

|  |
| --- |
|  |
| Related image  PUB LEASE CONTRACTS  Advanced Databases  Project Report  By Yash Sinojia [A00268852] |
| MASTER OF SCIENCE IN DATA ANALYTICS  2019 - 2020  FACULTY OF BUSINESS AND HOSPITALITY  SEMESTER 2 |

CONTENTS

[1. INTRODUCTION 2](#_Toc39818732)

[2. ORIGINAL PROJECT OUTLINE 3](#_Toc39818733)

[3. DATABASE 4](#_Toc39818734)

[4. QUERIES 4](#_Toc39818735)

[4.1. C.R.U.D. Operations 4](#_Toc39818736)

[4.1.1. Create 4](#_Toc39818737)

[4.1.2. Retrieve 10](#_Toc39818738)

[4.1.3. Update 15](#_Toc39818739)

[4.1.4. Delete 24](#_Toc39818740)

[4.1.5. Bulk Write 26](#_Toc39818741)

[4.2. Aggregation Framework 29](#_Toc39818742)

[4.2.1. Distinct 29](#_Toc39818743)

[4.2.2. $match, $group, $count, $sort 30](#_Toc39818744)

[4.2.3. $project, $out, $unwind 37](#_Toc39818745)

[4.2.4. $sample, $collStats 48](#_Toc39818746)

[4.3. MongoDB Compass 51](#_Toc39818747)

[5. REFERENCES 57](#_Toc39818748)

# INTRODUCTION

MongoDB is a cross-platform, document-oriented NoSQL database query language and much more. Documents are stored as JSON or BSON type. This project utilizes the opportunity to explore MongoDB in depth to demonstrate the understanding of its main components/features. The database and collection to be used in this project is already created and populated with 1000 documents.

The project scenario is about Pub Leasing Contracts (PLC). The dataset is populated on a .csv file, which is then copied to a .json file, which then is imported on Mongo Shell with mongoimport.

This report focuses on exploring the features of the PLC database by writing a number of queries that either manipulates the data or generates an output or both. The queries are divided into three main categories: 1) CRUD Operations 2) Aggregation Framework and 3) MongoDB Compass. In this report, for every query a question is framed around the possible outcomes from all the prior-knowledge in regards to a query. This question is resolved by the resulting output generated by the query. These set of question-answers serves as a proof to the conceptual understanding of MongoDB and the contrasts and similarities with relational databases such as Oracle SQL and PL/SQL.

Here, MongoDB Enterprise Edition 4.2 with MongoDB Compass has been utilized for every query operation performed.

# ORIGINAL PROJECT OUTLINE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 21 | Yash | Sinojia | Pub Lease Contracts | PLC |

**Pub** *(*\_Id, Name, Address, Town, County, Proprietor\_Name, Size\_Square\_Foot, Year\_Opened, Bar\_Food (*Yes/No*), Reputation *(Fair, Good, Excellent*), Annual\_Turnover)

Array – Food (Snacks, Sandwiches, Tea, Coffee, Hot Lunches, Salads etc), i.e a pub may offer one or more of the foods listed above

Array – Bookings (Corporate Parties, Birthdays, Weddings, Private Parties, Table Quizzes etc),

Embedded Documents – Assume that a pub has a number of sources of income from the range of activities it offers – therefore an embedded document such as Income.Alcohol, Income.Food, Income.Parties etc might be applicable

Embedded Documents – Assume that a pub has a number of main expenses of that it needs to manage – therefore an embedded document such as Expense.Wages, Expense.Insurance, Expense.Rent etc might be applicable

# DATABASE

Database Name: PLC\_Project

Collection Name: advanced\_db\_plc

Number of Instances: 1000

Snip:



# QUERIES

## **4.1. C.R.U.D. Operations**

### 4.1.1. Create

Question.1

A new bar called ‘Chruch Bar’ in Kildare needs to be registered, enter details in database.

Solution

db.advanced\_db\_plc.insertOne(

{

Name: "Chruch Bar",

Address: "H10, Maynooth Business Campus",

Town: "Maynooth",

County: "Kildare",

Proprietor\_Name: "Aine Daly",

Size\_Square\_Foot: 13500,

Year\_Opened: 2010,

Bar\_Food: "Yes",

Reputation: "Good",

Annual\_Turnover: 395500,

Food: ["Tea Coffee", "Sandwiches", "Snacks"],

Bookings: ["Private Parties", "Birthdays", "Table Quizzes"],

Income:

{

Alcohol: 75000,

Food: 25000,

Parties: 1000000

},

Expense:

{

Wages: 400000,

Insurance: 150000,

Rent: 304500

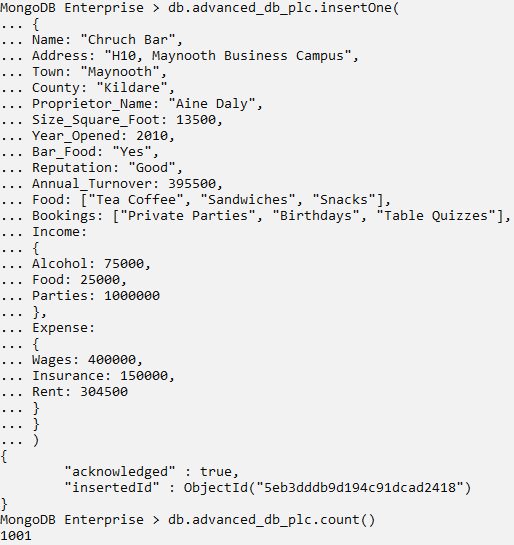
}

}

)

db.advanced\_db\_plc.count()

Snip



Question.2

Two new bars have just opened up in Birr. Enter the details in the database with a single query.

Solution

db.advanced\_db\_plc.insertMany(

[

{

Name: "Heavy Pub",

Address: "23, Sunshine Valley",

Town: "Birr",

County: "Westmeath",

Proprietor\_Name: "James Hatfield",

Size\_Square\_Foot: 13500,

Year\_Opened: 2020,

Bar\_Food: "Yes",

Reputation: "Good",

Annual\_Turnover: 395500,

Food: ["Tea Coffee", "Sandwiches", "Snacks"],

Bookings: ["Private Parties", "Birthdays", "Table Quizzes"],

Income:

{

Alcohol: 75000,

Food: 25000,

Parties: 1000000

},

Expense:

{

Wages: 400000,

Insurance: 150000,

Rent: 304500

}

},

{

Name: "Commune Bar",

Address: "22, Cartrontroy",

Town: "Birr",

County: "Westmeath",

Proprietor\_Name: "Sorca Clarke",

Size\_Square\_Foot: 22500,

Year\_Opened: 2020,

Bar\_Food: "Yes",

Reputation: "Excellent",

Annual\_Turnover: 955000,

Food: ["Tea Coffee", "Hot Lunches", "Salads"],

Bookings: ["Birthdays", "Table Quizzes", "Corporate Parties", "Weddings"],

Income:

{

Alcohol: 175000,

Food: 55000,

Parties: 2000000

},

Expense:

{

Wages: 750000,

Insurance: 100000,

Rent: 425000

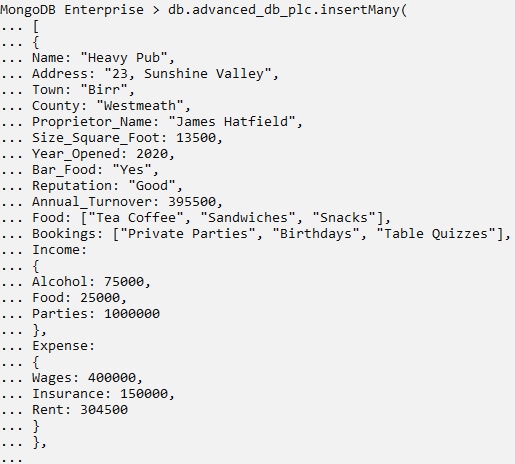
}

},

])

db.advanced\_db\_plc.count()

Snip





### 4.1.2. Retrieve

Question.3

Display all the pubs that have been opened in Ireland between 2000 and 2010.

Solution

db.advanced\_db\_plc.find(

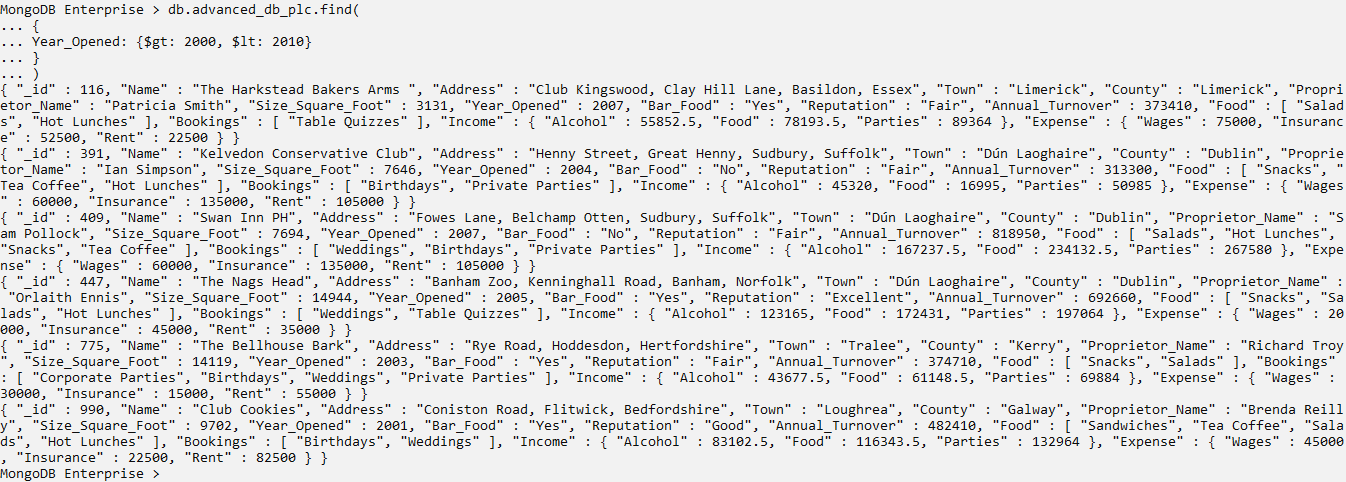
{

Year\_Opened: {$gt: 2000, $lt: 2010}

}

)

Snip



Explanation:

$gt and $lt are comparison operators.

Question.4

Show all the pubs in Ireland that can host Private and Corporate Parties.

Solution

db.advanced\_db\_plc.find(

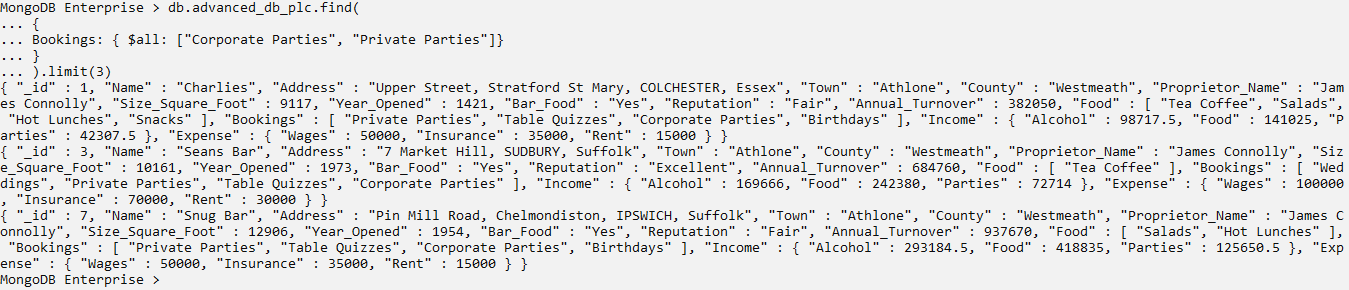
{

Bookings: {$all: ["Corporate Parties", "Private Parties"]}

}

).limit(3)

Snip



Explanation:

$all covers all the combinations of elements ‘Corporate Parties’ and ‘Private Parties’ in every order of array elements.

Question.5

Display all the pubs who earn more than 300k just from Alcohol sales. Show the name of Pub, Proprietor Name, Town and Income from Alcohol only.

Solution

db.advanced\_db\_plc.find(

{

"Income.Alcohol": {$gt: 300000}

},

{

Name: 1,

Proprietor\_Name: 1,

Town: 1,

"Income.Alcohol": 1

}

)

Snip





Explanation:

The first stage of find works as a filter and the second stage specifies which attributes are to be projected on results (*db.collection.find() — MongoDB Manual*, no date).

Question.6

The documents with Pubs owned by Barry Sands with size greater than 6k sq. ft. and reputation excellent needs to be sorted by size and indexed separately.

Solution

db.advanced\_db\_plc.findAndModify(

{

query: {Proprietor\_Name: "Barry Sands", Size\_Square\_Foot: {$gt: 6000}, Reputation: "Excellent"},

sort: {Size\_Square\_Foot: 1},

update: {$inc: {index: 1}},

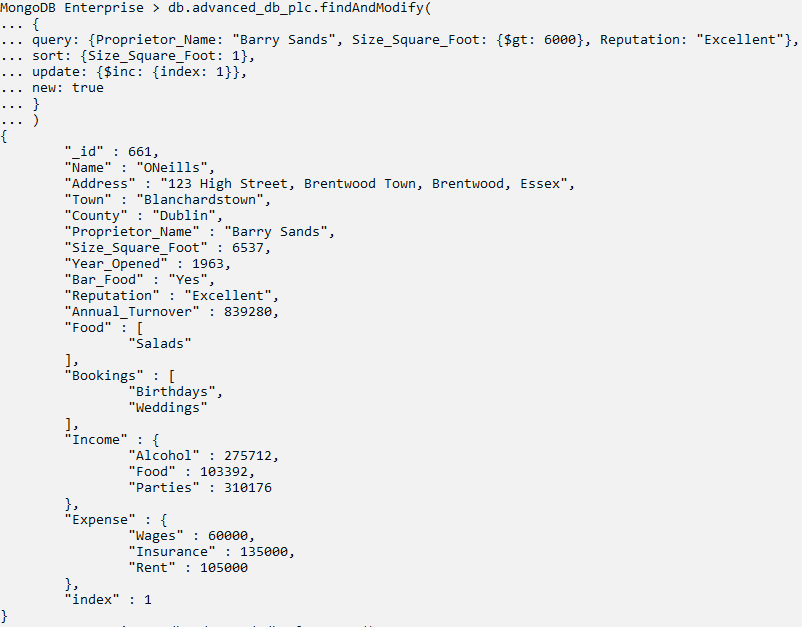
new: true

}

)

db.advanced\_db\_plc.count()

Snip



Explanation:

‘new’ argument gathers the output as a separate document. ‘update’ arguments adds index to the results (*db.collection.findAndModify() — MongoDB Manual*, no date).

### 4.1.3. Update

Question.7

In 1565, Irish pubs only served alcohol and no bar food. There is an entry in the database that says that a pub opened on 1565 serves food, correct that.

Solution

db.advanced\_db\_plc.find(

{

Year\_Opened: 1565

}

).pretty()

db.advanced\_db\_plc.updateOne(

{

Year\_Opened: 1565

},

{

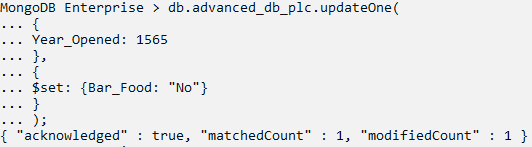
$set: {Bar\_Food: "No"}

}

);

Snip





Question.8

To support the Irish pub businesses and introduce mutually healthy competition, set the Reputation of pubs whose Annual Turnover is less than 202k to “Good”.

Solution

db.advanced\_db\_plc.find(

{

Annual\_Turnover: {$lt: 202000}

}

)

db.advanced\_db\_plc.updateMany(

{

Annual\_Turnover: {$lt: 202000}

},

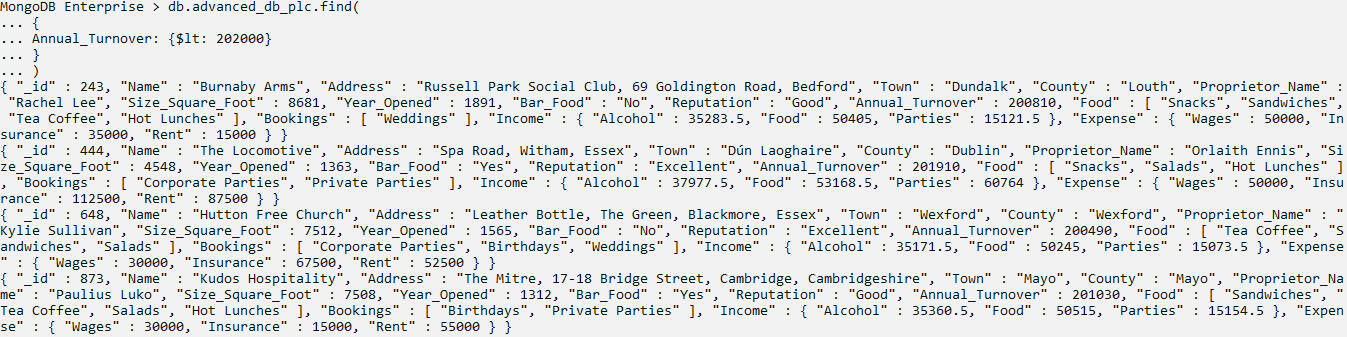
{

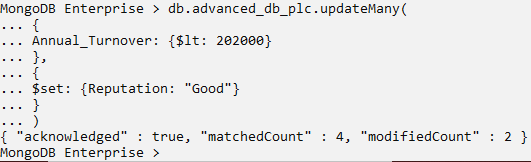
$set: {Reputation: "Good"}

}

)

Snip





Question.9

Kudos Hospitality has now shifted from Mayo to Roscommon and have changed their attributes according to new location, replace the details in database.

Solution

db.advanced\_db\_plc.find(

{

"Name" : "Kudos Hospitality"

}

).pretty()

db.advanced\_db\_plc.replaceOne(

{

"Name" : "Kudos Hospitality"

},

{

"Name" : "Kudos Hospitality",

"Address" : "108, Hudson Bay",

"Town" : "Roscommon",

"County" : "Roscommon",

"Proprietor\_Name" : "Paulius Luko",

"Size\_Square\_Foot" : 7508,

"Year\_Opened" : 1312,

"Bar\_Food" : "Yes",

"Reputation" : "Good",

"Annual\_Turnover" : 201030,

"Food" : [

"Sandwiches",

"Tea Coffee",

"Salads",

"Hot Lunches"

],

"Bookings" : [

"Birthdays",

"Private Parties"

],

"Income" : {

"Alcohol" : 35360.5,

"Food" : 50515,

"Parties" : 15154.5

},

"Expense" : {

"Wages" : 30000,

"Insurance" : 15000,

"Rent" : 55000

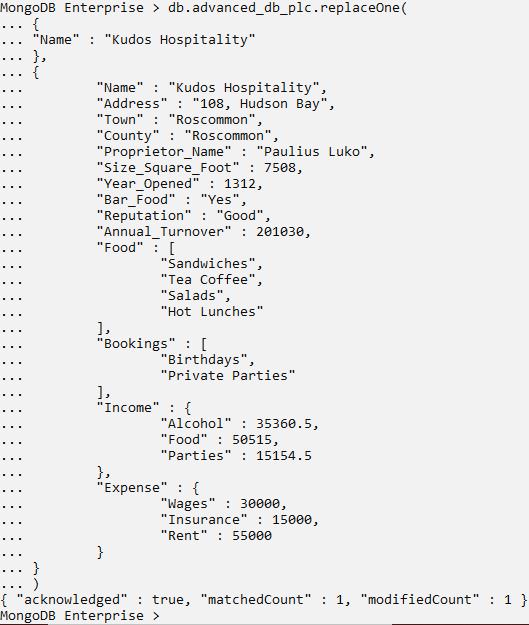
}

}

)

Snip





Question.10

The proprietors of The Star Inn in Dundalk have split, the original proprietor still owns the original pub but the secondary proprietor has started a new pub in Galway, the names have been changed to The Sun Inn. Update the database.

Solution

db.advanced\_db\_plc.find(

{

"Name" : "The Star Inn"

}

).pretty()

db.advanced\_db\_plc.replaceOne(

{

"Name" : "The Star Inn"

},

{

"Name" : "The Sun Inn",

"Address" : "Cliffs of Moher Road",

"Town" : "Galway",

"County" : "Galway",

"Proprietor\_Name" : "Gemini Guin",

"Size\_Square\_Foot" : 5700,

"Year\_Opened" : 2020,

"Bar\_Food" : "Yes",

"Reputation" : "Good",

"Annual\_Turnover" : 235000,

"Food" : [

"Snacks",

"Sandwiches",

"Tea Coffee"

],

"Bookings" : [

"Corporate Parties",

"Private Parties"

],

"Income" : {

"Alcohol" : 58779,

"Food" : 83970,

"Parties" : 25191

},

"Expense" : {

"Wages" : 20000,

"Insurance" : 45000,

"Rent" : 35000

}

},

{

upsert: true

}

)

db.advanced\_db\_plc.find(

{

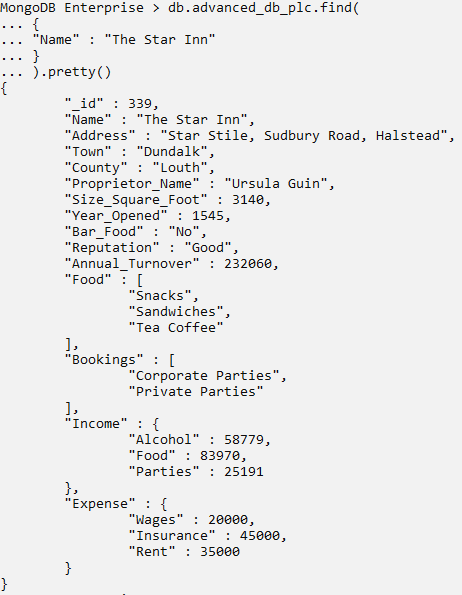
"Name" : "The Sun Inn"

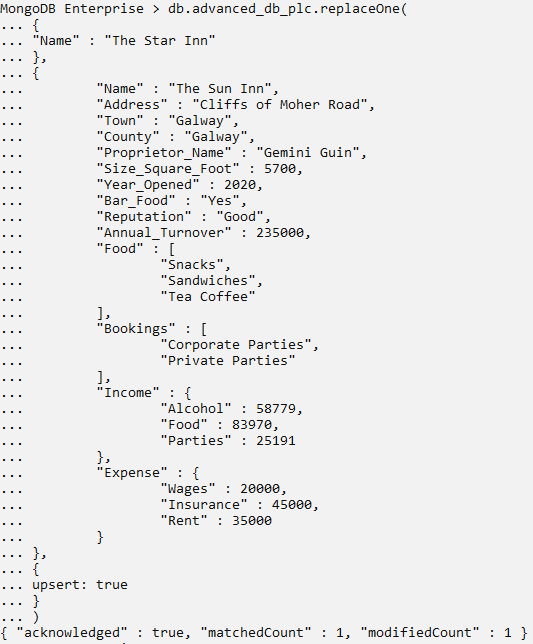
}

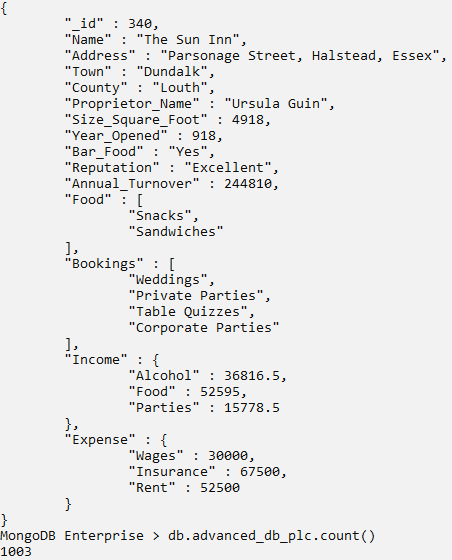
).pretty()

db.advanced\_db\_plc.count()

Snip







Explanation:

The ‘upsert’ argument is mentioned true, which creates a new document if no match is found on first stage, otherwise replaces as default.

### 4.1.4. Delete

Question.11

The Chruch Bar has been closed. Delete the entry from database.

Solution

db.advanced\_db\_plc.deleteOne(

{

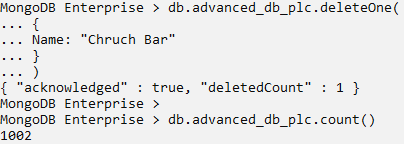
Name: "Chruch Bar"

}

)

db.advanced\_db\_plc.count()

Snip



Question.12

Rialtas na hÉireann have framed new policies for all the pubs opening in 2020. So meanwhile remove all the entries stating the pub started from 2020.

Solution

try {

db.advanced\_db\_plc.deleteMany(

{

Year\_Opened: 2020

}

)

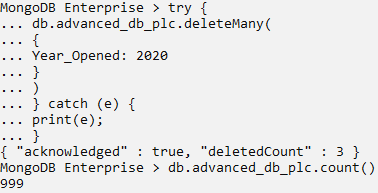
} catch (e) {

print(e);

}

db.advanced\_db\_plc.count()

Snip



Explanation:

‘try and catch’ blocks are used to catch exceptions if any error is generated by the query, it is caught and displayed.

### 4.1.5. Bulk Write

Question.13

There needs to be a new entry for Red and Blue Pub in Limerick. However, some details are incorrectly mentioned, so upload the original details and perform modifications as the correct name is Red, Black and Blue and is located in Tipperary. All these needs to be done with a single query.

Solution

try {

db.advanced\_db\_plc.bulkWrite(

[

{ insertOne :

{

"document" :

{

Name : "Red and Blue",

Address: "Willow Park FC",

Town: "Limerick",

County: "Limerick",

Proprietor\_Name: "Henreita Lacks",

Size\_Square\_Foot: 13500,

Year\_Opened: 2010,

Bar\_Food: "Yes",

Reputation: "Good",

Annual\_Turnover: 395500,

Food: ["Tea Coffee", "Sandwiches", "Snacks"],

Bookings: ["Private Parties", "Birthdays", "Table Quizzes"],

Income:

{

Alcohol: 75000,

Food: 25000,

Parties: 1000000

},

Expense:

{

Wages: 400000,

Insurance: 150000,

Rent: 304500

}

}

}

},

{ updateOne :

{

"filter" : { "Name" : "Red and Blue" },

"update" : { $set : { "Town" : "Tipperary" } }

}

},

{ replaceOne :

{

"filter" : { "Name" : "Red and Blue" },

"replacement" : { "Name" : "Red, Blue and Black", "Proprietor\_Name" : "Henreita Whyte Lacks"}

}

},

{ deleteOne :

{ "filter" : { "Name" : "Red, Blue and Black"} }

}

]

);

}catch (e) {

print(e);

}

Snip





Explanation:

insertOne(), deleteOne(), updateOne(), updateMany(), deleteMany() and replaceOne() are valid operations for bulkWrite()(*db.collection.bulkWrite() — MongoDB Manual*, no date).

## **4.2. Aggregation Framework**

Aggregation framework provides a multi-staged data-pipelining model, where the order of the stages needs to be followed. Different stages have specific transformational functionalities, combinedly generating aggregated document results. Aggregation also adds optimization to queries as compared to simple queries.

### 4.2.1. Distinct

Question.1

Show distinctively, which towns are present in database, also show all the bar food categories.

Solution

db.advanced\_db\_plc.distinct("Town")

db.advanced\_db\_plc.distinct("Food")

Snip



### 4.2.2. $match, $group, $count, $sort

Question.2

Display Proprietor Names and Annual Turnover of all the Pubs situated in Athlone.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match: {"Town" : "Athlone"}

},

{

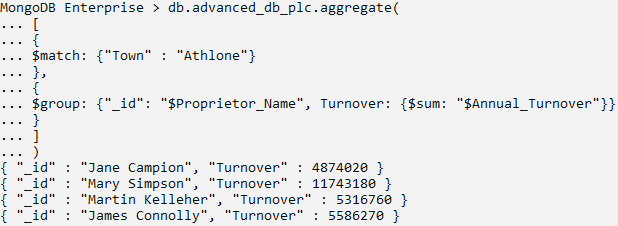
$group: {"\_id": "$Proprietor\_Name", Turnover: {$sum: "$Annual\_Turnover"}}

}

]

)

Snip



Explanation:

$sum accumulates ‘Annual Turnover’ with respect to ‘Proprietor Name’ groups.

Question.3

Show the total count of Proprietors grouped with respect to their corresponding Town, who owns Pubs in Westmeath and have an excellent Reputation.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match: {"Reputation" : "Excellent"}

},

{

$match: {"County" : "Westmeath"}

},

{

$group: {"\_id": {Town: "$Town", Proprietor\_Name: "$Proprietor\_Name"}}

},

{

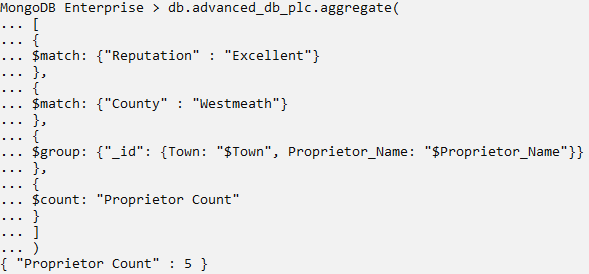
$count: "Proprietor Count"

}

]

)

Snip



Question.4

Display the names of Proprietors and their corresponding Towns in Westmeath with excellent Reputation. Also, sort the results by Proprietor Name in ascending order.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

$and:

[

{"County" : "Westmeath"},

{"Reputation" : "Excellent"}

]

}

},

{

$group: {"\_id": {Town: "$Town", Proprietor\_Name: "$Proprietor\_Name"}}

},

{

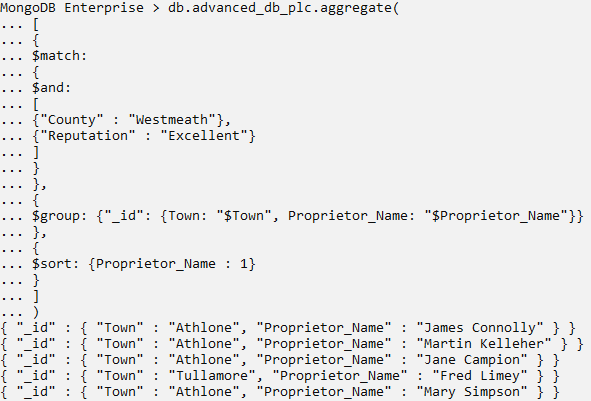
$sort: {Proprietor\_Name : 1}

}

]

)

Snip



Question.5

Display survey results stating how many Pubs in Dublin having excellent Reputation, serves Food and how many do not.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

$and:

[

{"County" : "Dublin"},

{"Reputation" : "Excellent"}

]

}

},

{

$group:

{

\_id: "$Bar\_Food",

count: {$sum : 1}

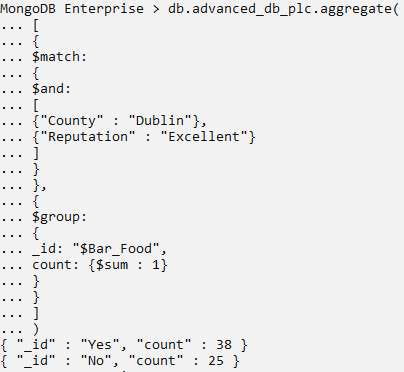
}

}

]

)

Snip



Question.6

Display the Proprietor Names and their average Annual Turnover who pay wages less than 20k.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

"Expense.Wages" : {$lt: 20000}

}

},

{

$group:

{

\_id: "$Proprietor\_Name",

avgTurnover: {$avg: "$Annual\_Turnover"}

}

}

]

)

Snip



Question.7

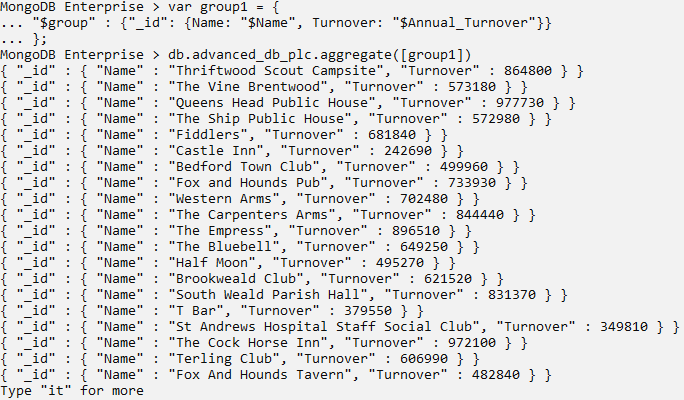
Show grouped results of all the Pubs with its Annual Turnover.

Solution

var group1 = {"$group" : {"\_id": {Name: "$Name", Turnover: "$Annual\_Turnover"}}};

db.advanced\_db\_plc.aggregate([group1])

Snip



Explanation:

MongoDB supports stored variables that can be used over multiple times in the code.

Question.8

Get all the details of Pubs whose

1. Name starts with ‘The’
2. Address has ‘shire’ (case insensitive)
3. And Town ends with ‘y’

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

$and:

[

{Name: {$regex: /^The/}},

{Address: {$regex: /shire/i}},

{Town: {$regex: /y$/}}

]

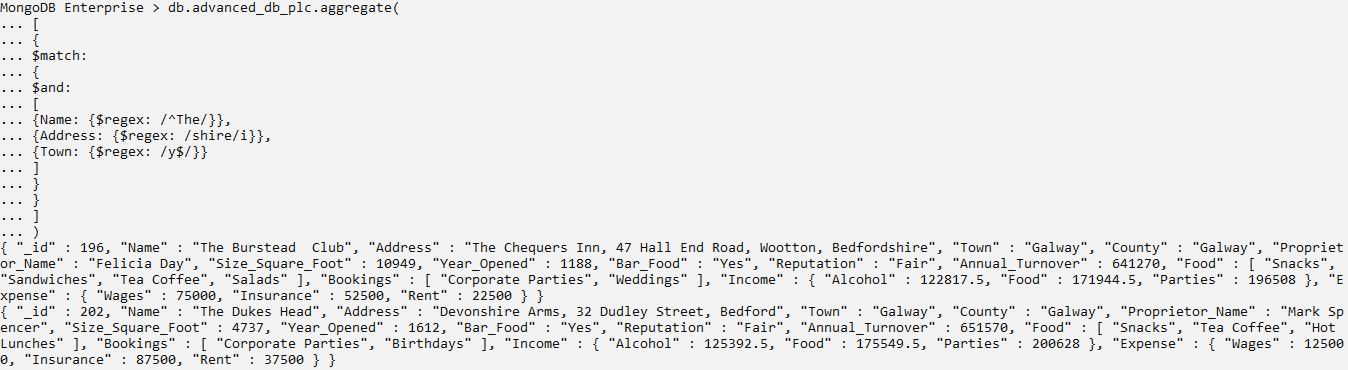
}

}

]

)

Snip



Explanation:

1. /^/ - start of a string
2. /i – case insensitivity
3. $/ - end of a string (*$regex — MongoDB Manual*, no date)

### 4.2.3. $project, $out, $unwind

Question.9

Show the pub details which opened after 1970 and pays 50k to 100k as rent. Show only Pub Name, Town, Size and whether the rent is in the range as the output.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

"Year\_Opened" : {$gt: 1970}

}

},

{

$project:

{

Pub\_Name: "$Name",

Town: "$Town",

Size\_SqFoot: "$Size\_Square\_Foot",

RentIn50k100k:

{

$and:

[

{$gte: ["$Expense.Rent", 50000]},

{$lte: ["$Expense.Rent", 100000]}

]

}

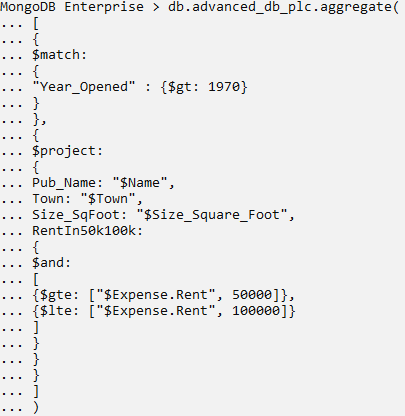
}

}

]

)

Snip





Explanation:

$project stage allows to limit fields in the output. It also provides functionalities to add new calculated variables, i.e. RentIn50k100k (*$project (aggregation) — MongoDB Manual*, no date).

Question.10

Display the datatypes in Charlies Bar of attribute variables: \_id, Name, Size, Food and Income as a database entry in a single query.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

"Name" : "Charlies"

}

},

{

$project:

{

Pub\_Name: "$Name",

typeid: {$type: "$\_id"},

typeName: {$type: "$Name"},

typeSize: {$type: "$Size\_Square\_Foot"},

typeFood: {$type: "$Food"},

typeIncome: {$type: "$Income"}

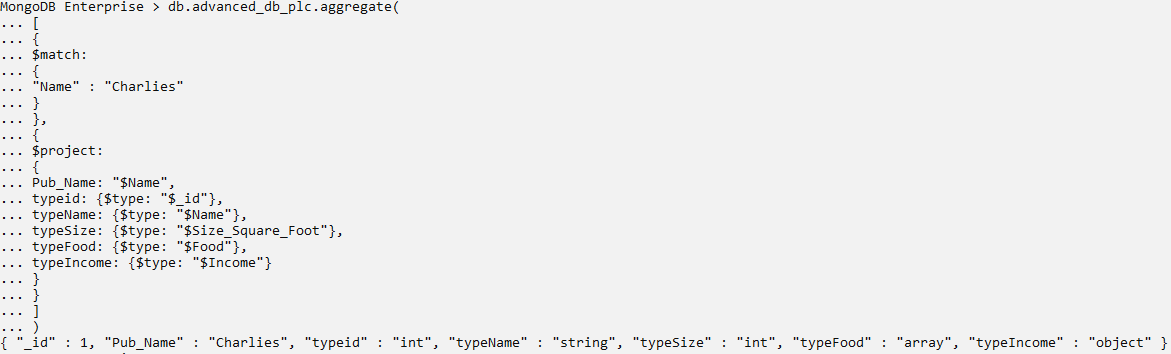
}

}

]

)

Snip



Question.11

Display all the Pubs owned by Martin Kelleher. Display the pub names and the income sources as key – value arrays rather than as embedded documents.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

"Proprietor\_Name" : "Martin Kelleher"

}

},

{

$project:

{

Pub: "$Name",

Income\_Sources: {$objectToArray : "$Income"}

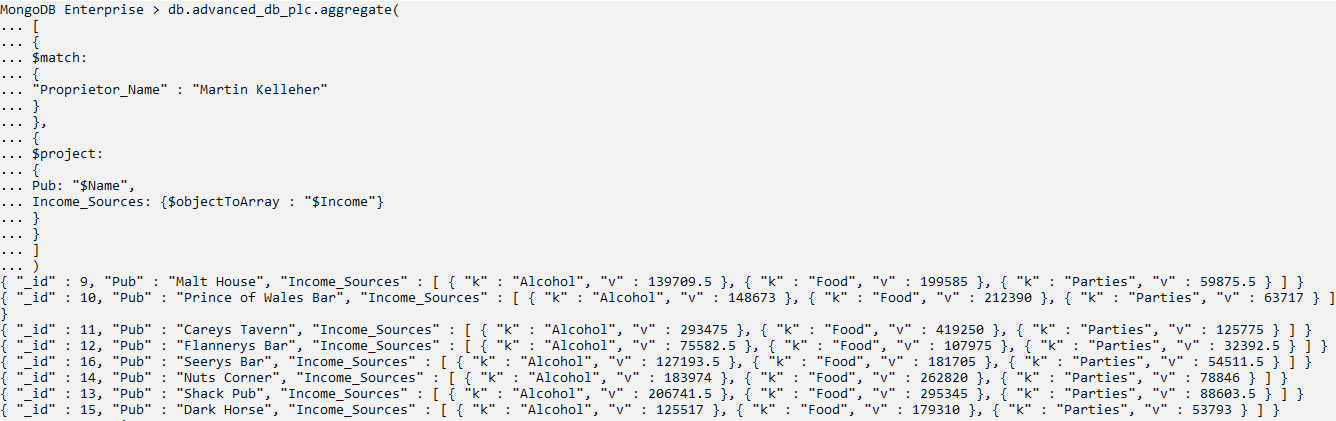
}

}

]

)

Snip



Question.12

Display the names of all pubs in Sligo and add labels of whether it is Large (>= 10k sq. ft) or Small.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$match:

{

"Town" : "Sligo"

}

},

{

$project:

{

Pub: "$Name",

pubSize:

{

$cond:

{

if:

{

$gte: ["$Size\_Square\_Foot", 10000]

},

then: "Large",

else: "Small"

}

}

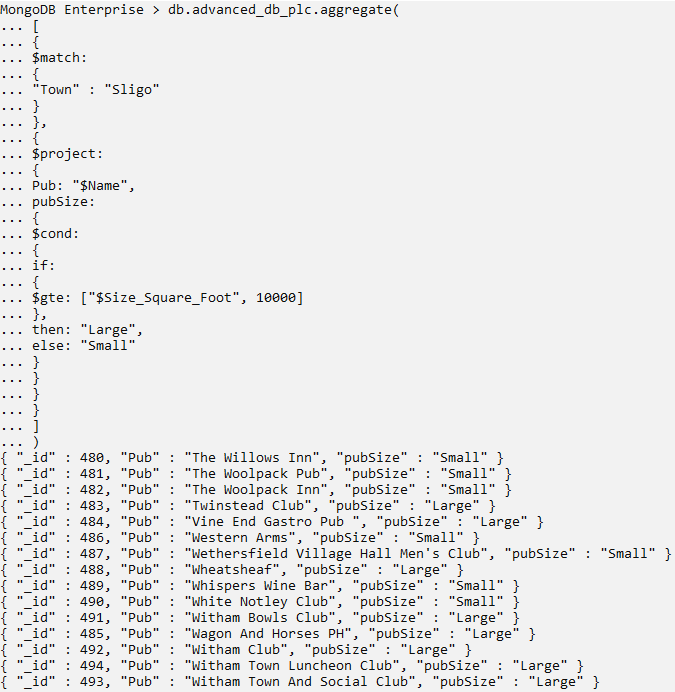
}

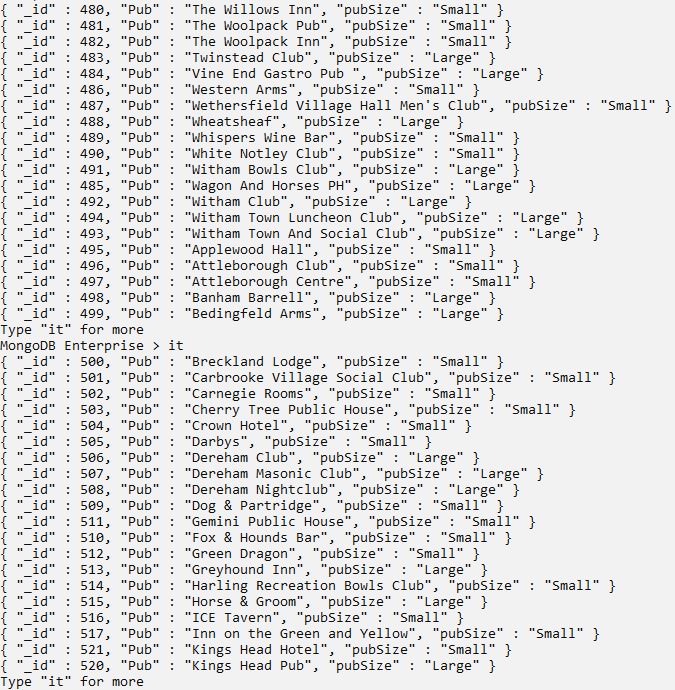
}

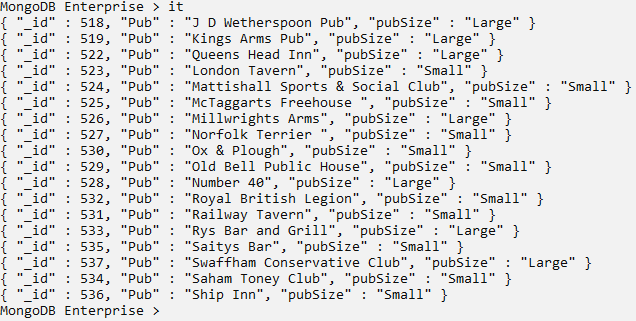
]

)

Snip







Explanation:

Like programming languages, MongoDB provides conditional statement as $cond, which is utilized to add extra insightful labels to the data (*$cond (aggregation) — MongoDB Manual*, no date).

Question.13

Project \_id, Name, Town, Proprietor Name and Bookings of all Pubs located in Loughrea. Also, display the results for each separate bookings unwinded.

Solution

db.advanced\_db\_plc.aggregate(

[

{$unwind: "$Bookings"},

{

$project:

{

\_id: "$id",

Name: "$Name",

Town: "$Town",

Proprietor\_Name: "$Proprietor\_Name",

Bookings: "$Bookings"

}

},

{

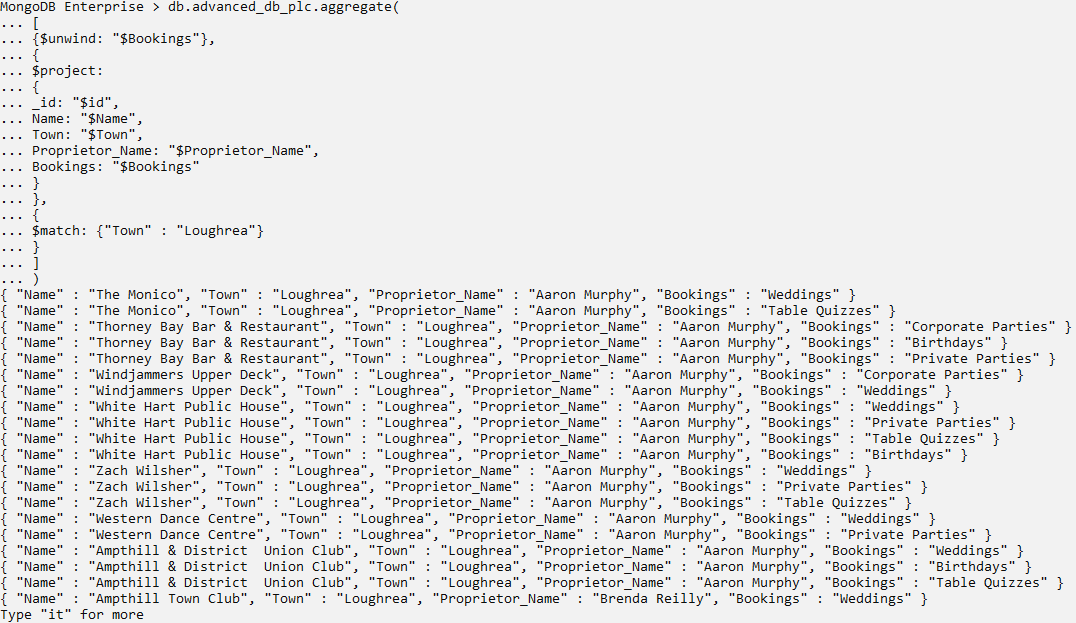
$match: {"Town" : "Loughrea"}

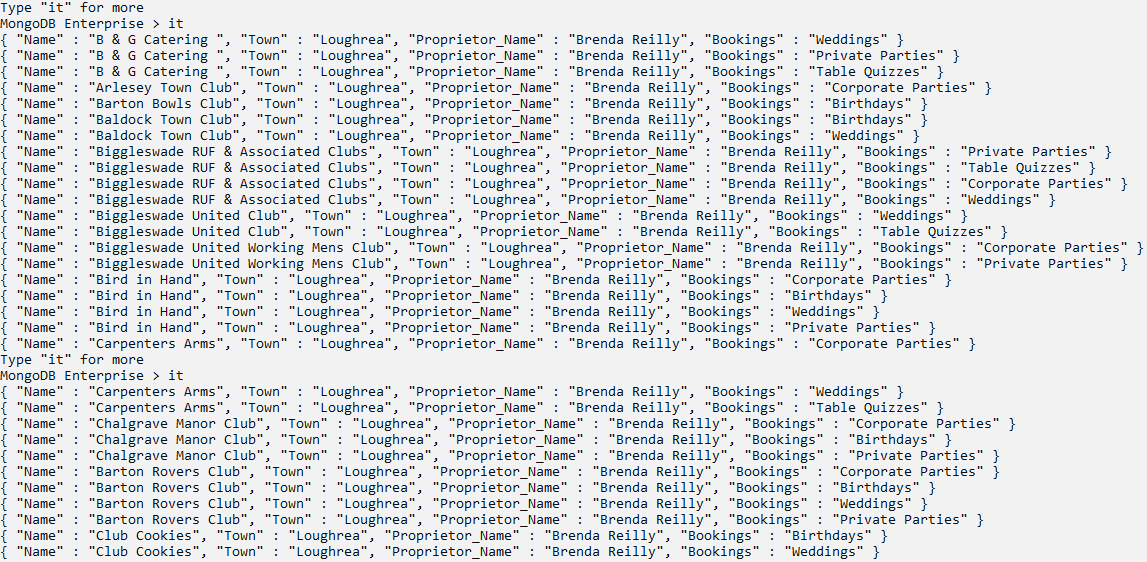
}

]

)

Snip





(*$unwind (aggregation) — MongoDB Manual*, no date)

Question.14

Display the proprietors with their corresponding pubs serially. Store the results in a separate collection: proprietors\_pub.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$group:

{

\_id: "$Proprietor\_Name", pubs: {$push: "$Name"}

}

},

{

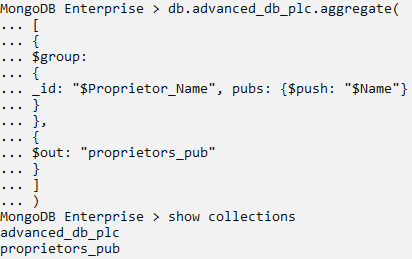
$out: "proprietors\_pub"

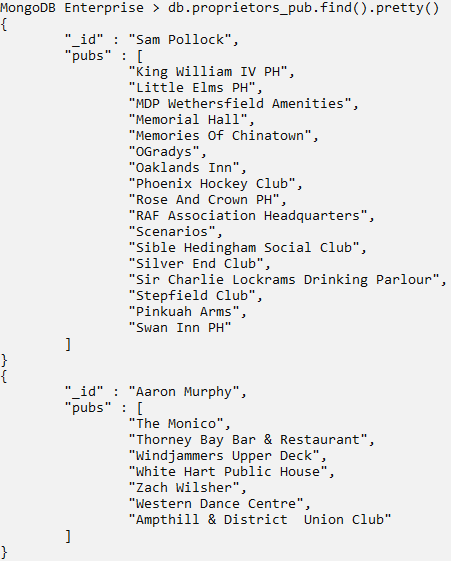
}

]

)

Snip







Explanation:

$out stores the aggregation results on a separate collection specified as a parameter (*$out (aggregation) — MongoDB Manual*, no date).

### 4.2.4. $sample, $collStats

Question.15

Generate a pseudo-random sample of 5% of the database.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$sample:

{

size: 50

}

}

]

)

Snip



Explanation:

It is essential for the sample size to be more than 5% of the total instances to generate pseudo-random sampling (*$sample (aggregation) — MongoDB Manual*, no date).

Question.16

Display latency statistics with histogram information and storage statistics as summary stats to the database.

Solution

db.advanced\_db\_plc.aggregate(

[

{

$collStats:

{

latencyStats: {histograms: true}

}

}

]

)

db.advanced\_db\_plc.aggregate(

[

{

$collStats:

{

storageStats: {}

}

}

]

)

Snip





Explanation:

latencyStats provides a collection of statistics related to request latency for a collection or view. ‘histogram: true’ adds latency histogram information to the embedded documents in latencyStats. storageStats provides a collection of statistics related to a collection’s storage engine. Default scale factor is 1 (*$collStats (aggregation) — MongoDB Manual*, no date).

## **4.3. MongoDB Compass**

MongoDB Compass provides a GUI that privileges user to understand the structure of the data in the database and execute ad-hoc queries without any prior experience with MongoDB query language (*Query Your Data — MongoDB Compass stable*, no date). It also adds a layer of visualization to comprehend the dataset more effectively.

Question.1

Insert a new document for a newly opened pub in Birr.

Snip

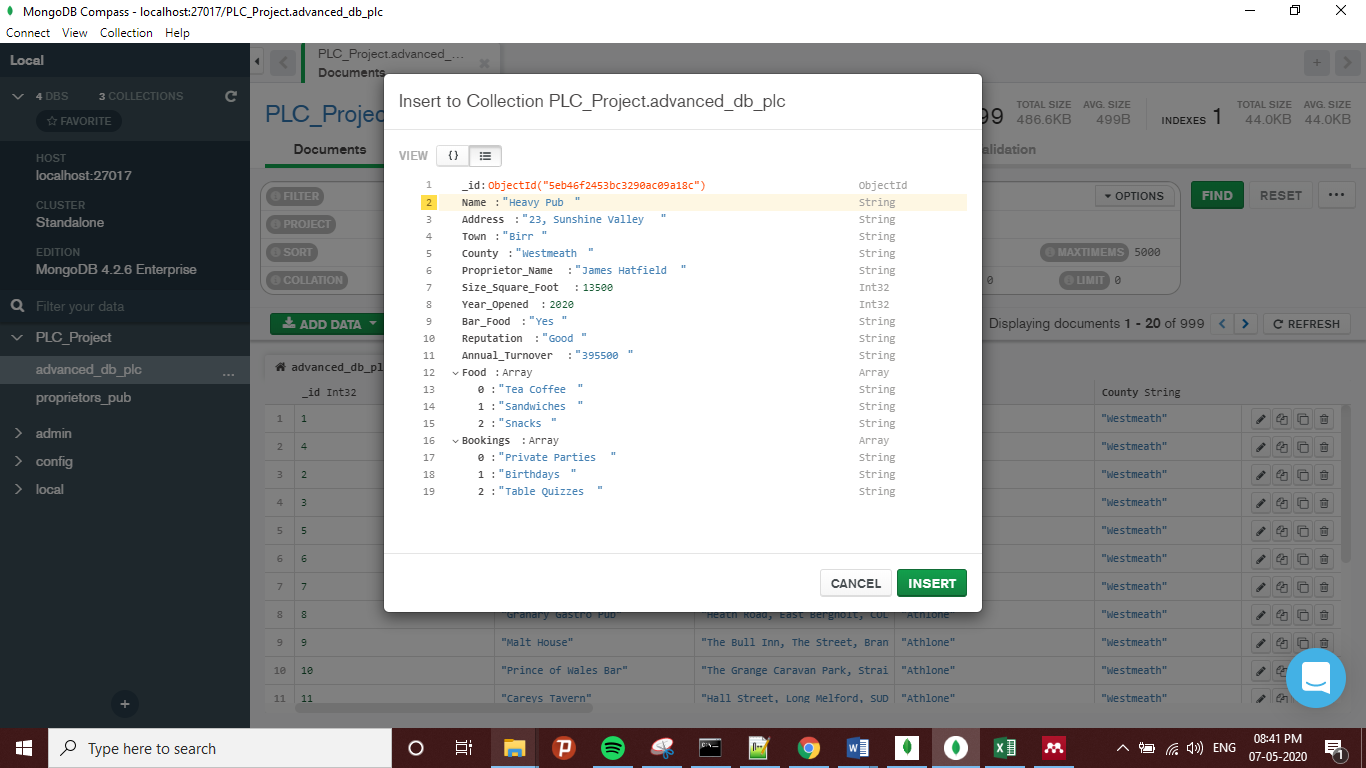


Fig 1: Inserting a document on Compass

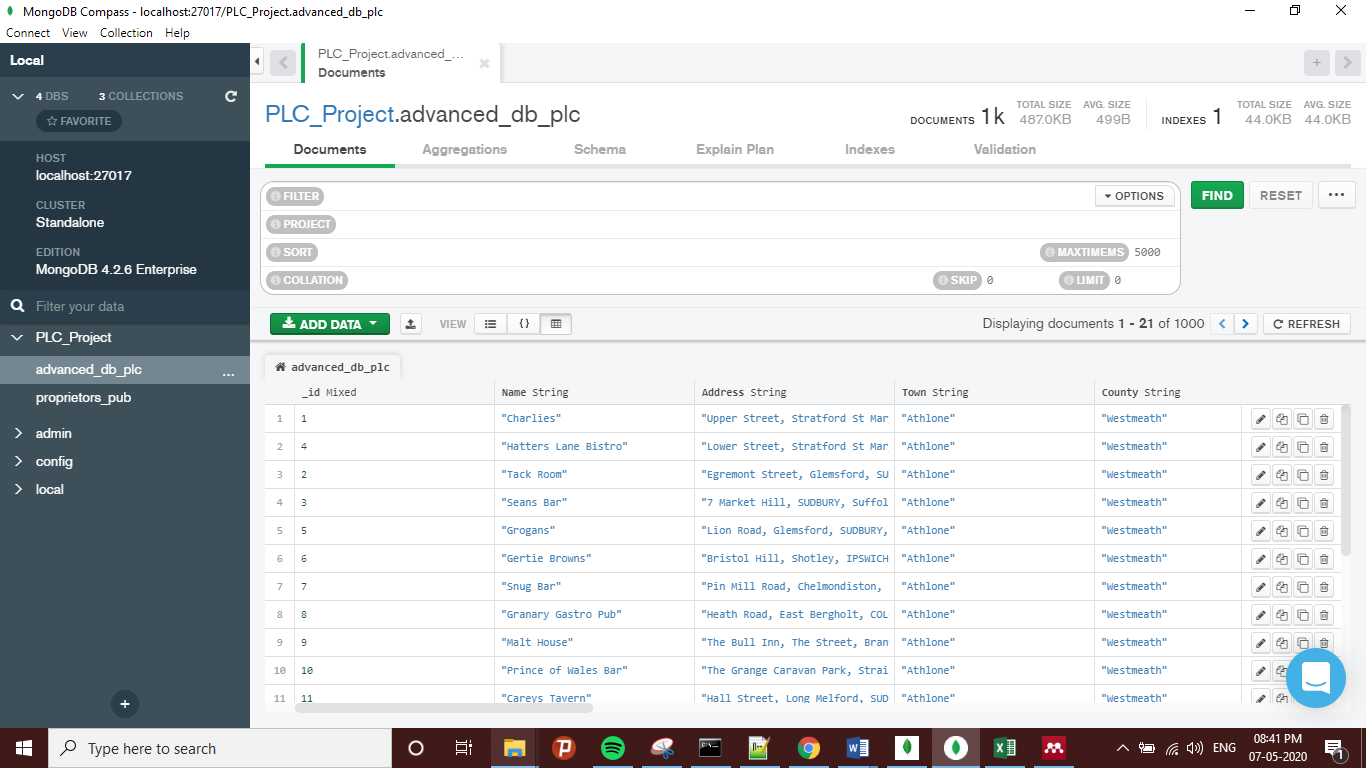


Fig 2: Updated Database

Explanation:

As the database is opened, the first section is documents. A document is inserted simply by going through Add Data, then insert document. Document details can be entered within any of two views.

Question.2

Filter out the documents of Pubs located in Wexford. Display Pub Name, Proprietor Name, Town, Size, Bookings and Income fields only. Sort the results by Size in descending order and then by Pub Name in ascending order. Skip first 10 entries and limit the total output to 10 documents.

Snip

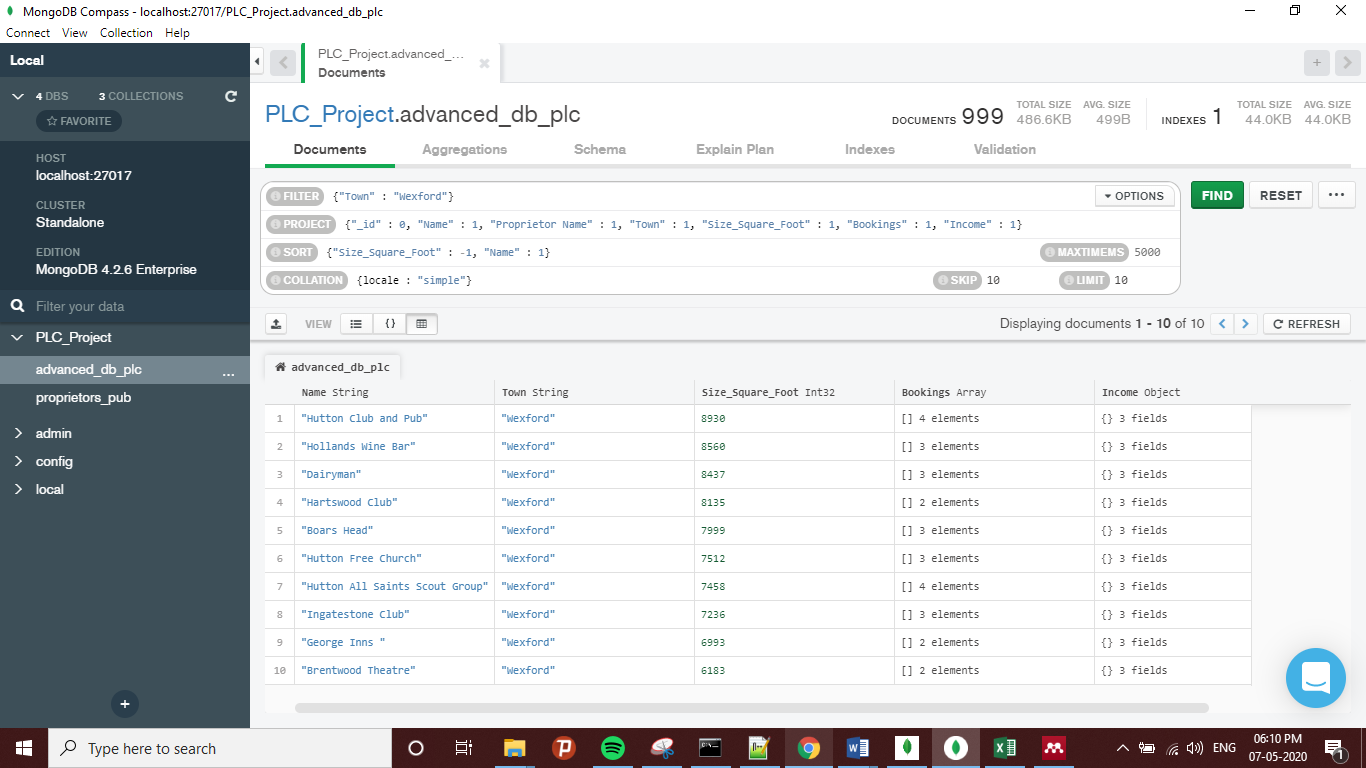


Fig 3: Executing Basic Search Operations

Explanation:

1. Filter: to filter out for searches, as an alias to match.
2. Project: to display or limit the fields in output.
3. Sort: multi-level sorting
4. Collation: to generate language specific rules for string comparisons. The ICU locale consists of codes for languages. Setting it to ‘simple’ specifies simple binary conversion (*Collation — MongoDB Manual*, no date).
5. Maxtimems: to optimize query execution time, can be adjusted according to the complexity of a query.
6. Skip: to skip first n documents.
7. Limit: to limit the result to n documents.

Question.3

Display Name, Town, Size, Bar\_Food, Bookings and Rent details of Pubs whose Size is greater than 10k sq. ft. Also find the maximum rent for each town listed and count the total number of towns in the result.

Snip

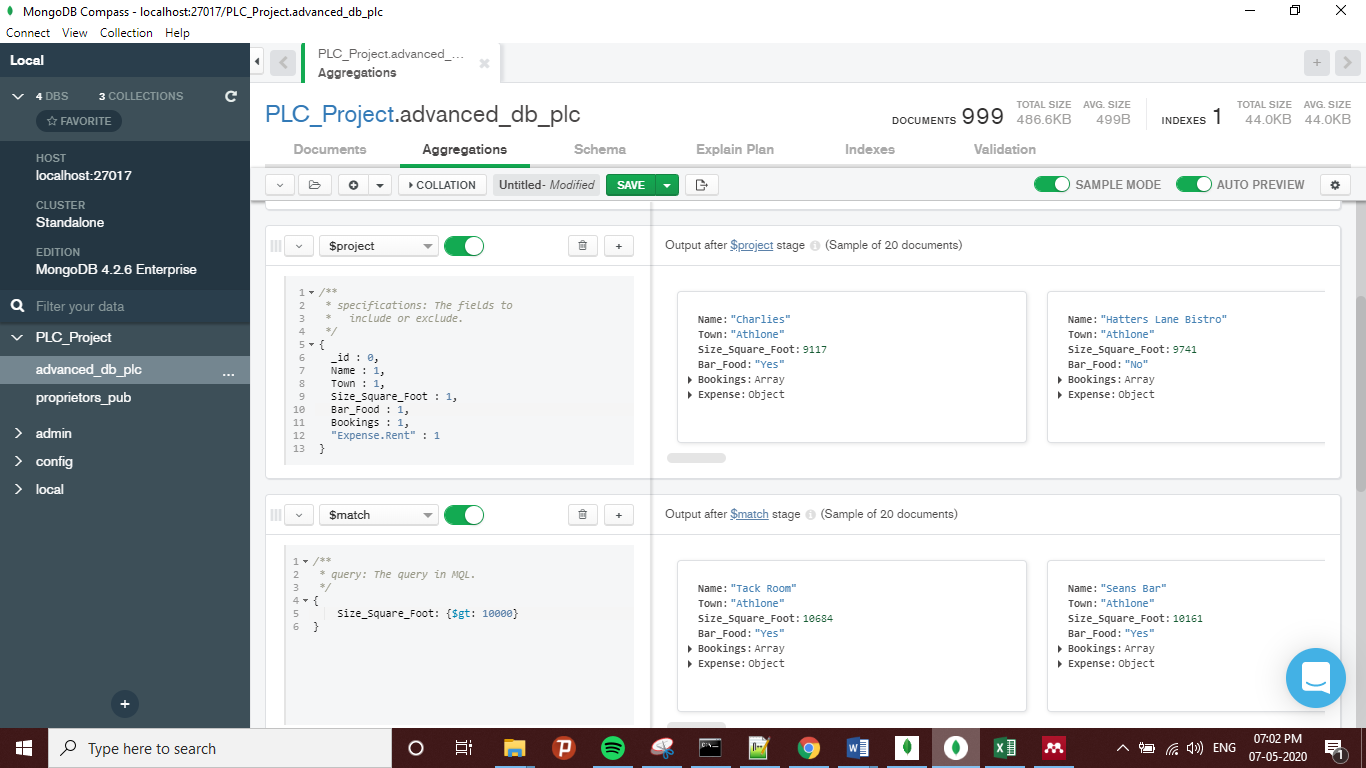


Fig 4: Aggregation pipeline stages: $project, $match

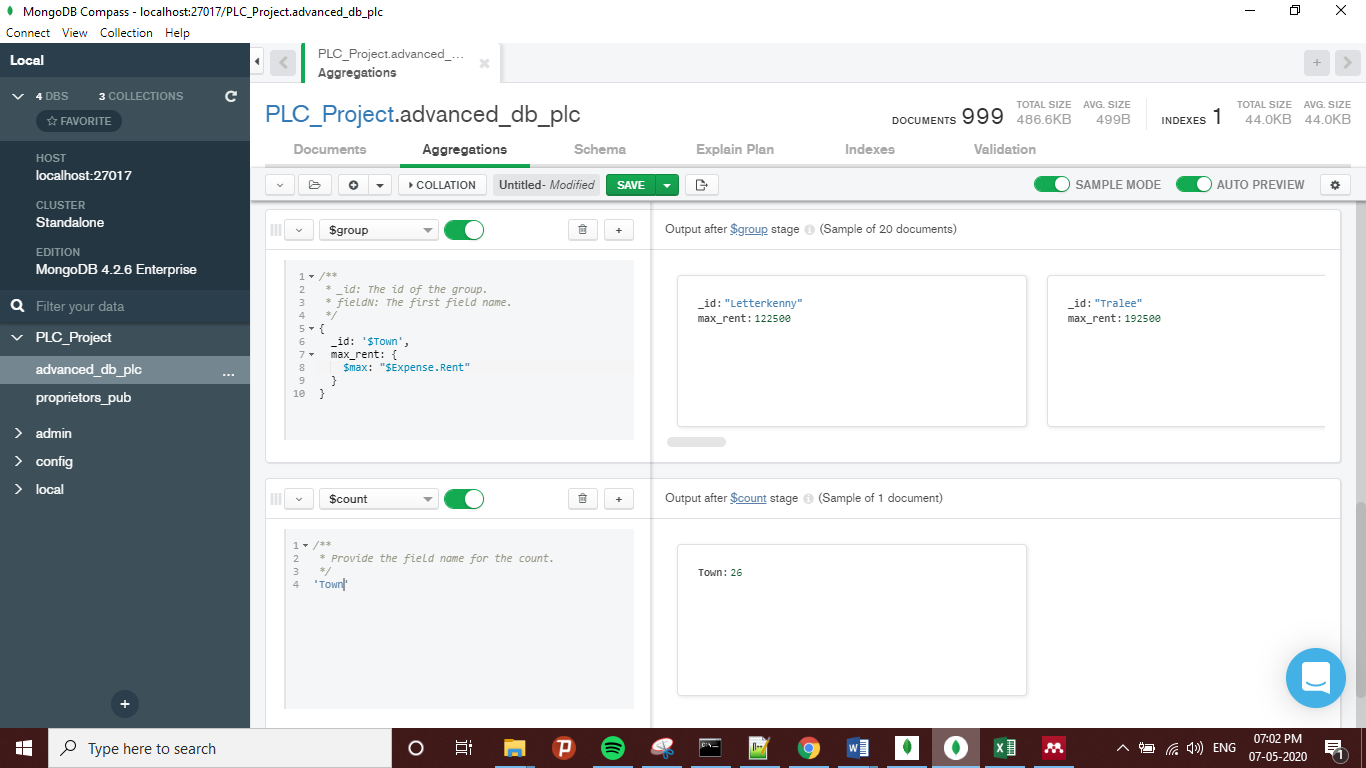


Fig 5: Aggregation pipeline stages: $group, $count

Explanation:

In group stage, a new variable max rent displays the maximum rent of Pubs in each resulting town.

Question.4

Display Name, Town, Size, Bar\_Food, Bookings and Rent details of Pubs which serves Bar Food. Display the maximum rent within each town and include the count of Pubs within that town. Sort the results by maximum rent in descending order.

Snip

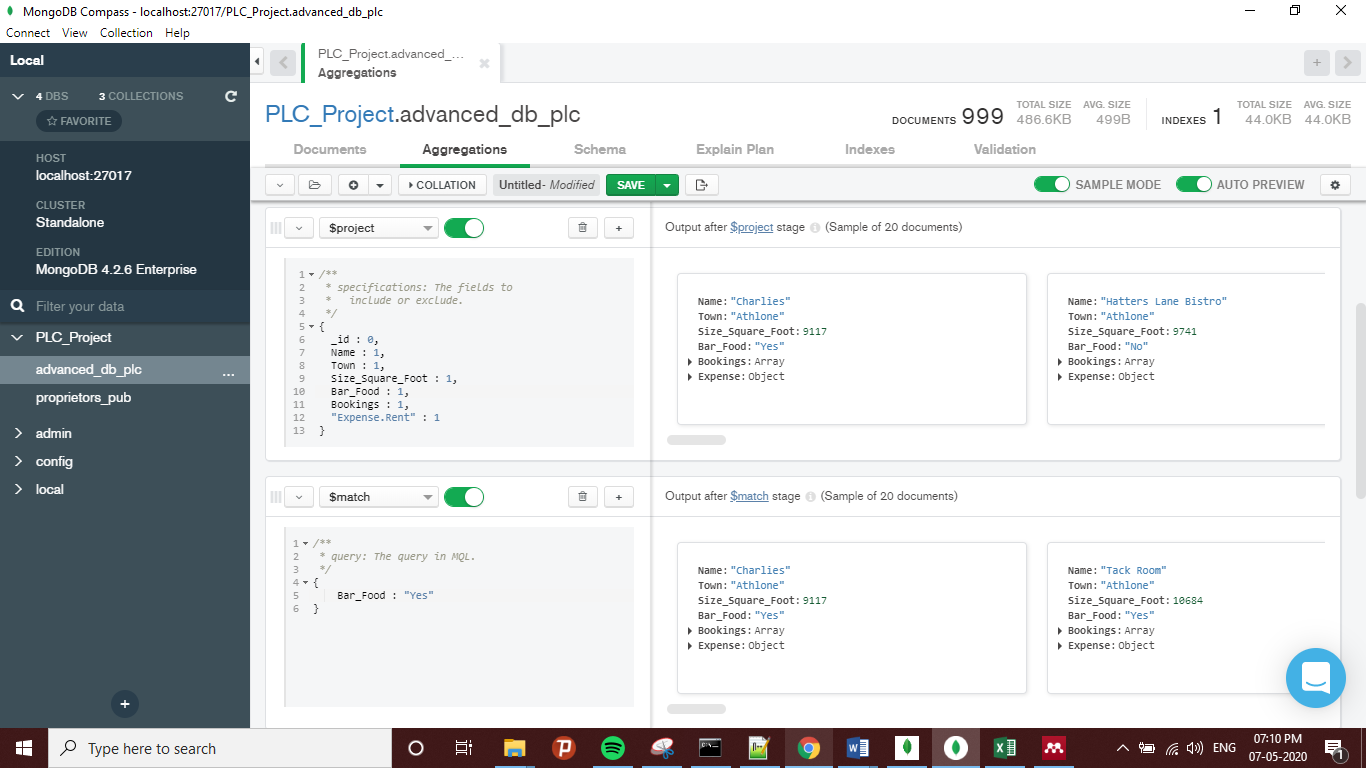


Fig 6: Aggregation pipeline stages: $project, $match

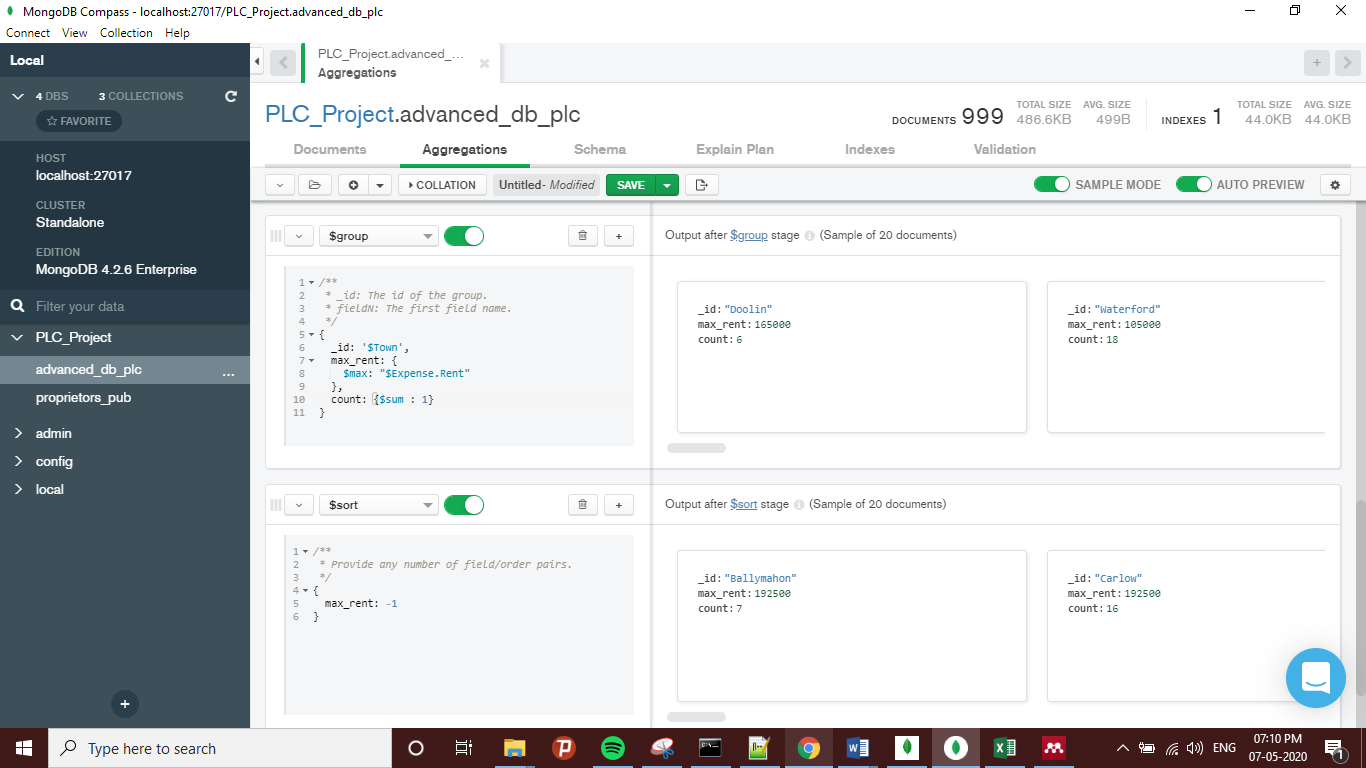


Fig 7: Aggregation pipeline stages: $group, $sort

Explanation:

In this case, the count has to be included with grouping as to count the sum of total entries in each group. The variable formed in $group stage is been used further in next $sort stage.

Question.5

Display Name, Town, Size, Bar\_Food, Bookings and Rent details of Pubs. There has been hike in property rents of 10% in Dublin and Cork, so include new rents. Display the new maximum rent within each town and include the count of Pubs within that town. Sort the results by updated maximum rent in descending order.

Snip

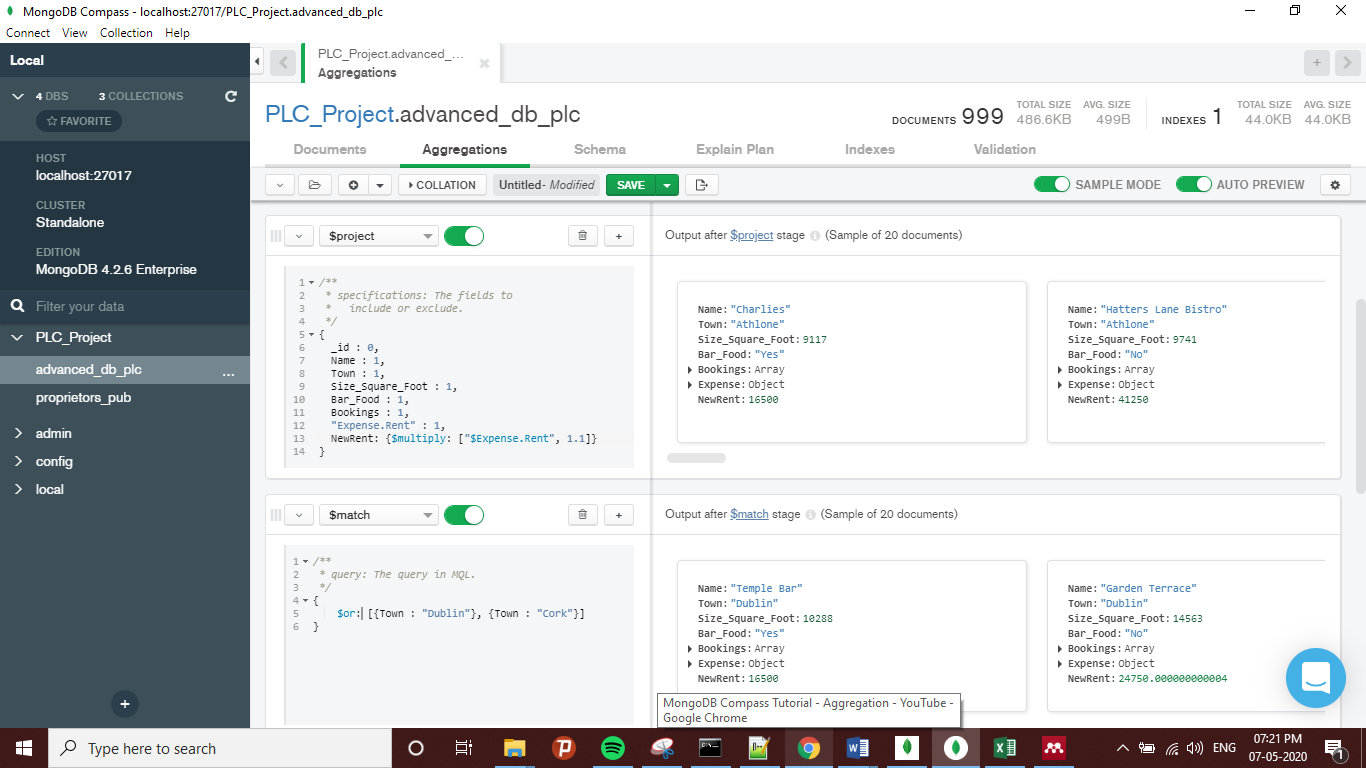


Fig 8: Aggregation pipeline stages: $project, $match

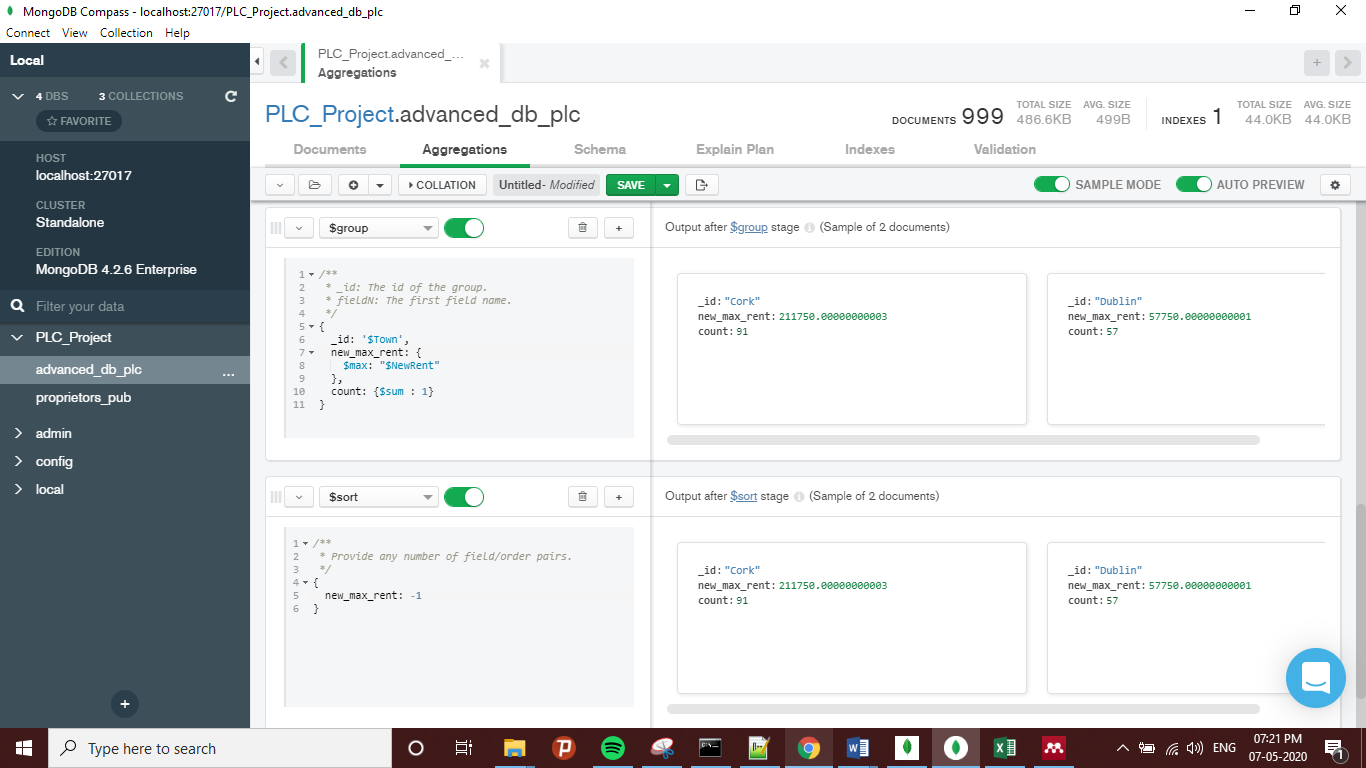


Fig 9: Aggregation pipeline stages: $group, $sort

Explanation:

A calculated variable ‘NewRent’ is formed on $project stage which is reused for $group and $sort stages.

# REFERENCES

*$collStats (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/collStats/ (Accessed: 7 May 2020).

*$cond (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/cond/ (Accessed: 7 May 2020).

*$out (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/out/ (Accessed: 7 May 2020).

*$project (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/project/ (Accessed: 7 May 2020).

*$regex — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/query/regex/ (Accessed: 7 May 2020).

*$sample (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/sample/ (Accessed: 7 May 2020).

*$unwind (aggregation) — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/unwind/ (Accessed: 7 May 2020).

*Collation — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/collation/ (Accessed: 7 May 2020).

*db.collection.bulkWrite() — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/ (Accessed: 7 May 2020).

*db.collection.find() — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/method/db.collection.find/ (Accessed: 7 May 2020).

*db.collection.findAndModify() — MongoDB Manual* (no date). Available at: https://docs.mongodb.com/manual/reference/method/db.collection.findAndModify/ (Accessed: 7 May 2020).

*Query Your Data — MongoDB Compass stable* (no date). Available at: https://docs.mongodb.com/compass/master/query/filter/ (Accessed: 7 May 2020).

# 6. YouTube Description

Screencast1: MongoDB Aggregation Example ($match, $project, $cond)

Screencast2: MongoDB Compass Aggregation: $project, $match, $group, $sort

[Link to Screencast1](https://youtu.be/FIfwx3xWlQU)

[Link to Screencast2](https://youtu.be/04edOpmlniI)