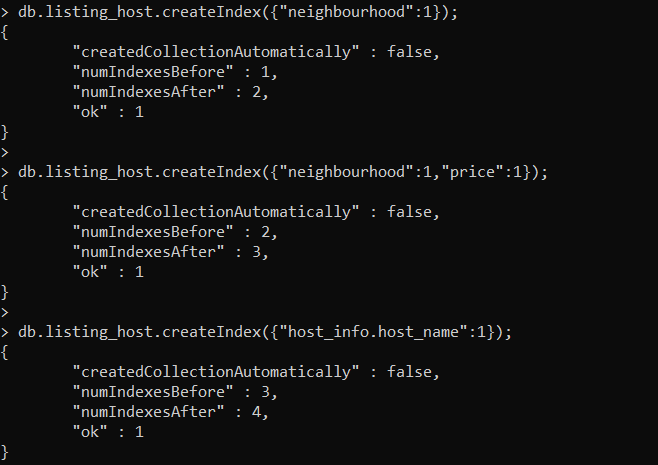


**Create index for embedding model**

db.listing\_host.createIndex({"neighbourhood":1});

db.listing\_host.createIndex({"neighbourhood":1,"price":1});

db.listing\_host.createIndex({"host\_info.host\_name":1});



The columns considered for indexing are neighbourhood, price and host\_name. We believe that according to the business case specified and the queries implemented, most frequently used columns will be regarding the neighbourhood or location, host of the accommodation or listings and the price specified for it. Since embedded model still has the same columns as the referencing model, the indexing keys still remain the same.

**--C.4. Database Comparison. REPORT--**

● Decide which database to use (either MongoDB or Cassandra).

We have decided to go with Cassandra. The reasons for the same are mentioned further in this section.

● List of the steps (including an explanation and a flowchart) on how the merging process could be implemented.

There are multiple ways online for implementing this task. Using the write and read branching techniques, using individual insert query for JSON objects (supported by Cassandra Version 3.0) or java looping on insert and many more techniques.

The easiest and most efficient solution discovered and implemented as of now is by exporting the collection of MongoDB to the csv file and importing in Cassandra (MongoDB Documentation, 2019). Cassandra provides only limited support for JSON. The steps for the same are mentioned in the form of a flowchart and explained as follows.

**Step 1:**

We already have JSON files for the collections *Listing* and *Host.* If we assume that the JSON files are not yet ready, we need to export the collections from MongoDB to JSON or direct CSV format using “mongoexport” command-line tool which is to be run directly from the system command line.

mongoexport --host --collection collection\_name --db keyspace --out filename.csv [additional options]

use --type=csv to export the file in csv format.

Note: If Step 1 doesn’t work, export the files into JSON format and then convert into CSV using JSON to CSV convertor.

**Step 2:**

Run the Cassandra Server by giving the command “Cassandra” in cmd.

**Step 3:**

Open a different cmd and drop to cql shell by giving the command “cqlsh”.

**Step 4:**

Select the keyspace which already has the table(column family) “Review” in Cassandra by the command “use keyspace\_name”.

**Step 5:**

Create table *Listing* and *Host* with proper data types of the columns and primary keys.

**Step 6:**

Import both the CSV files using the command

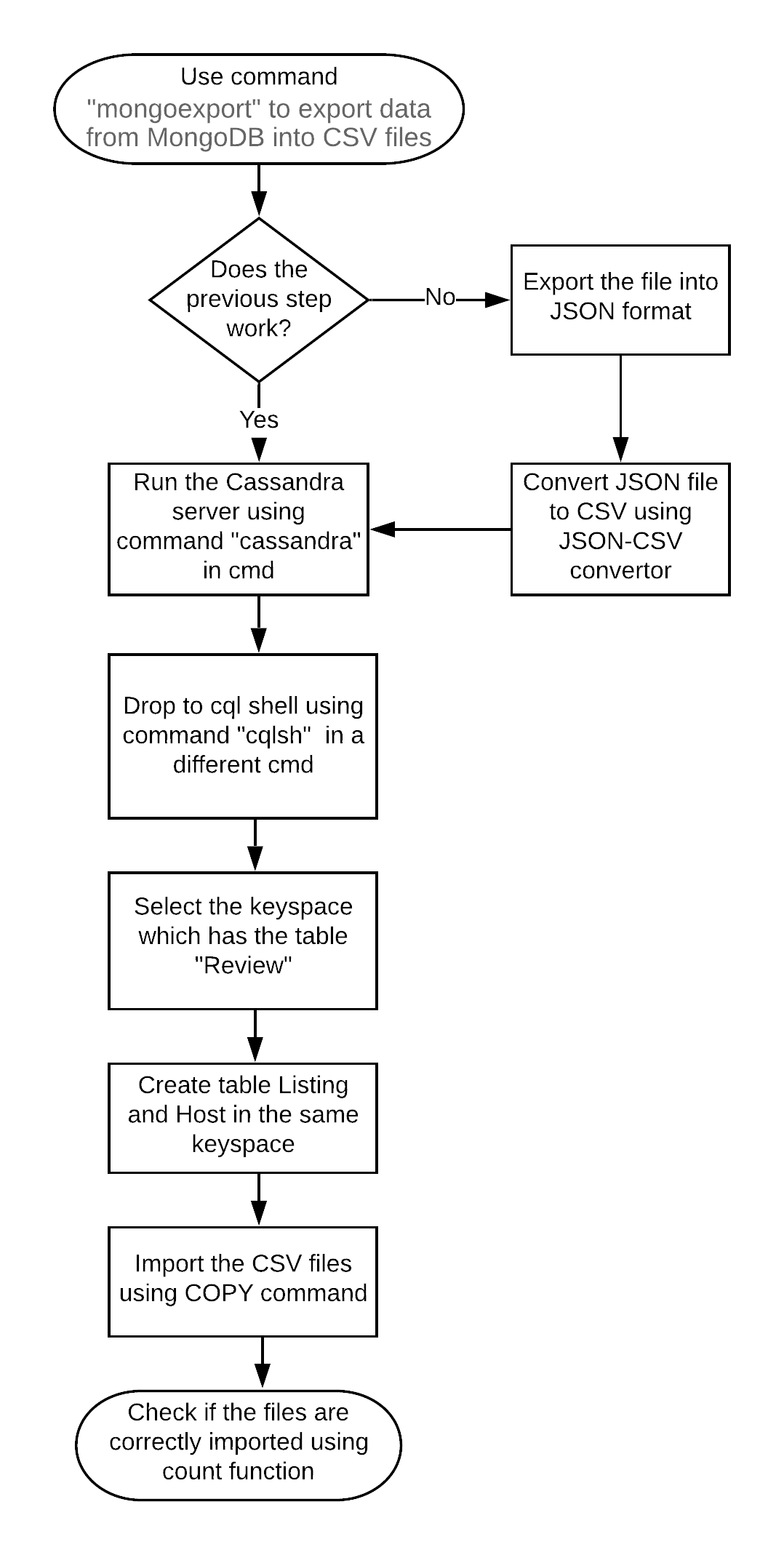
“COPY keypsace.tableName (col1,col2,col3.....) FROM 'file/file.csv' WITH DELIMITER=',' AND HEADER=TRUE;”

**Step 7:**

Check if the files are imported correctly by

“Select count(\*) from keyspace.table;”

* **Flow Chart**



● Comparison in a tabular format with details on the main strengths and weaknesses of each database (Cassandra Vs MongoDB In 2018(Panoply blog, 2018)).

|  |  |  |  |
| --- | --- | --- | --- |
| Category | NO | MongoDB | Cassandra |
| Syntax | 1 | MongoDB requires to learn a new language altogether. | CQL is easier than MongoDB if you already know SQL. |
| Replication | 2 | Replication is built-in, but requires additional setup | Replication is built-in and easy to use. |
| Data Storage | 3 | Requires JSON format to retrieve data and data is stored as documents in BSON | Flexible wide-column. i.e. Tables called as column families. Non-relational way |
| Analysis | 4 | Limits the number of users who can make changes to the database. This is because, the user access rights given by the administrator can either be full access or read-only. | The access rights for the users are defined per object by the database administrator. Hence, any user can get respective access from the DBA. |
|  | 5 | MongoDB does support number of programming languages. eg. LISP, Matlab, Powershell etc. | Does not support much programming languages as compared to MongoDB |
|  | 6 | Good for data without clear schema definition. | It can handle large amount of unstructured data. |
|  | 7 | MongoDB can be a great choice if you need scalability and caching for real-time analytics | Can be speedily scaled with minimal increase of DBA work while having high reliability |
|  | 8 | Good where you need high insert rate i.e. where write load is high. | Easy to setup and maintain if you expect rapid growth in database. |
| Use Case | 9 | Real-time analytics, mobile, IOT | Messaging system, Real-time analytics, Fraud detection, Storage |

● Explanation on why we have selected Cassandra database over MongoDB.

The case study suggests few requirements which are efficiently satisfied by the specifications and properties of Cassandra. The requirements according to their respective justification is briefly explained as given (MongoDB vs Cassandra - The Next Generation NOSQL Enterprise Database Environments, 2016).

1. MonashBnB wishes to design a new database due to the increase in the volume of people occupying the accommodations listed. Cassandra supports large amount of unstructured data and can be rapidly scaled up in case of large volume efficiently.
2. The management team wishes to have a control of the environment without any manual hassles and maintenance. Also, they are migrating the data from manual entries to a database. Cassandra suffices the requirement as it can handle unstructured data and store it in a non-relational way. Also, it is easy to maintain without any much DBA work.
3. The CQL is easy to learn for a layman as compared to that of MongoDB. As the management team is not as experienced, Cassandra suffices the requirement.
4. The access rights for the users can be defined by the DBA. As any member from the team can easily access for the rights.

* References

MongoDB Documentation. (2019). Retrieved from

https://docs.mongodb.com/manual/reference/program/mongoexport/#bin.mongoexport

MongoDB vs Cassandra - The Next Generation NOSQL Enterprise Database Environments. (2016). Plus Company Updates.

Panoply blog.(2018).Cassandra Vs MongoDB In 2018. Retrieved from <https://blog.panoply.io/cassandra-vs-mongodb>