Task C.1. Creating the Databases and Importing Data.

C.1.1: Using the Mongo Shell and appropriate Mongo Shell command(s), create a database called:FIT5137A1MRDB

use FIT5137A1MRDB

```
> use FIT5137A1MRDB
switched to db FIT5137A1MRDB
```

C.1.2: Using the Mongo Shell and appropriate Mongo Shell command(s), create the following 2 collections: userProfiles placeProfiles

db.createCollection("userProfiles")

```
> db.createCollection("userProfiles")
{ "ok" : 1 }
```

db.createCollection("placeProfiles")

```
> db.createCollection("placeProfiles")
{ "ok" : 1 }
```

C.1.3: Using MongoDB Compass and appropriate MongoDB Data Types, import the data in

C.1.3.a: userProfile.json into the userProfiles collection.

C.1.3.b: placeProfiles.json into the placeProfiles collection

C.1.4: In MongoDB, we understand that there are two data modelling methods, which are embedding and referencing.

C.1.4.a: In your own words, explain what you understand by embedding and referencing data models.

Embedded data modelling groups data together logically and could get relationships between data by storing related data in a single document structure. It would not overlap with other data.
for example:
{

```
    _id: 123,
    address: [
        {
            addr_id: 1
            street: '1sadas',
            suburb: '1dasd'
        },
        {
            addr_id: 2
            street: '2sadas',
            suburb: '2dasd'
        }
    ]
}
```

Reference data modelling splits data across collections and stores the relationships between data by including links or references from one document to another.
for example:

```
{
    _id: 123,
    address: [1, 2]
}
{
    addr_id: 1
    street: '1sadas',
    suburb: '1dasd'
},
{
    addr_id: 2
    street: '2sadas',
    suburb: '2dasd'
}
```

C.1.4.b: Import the data in openingHours.csv into your FIT5137A1MRDB database using appropriate data models.

```
db.openingHours.aggregate(
  [{
        $group: {
            _id: "$placeID",
            openingHours: {
                $addToSet: {
                    hours: "$hours",
                    days: "$days"
                }
            }
        }
    },
    {
        $project: {
```

```
            _id: 1,
            openingHours: 1
        }
    },
    {
        $out: "openingHours"
    }
  ]
)


db.placeProfiles.aggregate([{
    $lookup: {
        from: "openingHours",
        localField: "_id",
        foreignField: "_id",
        as: "openInfo"
    }
  },
  {
    $unwind: {
        "path": "$openInfo",
        "preserveNullAndEmptyArrays": true
    }
  },
  {
    $project: {
        _id: 1,
        acceptedPaymentModes: 1,
        address: 1,
        cuisines: 1,
        location: 1,
        parkingArragements: 1,
        placeFeatures: 1,
        placeName: 1,
        openingHours: "$openInfo.openingHours"
    }
  },
  {
    $out: "placeProfiles"
  }
])
```

```
_id: "132560"
acceptedPaymentModes: "cash"
address: Object
    city: "victoria"
    country: "Mexico"
    state: "tamaulipas"
    street: "frente al tecnologico"
cuisines: "Regional"
location: Object
    latitude: "23.7523041"
    longitude: "-99.166913"
parkingArragements: "public"
placeFeatures: Object
    accessibility: "no_accessibility"
    alcohol: "No_Alcohol_Served"
    area: "open"
    dressCode: "informal"
    franchise: "f"
    otherServices: "none"
    price: "low"
    smokingArea: "permitted"
placeName: "puesto de gorditas"
openingHours: Array
  0: Object
      hours: "00:00-00:00;"
      days: "Sat;"
  1: Object
      hours: "00:00-00:00;"
      days: "Sun;"
  2: Object
      hours: "08:00-12:00;"
      days: "Mon;Tue;Wed;Thu;Fri;"
```

C.1.4.c: Include in your report, an explanation behind the selection of the data model.

We use embedded data modelling. As for placeID, it is seen as an identifier of each document. We can aggregate the same placeID. There are different hours for different days. The _id which means place_id In the userProfiles ia unique. So creating an embedded sub-document called openingHours.

C.1.5: Create a keyspace called FIT5137A1_MRDB for the Cassandra database, with SimpleStrategy and replication factor of 1.

```
create keyspace FIT5137A1_MRDB with
   replication = {
      'class': 'SimpleStrategy',
      'replication_factor':1
   };
use FIT5137A1_MRDB;
```

C.1.6: Create the following column families using appropriate data types:

a. user_ratings

```
CREATE TYPE user_personal_traits_type (
   birth_year INT,
   weight INT,
   height FLOAT,
   marital_status TEXT
);
CREATE TYPE user_personality_type (
   interest TEXT,
   type_of_worker TEXT,
   fav_color TEXT,
   drink_level TEXT
);
CREATE TYPE user_preferences_type (
   budget TEXT,
   smoker BOOLEAN,
   dress_preference TEXT,
```

```
  ambience TEXT,
  transport TEXT
);
CREATE TYPE user_other_demographics_type (
  religion TEXT,
  employment TEXT
);
CREATE TABLE user_ratings (
  rating_id INT,
  user_id TEXT,
  place_id TEXT,
  rating_place INT,
  rating_food INT,
  rating_service INT,
  user_personal_traits FROZEN<user_personal_traits_type>,
  user_personality FROZEN<user_personality_type>,
  user_preferences FROZEN<user_preferences_type>,
  user_other_demographics FROZEN<user_other_demographics_type>,
  user_fav_cuisines SET<TEXT>,
  user_fav_payment_method SET<TEXT>,
  PRIMARY KEY (rating_id, user_id)
);
```

```
cqlsh:fit5137a1_mrdb> CREATE TYPE user_personal_traits_type (
                   ...     birth_year INT,
                   ...     weight INT,
                   ...     height FLOAT,
                   ...     marital_status TEXT
                   ... );
CREATE TYPE user_personality_type (
   interest TEXT,
   type_of_worker TEXT,
   fav_color TEXT,
   drink_level TEXT
);
CREATE TYPE user_preferences_type (
   budget TEXT,
   smoker BOOLEAN,
   dress_preference TEXT,
   ambience TEXT,
   transport TEXT
);
CREATE TYPE user_other_demographics_type (
   religion TEXT,
   employment TEXT
);
CREATE TABLE user_ratings (
   rating_id INT,
   user_id TEXT,
   place_id TEXT,
   rating_place INT,
   rating_food INT,
   rating_service INT,
   user_personal_traits FROZEN<user_personal_traits_type>,
   user_personality FROZEN<user_personality_type>,
   user_preferences FROZEN<user_preferences_type>,
   user_other_demographics FROZEN<user_other_demographics_type>,
   user_fav_cuisines SET<TEXT>,
   user_fav_payment_method SET<TEXT>,
   PRIMARY KEY (rating_id, user_id)
);
```

```
cqlsh:fit5137a1_mrdb> describe table user_ratings;

CREATE TABLE fit5137a1_mrdb.user_ratings (
    rating_id int,
    user_id text,
    place_id text,
    rating_food int,
    rating_place int,
    rating_service int,
    user_fav_cuisines set<text>,
    user_fav_payment_method set<text>,
    user_other_demographics frozen<user_other_demographics_type>,
    user_personal_traits frozen<user_personal_traits_type>,
    user_personality frozen<user_personality_type>,
    user_preferences frozen<user_preferences_type>,
    PRIMARY KEY (rating_id, user_id)
) WITH CLUSTERING ORDER BY (user_id ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

b. place_ratings

```
CREATE TYPE place_address_type (
  street TEXT,
  city TEXT,
  state TEXT,
  country TEXT
);
CREATE TYPE place_features_type (
  alcohol TEXT,
  smoking_area TEXT,
  dress_code TEXT,
  accessibility TEXT,
  price TEXT,
  franchise TEXT,
  area TEXT,
  other_services TEXT
);
CREATE TABLE place_ratings (
  rating_id INT,
  user_id TEXT,
  place_id TEXT,
  rating_place INT,
  rating_food INT,
  rating_service INT,
  place_name TEXT,
  place_address FROZEN<place_address_type>,
  place_features FROZEN<place_features_type>,
```

```
  parking_arrangements TEXT,
  accepted_payment_modes SET<TEXT>,
  cuisines FROZEN<SET<TEXT>>,
  PRIMARY KEY (rating_id, user_id)
);
```

```
cqlsh:fit5137a1_mrdb> CREATE TABLE place_ratings (
                   ...     rating_id INT,
                   ...     user_id TEXT,
                   ...     place_id TEXT,
                   ...     rating_place INT,
                   ...     rating_food INT,
                   ...     rating_service INT,
                   ...     place_name TEXT,
                   ...     place_address FROZEN<place_address_type>,
                   ...     place_features FROZEN<place_features_type>,
                   ...     parking_arrangements TEXT,
                   ...     accepted_payment_modes SET<TEXT>,
                   ...     cuisines FROZEN<SET<TEXT>>,
                   ...     PRIMARY KEY (rating_id, user_id)
                   ... );
```

```
cqlsh:fit5137a1_mrdb> describe table place_ratings;

CREATE TABLE fit5137a1_mrdb.place_ratings (
    rating_id int,
    user_id text,
    accepted_payment_modes set<text>,
    cuisines frozen<set<text>>,
    parking_arrangements text,
    place_address frozen<place_address_type>,
    place_features frozen<place_features_type>,
    place_id text,
    place_name text,
    rating_food int,
    rating_place int,
    rating_service int,
    PRIMARY KEY (rating_id, user_id)
) WITH CLUSTERING ORDER BY (user_id ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

C.1.7

a. user_ratings.csv into the user_ratings table.

COPY FIT5137A1_MRDB.user_ratings (rating_id, user_id, place_id, rating_place, rating_food, rating_service, user_personal_traits, user_personality, user_preferences, user_other_demographics, user_fav_cuisines, user_fav_payment_method) FROM 'user_ratings.csv' WITH HEADER = true;

```
cqlsh:fit5137a1_mrdb> COPY FIT5137A1_MRDB.user_ratings (rating_id, user_id, place
_id, rating_place, rating_food, rating_service, user_personal_traits, user_person
ality, user_preferences, user_other_demographics, user_fav_cuisines, user_fav_pay
ment_method) FROM 'user_ratings.csv' WITH HEADER = true;
Using 7 child processes

Starting copy of fit5137a1_mrdb.user_ratings with columns [rating_id, user_id, pl
ace_id, rating_place, rating_food, rating_service, user_personal_traits, user_per
sonality, user_preferences, user_other_demographics, user_fav_cuisines, user_fav_
payment_method].
Processed: 1161 rows; Rate:     186 rows/s; Avg. rate:     360 rows/s
1161 rows imported from 1 files in 3.229 seconds (0 skipped).
cqlsh:fit5137a1_mrdb>
```

b. place_ratings.csv into the place_ratings table.

COPY FIT5137A1_MRDB.place_ratings (rating_id, user_id, place_id, rating_place, rating_food, rating_service, place_name, place_address, place_features, parking_arrangements, accepted_payment_modes, cuisines) FROM 'place_ratings.csv' WITH HEADER = true;

```
cqlsh:fit5137a1_mrdb> COPY FIT5137A1_MRDB.place_ratings (rating_id, user_id, plac
e_id, rating_place, rating_food, rating_service, place_name, place_address, place
_features, parking_arrangements, accepted_payment_modes, cuisines) FROM 'place_ra
tings.csv' WITH HEADER = true;
Using 7 child processes

Starting copy of fit5137a1_mrdb.place_ratings with columns [rating_id, user_id, p
lace_id, rating_place, rating_food, rating_service, place_name, place_address, pl
ace_features, parking_arrangements, accepted_payment_modes, cuisines].
Processed: 1161 rows; Rate:     360 rows/s; Avg. rate:     676 rows/s
1161 rows imported from 1 files in 1.717 seconds (0 skipped).
cqlsh:fit5137a1_mrdb>
```

Task C.2. Modifying the Databases.

C.2.1: MonR has gained some new information about a trendy new place. Therefore, without creating any new fields, insert all of the information provided in Table 1.

```
db.placeProfiles.insertOne({
   "_id": "70000",
   "acceptedPaymentModes": "any",
   "address": {
      "city": "San Luis Potosi",
      "country": "Mexico",
      "state": "SLP",
      "street": "Carretera Central Sn"
   },
   "cuisines": "Mexican, Burgers",
   "parkingArragements": "none",
   "placeFeatures": {
      "accessibility": "completely",
      "alcohol": "No_Alcohol_Served",
      "area": "open",
      "dressCode": "informal",
      "franchise": "f",
      "otherServices": "Internet",
      "price": "medium",
      "smokingArea": "not permitted"
   },
   "placeName": "Taco Jacks",
   "openingHours": [{
         "hours": "09:00-20:00;",
         "days": "Mon;Tue;Wed;Thu;Fri;"
      },
      {
         "hours": "12:00-18:00;",
         "days": "Sat;Sun;"
      }
   ]
})
```

```
> db.placeProfiles.insertOne({
...     "_id": "70000",
...     "acceptedPaymentModes": "any",
...     "address": {
...         "city": "San Luis Potosi",
...         "country": "Mexico",
...         "state": "SLP",
...         "street": "Carretera Central Sn"
...     },
...     "cuisines": "Mexican, Burgers",
...     "parkingArragements": "none",
...     "placeFeatures": {
...         "accessibility": "completely",
...         "alcohol": "No_Alcohol_Served",
...         "area": "open",
...         "dressCode": "informal",
...         "franchise": "f",
...         "otherServices": "Internet",
...         "price": "medium",
...         "smokingArea": "not permitted"
...     },
...     "placeName": "Taco Jacks",
...     "openingHours": [{
...             "hours": "09:00-20:00;",
...             "days": "Mon;Tue;Wed;Thu;Fri;"
...         },
...         {
...             "hours": "12:00-18:00;",
...             "days": "Sat;Sun;"
...         }
...     ]
... })
{ "acknowledged" : true, "insertedId" : "70000" }
>
```

C.2.2: They have also realised that the user with user_id 1108, no longer prefers Fast_Food and also prefers to pay using debit_cards instead of cash. Therefore, without looking up the existing values or adding any new fields, update user 1108's favorite cuisines and favorite payment methods.

```
db.userProfiles.updateOne({
    _id: "1108"
  },
  [
    {
```

```
        $set: {
          favCuisines: {
            $replaceOne: {
              input: "$favCuisines",
              find: "Fast_Food, ",
              replacement: ""
            }
          },
          favPaymentMethod: {
            $replaceOne: {
              input: "$favPaymentMethod",
              find: "cash",
              replacement: "debit_cards"
            }
          }
        }
      },
      {
        $set: {
          favCuisines: {
            $replaceOne: {
              input: "$favCuisines",
              find: "Fast_Food",
              replacement: ""
            }
          }
        }
      }
    ]
)
```

```
> db.userProfiles.updateOne({
...         _id: "1108"
...     },
...     [
...         {
...             $set: {
...                 favCuisines: {
...                     $replaceOne: {
...                         input: "$favCuisines",
...                         find: "Fast_Food, ",
...                         replacement: ""
...                     }
...                 },
...                 favPaymentMethod: {
...                     $replaceOne: {
...                         input: "$favPaymentMethod",
...                         find: "cash",
...                         replacement: "debit_cards"
...                     }
...                 }
...             }
...         },
...         {
...             $set: {
...                 favCuisines: {
...                     $replaceOne: {
...                         input: "$favCuisines",
...                         find: "Fast_Food",
...                         replacement: ""
...                     }
...                 }
...             }
...         }
...     ]
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.userProfiles.find({_id: "1108"}).pretty()
{
        "_id" : "1108",
        "favCuisines" : "Cafe-Coffee_Shop, Sushi, Latin_American, Deli-Sandwiches, Mexican, Hot_Dogs, American, Burgers, Asian, Pizzeria, Chinese, Dessert-Ice_Cream,
Cafeteria, Japanese, Game, Family, Seafood",
        "favPaymentMethod" : "VISA, debit_cards, MasterCard-Eurocard",
        "location" : {
                "latitude" : "22.143524",
                "longitude" : "-100.98756"
        },
        "otherDemographics" : {
                "employment" : "student",
                "religion" : "Catholic"
        },
        "personalTraits" : {
                "birthYear" : "1983",
                "height" : "1.81",
                "maritalStatus" : "single",
                "weight" : "76"
        },
        "personality" : {
                "drinkLevel" : "abstemious",
                "favColor" : "blue",
                "interest" : "technology",
                "typeOfWorker" : "thrifty-protector"
        },
        "preferences" : {
                "ambience" : "solitary",
                "budget" : "medium",
                "dressPreference" : "informal",
                "smoker" : "FALSE",
                "transport" : "public"
        }
}
```

C.2.3: The management has realised that the user with user_id 1063 was an error. Therefore delete the user 1063 from the database.

```
db.userProfiles.deleteOne(
  {
    _id: "1063"
  }
)
```

```
> db.userProfiles.deleteOne(
...     {
...         _id: "1063"
...     }
... )
{ "acknowledged" : true, "deletedCount" : 1 }
```

```
> db.userProfiles.find({_id: "1063"}).pretty()
>
```

C.2.4: To be consistent with the changes made in Task C.2 (1), (2), and (3), the management has also requested to update the reviews information to reflect the changes made to the users 1108 and remove the user 1063's reviews. They have looked up the data in the reviews table and have provided the information in table 2.

```
UPDATE user_ratings
SET user_fav_cuisines = user_fav_cuisines - {'Fast_Food'}
```

WHERE rating_id IN (65, 66, 67, 68, 69, 70, 71, 72, 73, 74) AND user_id = '1108';

```
cqlsh:fit5137a1_mrdb> UPDATE user_ratings
                  ... SET user_fav_cuisines = user_fav_cuisines - {'Fast_Food'}
                  ... WHERE rating_id IN (65, 66, 67, 68, 69, 70, 71, 72, 73, 74) AND user_id = '1108';
```

UPDATE user_ratings
SET user_fav_payment_method = user_fav_payment_method - {'cash'}, user_fav_payment_method = user_fav_payment_method + {'debit_cards'}
WHERE rating_id IN (65, 66, 67, 68, 69, 70, 71, 72, 73, 74) AND user_id = '1108';

```
cqlsh:fit5137a1_mrdb> UPDATE user_ratings
                  ... SET user_fav_payment_method = user_fav_payment_method - {'cash'}, user_fav_payment_method = user_fav_payment_method + {'debit_cards'}
                  ... WHERE rating_id IN (65, 66, 67, 68, 69, 70, 71, 72, 73, 74) AND user_id = '1108';
```

```
cqlsh:fit5137a1_mrdb> select * from user_ratings where rating_id = 65 and user_id = '1108';

 rating_id | user_id | place_id | rating_food | rating_place | rating_service | user_fav_cuisines
                                                                                                   | user_fav_payment_method
          | user_other_demographics                          | user_personal_traits                                    | user_personality
                                                                    | user_preferences
-----------+---------+----------+-------------+--------------+----------------+------------------------------------------
---------------------------------------------------------------------------------------------------+------------------------
----------------+------------------------------------------------+-------------------------------------------+--------------
---------------------------------------------------------------------+
-----------------------------
        65 |    1108 |   135075 |           2 |            2 |              2 | {'American', 'Asian', 'Burgers', 'Cafe-Coffee_Shop', 'Cafeteria', 'Chinese', 'Deli-Sa
ndwiches', 'Dessert-Ice_Cream', 'Family', 'Game', 'Hot_Dogs', 'Japanese', 'Latin_American', 'Mexican', 'Pizzeria', 'Seafood', 'Sushi'} | {'MasterCard-Eurocard', 'VIS
A', 'debit_cards'} | {religion: 'Catholic', employment: 'student'} | {birth_year: 1983, weight: 76, height: 1.81, marital_status: 'single'} | {interest: 'technology'
, type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'abstemious'} | {budget: 'medium', smoker: False, dress_preference: 'informal', ambience: 'sol
itary', transport: 'public'}

(1 rows)
```

DELETE FROM user_ratings WHERE rating_id IN (137, 138, 139, 140, 141) AND user_id = '1063';

```
cqlsh:fit5137a1_mrdb> DELETE FROM user_ratings WHERE rating_id IN (137, 138, 139, 140, 141) AND user_id = '1063';
```

```
cqlsh:fit5137a1_mrdb> select * from user_ratings where rating_id = 137 and user_id = '1063';

 rating_id | user_id | place_id | rating_food | rating_place | rating_service | user_fav_cuisines | user_fav_payment_method | user_other_demographics | user_personal
_traits | user_personality | user_preferences
-----------+---------+----------+-------------+--------------+----------------+-------------------+-------------------------+-------------------------+--------------
--------+------------------+------------------

(0 rows)
```

C.2.5: It was also seen that user 1060 has reviewed Taco Jacks (ie. the new place with place id 70000), therefore using the information from table 3, insert the following data: (for this insert only you may look up the details of user 1060).

```
INSERT INTO user_ratings (
  rating_id,
  user_id,
  place_id,
  rating_place,
  rating_food,
  rating_service,
  user_personal_traits,
  user_personality,
  user_preferences,
```

```
    user_other_demographics,
    user_fav_cuisines,
    user_fav_payment_method
) VALUES (
    7777,
    '1060',
    '70000',
    2,
    1,
    2,
    {birth_year: 1991, weight: 82, height: 1.84, marital_status: 'single'},
    {interest: 'technology', type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'casual drinker'},
    {budget: 'medium', smoker: False, dress_preference: 'formal', ambience: 'family', transport: 'public'},
    {religion: 'Catholic', employment: 'student'},
    {'American', 'Burgers', 'Cafe-Coffee_Shop', 'Cafeteria', 'Fast_Food', 'Hot_Dogs', 'Italian', 'Juice', 'Mexican', 'Pizzeria', 'Soup', 'Spanish', 'Tex-Mex'},
    {'cash'}
);
```

```
cqlsh:fit5137a1_mrdb> INSERT INTO user_ratings (
                  ...     rating_id,
                  ...     user_id,
                  ...     place_id,
                  ...     rating_place,
                  ...     rating_food,
                  ...     rating_service,
                  ...     user_personal_traits,
                  ...     user_personality,
                  ...     user_preferences,
                  ...     user_other_demographics,
                  ...     user_fav_cuisines,
                  ...     user_fav_payment_method
                  ... ) VALUES (
                  ...     7777,
                  ...     '1060',
    '70000',
    2,
    1,
    2,
    {birth_year: 1991, weight: 82, height: 1.84, marital_status: 'single'},
    {interest: 'technology', type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'casual drinker'},
    {budget: 'medium', smoker: False, dress_preference: 'formal', ambience: 'family', transport: 'public'},
    {religion: 'Catholic', employment: 'student'},
    {'American', 'Burgers', 'Cafe-Coffee_Shop', 'Cafeteria', 'Fast_Food', 'Hot_Dogs', 'Italian', 'Juice', 'Mexican', 'Pizzeria', 'Soup', 'Spanish', 'Tex-Mex'},
    {'cash'}
);                  ...     '70000',
                  ...     2,
                  ...     1,
                  ...     2,
                  ...     {birth_year: 1991, weight: 82, height: 1.84, marital_status: 'single'},
                  ...     {interest: 'technology', type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'casual drinker'},
                  ...     {budget: 'medium', smoker: False, dress_preference: 'formal', ambience: 'family', transport: 'public'},
                  ...     {religion: 'Catholic', employment: 'student'},
                  ...     {'American', 'Burgers', 'Cafe-Coffee_Shop', 'Cafeteria', 'Fast_Food', 'Hot_Dogs', 'Italian', 'Juice', 'Mexican', 'Pizzeria', 'Soup', 'Spani
sh', 'Tex-Mex'},
                  ...     {'cash'}
                  ... );
cqlsh:fit5137a1_mrdb>
```

```
cqlsh:fit5137a1_mrdb> select * from user_ratings where rating_id = 7777;

 rating_id | place_id | rating_food | rating_place | rating_service | user_fav_cuisines
                                                    | user_fav_payment_method | user_id | user_other_demographics                    | user_personal_trait
s                                             | user_personality
 | user_preferences
-----------+----------+-------------+--------------+----------------+------------------------------------------------------------------------------------
-------------------------------------------------------+-----------------------+---------+-------------------------------------------+-------------------
-----------------------------------------------+-----------------------------------------------------------------------------------------------------
-+------------------------------------------------------------------------------------
      7777 |    70000 |           1 |            2 |              2 | {'American', 'Burgers', 'Cafe-Coffee_Shop', 'Cafeteria', 'Fast_Food', 'Hot_Dogs', 'Italian', 'J
uice', 'Mexican', 'Pizzeria', 'Soup', 'Spanish', 'Tex-Mex'} |                {'cash'} |    1060 | {religion: 'Catholic', employment: 'student'} | {birth_year: 1991,
weight: 82, height: 1.84, marital_status: 'single'} | {interest: 'technology', type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'casual drinker'}
 | {budget: 'medium', smoker: False, dress_preference: 'formal', ambience: 'family', transport: 'public'}

(1 rows)
```

Task C.3. Querying the Management.

C.3.1

1. db.userProfiles.count()

```
> db.userProfiles.count()
137
>
```

C.3.2

db.placeProfiles.count()

```
> db.placeProfiles.count()
131
>
```

C.3.3

select count(*) from place_ratings;

```
cqlsh:fit5137a1_mrdb> select count(*) from place_ratings;

 count
-------
  1161
```

C.3.4

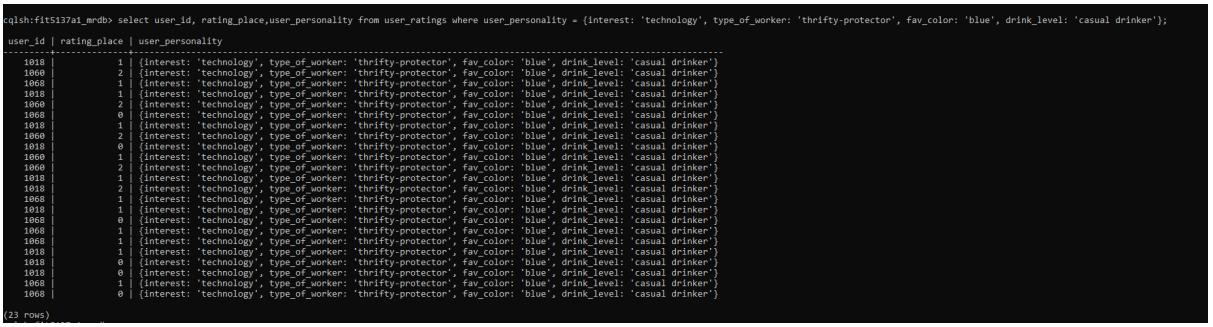
CREATE INDEX parking ON place_ratings (parking_arrangements);
select count(*) from place_ratings where parking_arrangements = 'public';

```
cqlsh:fit5137a1_mrdb> select count(*) from place_ratings where parking_arrangements = 'public';

 count
-------
   182

(1 rows)
```

C.3.5

CREATE INDEX user_persenality ON fit5137a1_mrdb.user_ratings(user_personality);

select user_id, rating_place,user_personality from user_ratings where user_personality = {interest: 'technology', type_of_worker: 'thrifty-protector', fav_color: 'blue', drink_level: 'casual drinker'};



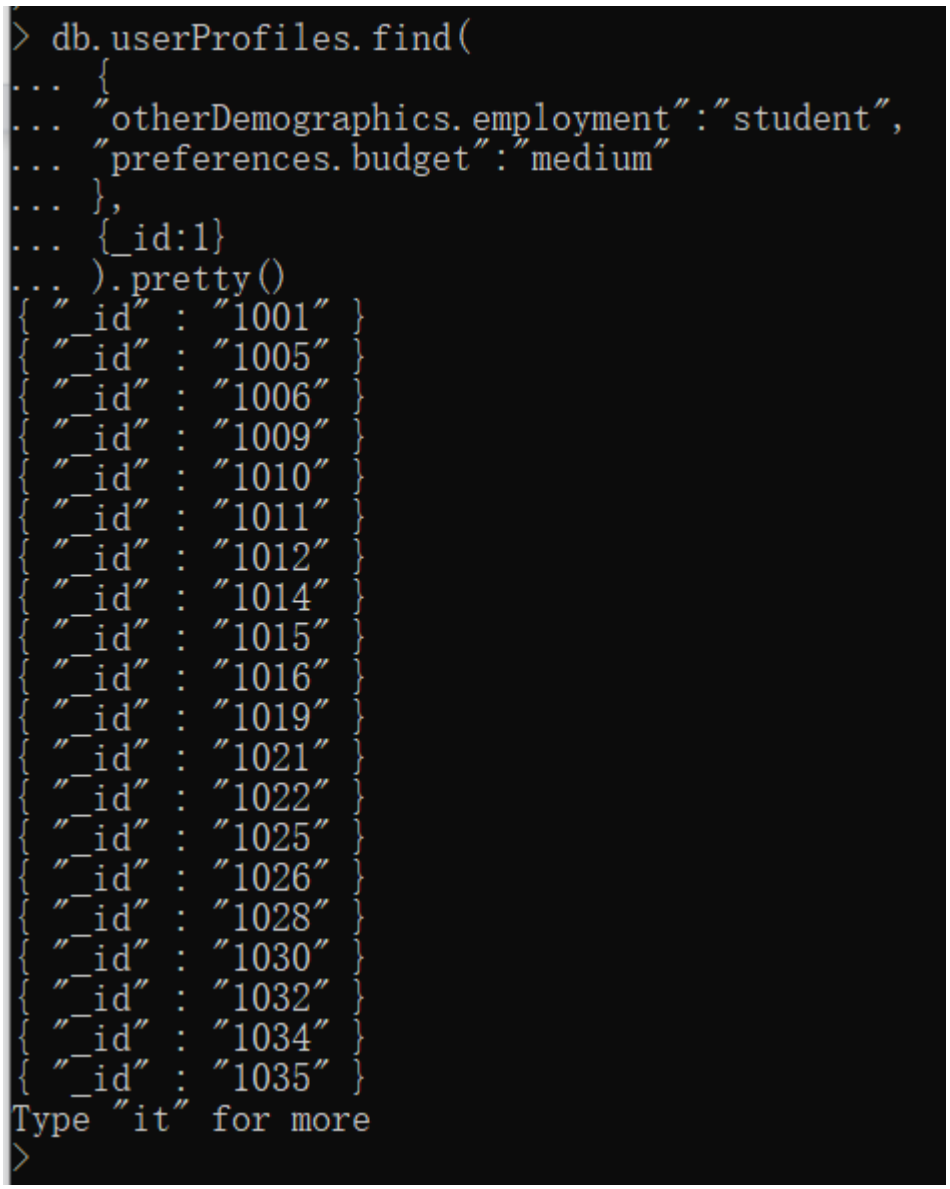
C.3.6

create index cuisin_full on place_ratings(full(cuisines));
select place_id,rating_food,cuisines from place_ratings where cuisines = {'Pizzeria'};

```
cqlsh:fit5137a1_mrdb> select place_id,rating_food,cuisines from place_ratings where cuisines = {'Pizzeria'};

 place_id | rating_food | cuisines
----------+-------------+--------------
   132733 |           2 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   132733 |           2 | {'Pizzeria'}
   132869 |           2 | {'Pizzeria'}
   132733 |           2 | {'Pizzeria'}
   132869 |           1 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   132733 |           0 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   132869 |           2 | {'Pizzeria'}
   132733 |           1 | {'Pizzeria'}
   132733 |           1 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   132869 |           0 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   132733 |           1 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   132733 |           1 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   135058 |           2 | {'Pizzeria'}
   135058 |           1 | {'Pizzeria'}
   132733 |           2 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   132869 |           1 | {'Pizzeria'}
   135058 |           0 | {'Pizzeria'}
   132869 |           2 | {'Pizzeria'}
   132733 |           1 | {'Pizzeria'}

(34 rows)
```

C.3.7

```
db.userProfiles.find(
{
        "otherDemographics.employment":"student",
        "preferences.budget":"medium"
},
{_id:1}
).pretty()
```



OR with showing the full information:

```
db.userProfiles.find(
{
        "otherDemographics.employment":"student",
        "preferences.budget":"medium"
}
)
```

C.3.8

```
db.userProfiles.aggregate([

	{$match:{
		favCuisines : {$regex:"Bakery"}
	}},
	{$project:{
		_id:0,
		"user":"$_id"
	}},
	{
		$merge:{
			into:"Q8"
		}
	}

])

db.placeProfiles.aggregate([
	{
		$match:{
			cuisines:{$regex:"Bakery"}
		}
	},
	{
		$project:{
			cuisines:1,
			"Restaruant":"$_id",
			_id:0
		}
	},
	{
		$merge:{
```

```
                into:"Q8"
            }
        }
])

db.Q8.find({})
```

```
> db.Q8.find({})
{ "_id" : ObjectId("5f7c56fe7e52d7cf69c1aeee"), "user" : "1004" }
{ "_id" : ObjectId("5f7c56fe7e52d7cf69c1aeef"), "user" : "1052" }
{ "_id" : ObjectId("5f7c56fe7e52d7cf69c1aef0"), "user" : "1135" }
{ "_id" : ObjectId("5f7c570c7e52d7cf69c1aefc"), "Restaruant" : "132866", "cuisines" : "Bakery, Cafeteria" }
>
```

C.3.9

```
db.placeProfiles.aggregate([
        {
            $match:{
                cuisines:"International",
                "openingHours.days":"Sun;",
                "openingHours.hours":{
                    $nin:["0:00-0:00;"]
                }
            }
        },
        {
            $project:{
                placeName:1
            }
        }
])
```

```
> db.placeProfiles.aggregate([{
...         $match: {
...             cuisines: "International",
...             "openingHours.days": "Sun;",
...             "openingHours.hours": {
...                 $nin: ["0:00-0:00;"]
...             }
...         }
...     },
...     {
...         $project: {
...             placeName: 1
...         }
...     }
... ])
{ "_id" : "132846", "placeName" : "el lechon potosino" }
{ "_id" : "132854", "placeName" : "Sirlone" }
{ "_id" : "132862", "placeName" : "La Posada del Virrey" }
{ "_id" : "134986", "placeName" : "Restaurant Las Mananitas" }
```

C.3.10

create index place_names on place_ratings(place_name);

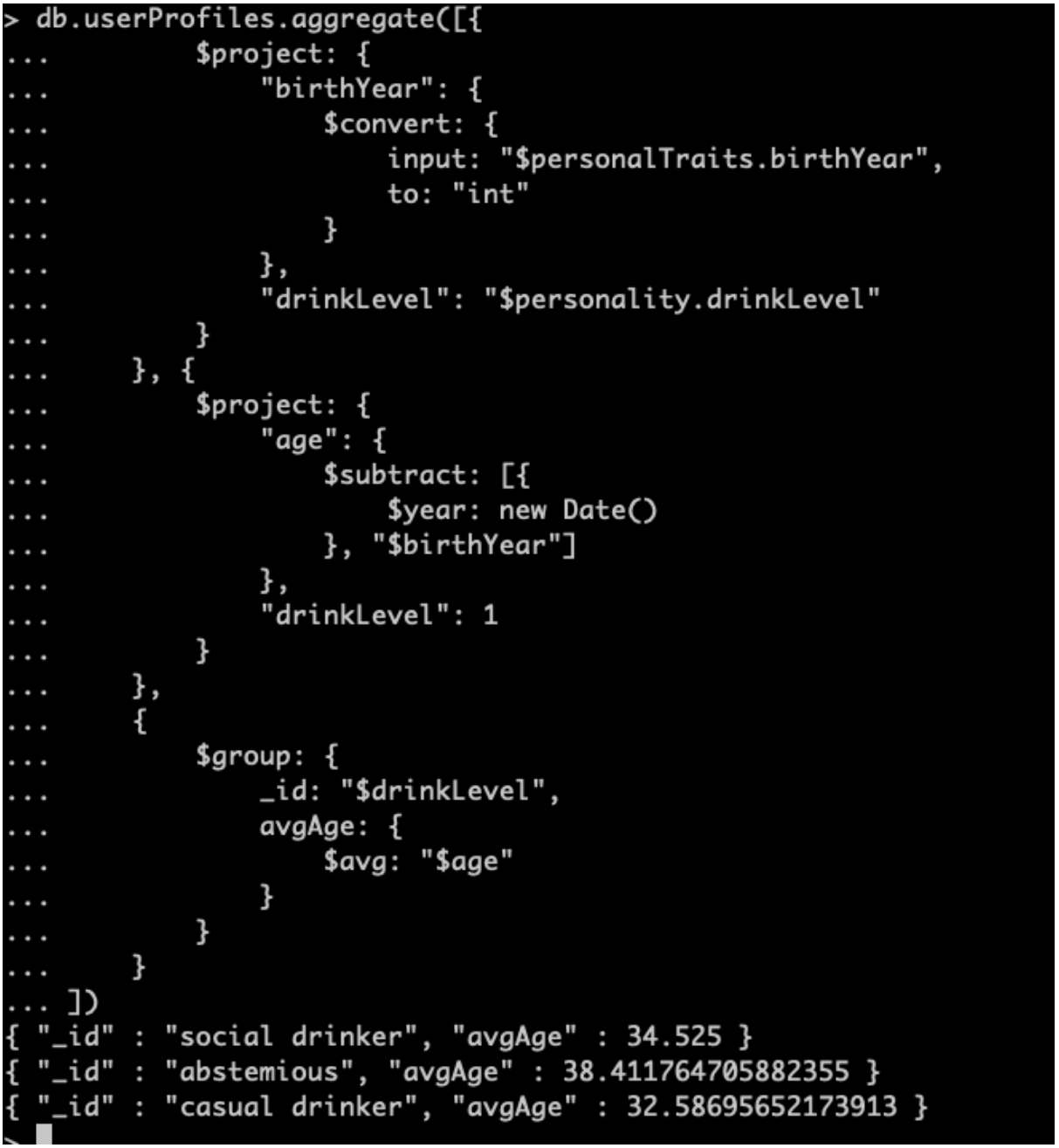
SELECT avg(cast(rating_place as float)) AS "place rating",avg(cast(rating_food as float)) AS "food rating",avg(cast(rating_service as float)) AS "service rating" FROM place_ratings WHERE place_name = 'puesto de tacos';

```
cqlsh:fit5137a1_mrdb> SELECT avg(cast(rating_place as float)) AS "place rating",avg(cast(rating_food as float)) AS "food rating",avg(cast(rating_se
os';

 place rating | food rating | service rating
--------------+-------------+----------------
      1.28125 |     1.34375 |         0.9375

(1 rows)
```

C.3.11

```
db.userProfiles.aggregate([
	{
		$project:{
			"birthYear":{
				$convert:{
					input:"$personalTraits.birthYear",
					to:"int"
		}},
			"drinkLevel":"$personality.drinkLevel"
		}
	},
	{
		$project:{
```

```
                "age":{
                    $subtract:[{$year:new Date()},"$birthYear"]
                },
                "drinkLevel":1
            }
        },
        {
            $group:{
                _id:"$drinkLevel",
                avgAge:{
                    $avg:"$age"
                }
            }
        }
])
```



C.3.12

create index fav_cuisine on user_ratings(user_fav_cuisines);

select user_id, place_id, rating_place, rating_food, user_preferences.budget from user_ratings where user_fav_cuisines contains 'Family';

```
cqlsh:fit5137a1_mrdb>
cqlsh:fit5137a1_mrdb> select user_id, place_id, rating_place, rating_food, user_preferences.budget from user_ratings where user_fav_cuisines contains 'Family';

 user_id | place_id | rating_place | rating_food | user_preferences.budget
---------+----------+--------------+-------------+-------------------------
    1019 |   135079 |            0 |           0 |                  medium
    1019 |   132921 |            0 |           0 |                  medium
    1135 |   135042 |            0 |           0 |                     low
    1135 |   135060 |            0 |           0 |                     low
    1135 |   132825 |            0 |           0 |                     low
    1027 |   135042 |            1 |           1 |                     low
    1027 |   135066 |            0 |           0 |                     low
    1007 |   135057 |            1 |           1 |                     low
    1108 |   135074 |            2 |           2 |                  medium
    1135 |   132834 |            0 |           0 |                     low
    1019 |   132830 |            0 |           0 |                  medium
    1027 |   132951 |            1 |           1 |                     low
    1108 |   132723 |            2 |           2 |                  medium
    1027 |   135052 |            1 |           1 |                     low
    1027 |   132937 |            1 |           1 |                     low
    1027 |   132872 |            1 |           1 |                     low
    1009 |   135060 |            1 |           1 |                  medium
    1007 |   135086 |            0 |           0 |                     low
```

C.3.13

```
db.userProfiles.aggregate([
        {
                $match:{
                        favCuisines :{$regex:"Japanese"},
                        "personalTraits.maritalStatus":"single"
                }
        },
        {
                $project:{
                        favCuisines:1,
                        "marital":"$personalTraits.maritalStatus",
                        "ambience":"$preferences.ambience"
                }
        },
        {
                $group:{
                        _id:"$ambience",
                        count:{$sum:1}
                }
        },
        {
                $sort:{
                        count:-1
                }
        },
        {
                $limit:3
        }
])
```

```
> db.userProfiles.aggregate([{
...         $match: {
...             favCuisines: {
...                 $regex: "Japanese"
...             },
...             "personalTraits.maritalStatus": "single"
...         }
...     },
...     {
...         $project: {
...             favCuisines: 1,
...             "marital": "$personalTraits.maritalStatus",
...             "ambience": "$preferences.ambience"
...         }
...     },
...     {
...         $group: {
...             _id: "$ambience",
...             count: {
...                 $sum: 1
...             }
...         }
...     },
...     {
...         $sort: {
...             count: -1
...         }
...     },
...     {
...         $limit: 3
...     }
... ])
{ "_id" : "family", "count" : 3 }
{ "_id" : "friends", "count" : 2 }
{ "_id" : "solitary", "count" : 1 }
```

C.3.14

```
db.placeProfiles.aggregate([
	{
		$project:{
			cuisines:{$split:["$cuisines",","]},
			_id:1
		}
	},
```

```
        {
                $unwind:"$cuisines"
        },
        {
                $group:{
                        _id:null,
                        uniqueCuisines:{
                                $addToSet:"$cuisines"
                        }
                }
        },
        {
                $project:
                        {uniqueCuisines:1}
        }
])
```

```
> db.placeProfiles.aggregate([{
...         $project: {
...             cuisines: {
...                 $split: ["$cuisines", ","]
...             },
...             _id: 1
...         }
...     },
...     {
...         $unwind: "$cuisines"
...     },
...     {
...         $group: {
...             _id: null,
...             uniqueCuisines: {
...                 $addToSet: "$cuisines"
...             }
...         }
...     },
...     {
...         $project: {
...             uniqueCuisines: 1
...         }
...     }
... ])
{ "_id" : null, "uniqueCuisines" : [ "International", "Contemporary", "", "Japanese", "Family", "Bar", " Contemporary", "Regional", "Armenian", "Seafood", "Pizzeria"
, "Cafeteria", " Mexican", "Burgers", " Mediterranean", " Pizzeria", "Mexican", " Game", "Vietnamese", "Bakery", "American", "Breakfast-Brunch", " Cafeteria", "Bar_P
ub_Brewery", "Chinese", "Italian", " Fast_Food", " Bar_Pub_Brewery", "Fast_Food", " Burgers", "Cafe-Coffee_Shop" ] }
```

C.3.15

```
db.placeProfiles.aggregate(
        {
                $project:{
                        placeName:1,
                        cuisines:1,
                        Serving:{
                                $cond:{
                                        if:{
                                                $in:["$cuisines",["Mexico"]]
                                        },
                                        then:"Mexican Served",
                                        else:"Mexican Not Served"
                                }
                        }
                }
        }
)
```

```
> db.placeProfiles.aggregate(
... {
... $project:{
... placeName:1,
... cuisines:1,
... Serving:{
... $cond:{
... if:{
... $in:["$cuisines",["Mexico"]]
... },
... then:"Mexican Served",
... else:"Mexican Not Served"
... }
... }
... }
... }
... )
{ "_id" : "132560", "cuisines" : "Regional", "placeName" : "puesto de gorditas", "Serving" : "Mexican Not Served" }
{ "_id" : "132561", "cuisines" : "", "placeName" : "cafe ambar", "Serving" : "Mexican Not Served" }
{ "_id" : "132564", "cuisines" : "", "placeName" : "churchs", "Serving" : "Mexican Not Served" }
{ "_id" : "132572", "cuisines" : "Cafeteria", "placeName" : "Cafe Chaires", "Serving" : "Mexican Not Served" }
{ "_id" : "132583", "cuisines" : "American", "placeName" : "McDonalds Centro", "Serving" : "Mexican Not Served" }
{ "_id" : "132584", "cuisines" : "Mexican", "placeName" : "Gorditas Dona Tota", "Serving" : "Mexican Not Served" }
{ "_id" : "132594", "cuisines" : "Mexican", "placeName" : "tacos de barbacoa enfrente del Tec", "Serving" : "Mexican Not Served" }
{ "_id" : "132608", "cuisines" : "Mexican", "placeName" : "Hamburguesas La perica", "Serving" : "Mexican Not Served" }
{ "_id" : "132609", "cuisines" : "Fast_Food", "placeName" : "Pollo_Frito_Buenos_Aires", "Serving" : "Mexican Not Served" }
{ "_id" : "132613", "cuisines" : "Mexican", "placeName" : "carnitas_mata", "Serving" : "Mexican Not Served" }
{ "_id" : "132626", "cuisines" : "Italian", "placeName" : "la perica hamburguesa", "Serving" : "Mexican Not Served" }
{ "_id" : "132630", "cuisines" : "Mexican", "placeName" : "palomo tec", "Serving" : "Mexican Not Served" }
{ "_id" : "132654", "cuisines" : "", "placeName" : "Carnitas Mata Calle 16 de Septiembre", "Serving" : "Mexican Not Served" }
{ "_id" : "132660", "cuisines" : "", "placeName" : "carnitas mata calle Emilio Portes Gil", "Serving" : "Mexican Not Served" }
{ "_id" : "132663", "cuisines" : "Mexican", "placeName" : "tacos abi", "Serving" : "Mexican Not Served" }
{ "_id" : "132665", "cuisines" : "Mexican", "placeName" : "TACOS CORRECAMINOS", "Serving" : "Mexican Not Served" }
{ "_id" : "132667", "cuisines" : "Armenian", "placeName" : "little pizza Emilio Portes Gil", "Serving" : "Mexican Not Served" }
{ "_id" : "132668", "cuisines" : "Mexican", "placeName" : "TACOS EL GUERO", "Serving" : "Mexican Not Served" }
{ "_id" : "132706", "cuisines" : "Mexican", "placeName" : "Gorditas Dona Tota", "Serving" : "Mexican Not Served" }
{ "_id" : "132715", "cuisines" : "Mexican", "placeName" : "tacos de la estacion", "Serving" : "Mexican Not Served" }
Type "it" for more
```

Additional Query:

1. Check the average place rating for restaurants where there is no parking arrangement, so that MonR can see whether there is a connection between the availability of parking and the rating results.

   select avg(cast(rating_place AS Double)) AS averagePlaceRating,parking_arrangements from place_ratings where parking_arrangements = 'none';

```
cqlsh:fit5137a1_mrdb> select avg(cast(rating_place AS Double)) AS averagePlaceRating,parking_arrangements from place_rat
ings where parking_arrangements = 'none' ALLOW FILTERING;

 averageplacerating | parking_arrangements
--------------------+----------------------
            1.20321 |                 none

(1 rows)

Warnings :
```

2. Regarding the query results shown above, the average rating is quite low, therefore we decide to further display the percentage and count of restaurants in each parkingArrangements, so that the MonR can know whether they should make an improvement for the transportation for their restaurants.

```
db.placeProfiles.aggregate([
        {
                $project:{
                        parkingArragements:1,
                }
        },
        {
                $group:{
                        _id:"$parkingArragements",
```

```
                count:{
                        $sum:1
                }
        }
},
{
        $project:{
                count:1,
                parkingArragements:1,
                percentage:{
                        $concat:[
                        {
                                $substr:
                                [{
                                $multiply:[
                        {
                                $divide:["$count",{"$literal":
db.placeProfiles.count()}]
                        },100]
                        },0,5]
                        },"%"
                        ]
                }
                }
        }
])
```

```
> db.placeProfiles.aggregate([{
...         $project: {
...             parkingArragements: 1,
...         }
...     },
...     {
...         $group: {
...             _id: "$parkingArragements",
...             count: {
...                 $sum: 1
...             }
...         }
...     },
...     {
...         $project: {
...             count: 1,
...             parkingArragements: 1,
...             percentage: {
...                 $concat: [{
...                     $substr: [{
...                         $multiply: [{
...                             $divide: ["$count", {
...                                 "$literal": db.placeProfiles.count()
...                             }]
...                         }, 100]
...                     }, 0, 5]
...                 }, "%"]
...             }
...         }
...     }
... ])
{ "_id" : "public", "count" : 16, "percentage" : "12.21%" }
{ "_id" : "valet parking", "count" : 3, "percentage" : "2.290%" }
{ "_id" : "none", "count" : 66, "percentage" : "50.38%" }
{ "_id" : "yes", "count" : 46, "percentage" : "35.11%" }
```

3. Display the percentage of each individual unique payment method and sorting it in descending order, so that MonR can decide whether current payment method in stores should be improved

```
db.userProfiles.aggregate([
	{
		$project:{
			"favPaymentMethod":{
				$split:["$favPaymentMethod",","]
			}
		}
	},
	{
		$unwind:"$favPaymentMethod"
	},
```

```
	{
		$group:{
			_id:"$favPaymentMethod",
			count:{$sum:1}
		}
	},
	{
		$sort:{
			"count":-1
		}
	},
	{
		$project:{
			favPaymentMethod:1,
			count:1,
			percentage:{
				$concat:[
				{
					$substr:
					[{
					$multiply:[
				{
					$divide:["$count",{"$literal":
db.userProfiles.count()}]
				},100]
				},0,5]
				},"%"
				]
			}
		}
	}
])
```

4. For each group of users in different employment groups, show the number of people in different budget and sort the output in a descending order by the number of people, so that the MonR can know the main budget condition for each employment group and the number of people of it, so that MonR can adjust their selling strategy according to their main customer group's employment as well as the budget.

```
db.userProfiles.aggregate([
	{
		$project:{
			employement:"$otherDemographics.employment",
			budget:"$preferences.budget"
		}
	},
	{
		$group:{
			_id:{employement:"$employement",budget:"$budget"},
			numberOfUers:{
				$sum:1
			}
		}
	},
	{
		$sort:{
			numberOfUers:-1
		}
	}
])
```

```
> db.userProfiles.aggregate([{
...         $project: {
...             employement: "$otherDemographics.employment",
...             budget: "$preferences.budget"
...         }
...     },
...     {
...         $group: {
...             _id: {
...                 employement: "$employement",
...                 budget: "$budget"
...             },
...             numberOfUers: {
...                 $sum: 1
...             }
...         }
...     },
...     {
...         $sort: {
...             numberOfUers: -1
...         }
...     }
... ])
{ "_id" : { "employement" : "student", "budget" : "medium" }, "numberOfUers" : 70 }
{ "_id" : { "employement" : "student", "budget" : "low" }, "numberOfUers" : 34 }
{ "_id" : { "employement" : "professional", "budget" : "medium" }, "numberOfUers" : 15 }
{ "_id" : { "employement" : "student", "budget" : "high" }, "numberOfUers" : 4 }
{ "_id" : { "employement" : "", "budget" : "" }, "numberOfUers" : 4 }
{ "_id" : { "employement" : "unemployed", "budget" : "medium" }, "numberOfUers" : 2 }
{ "_id" : { "employement" : "", "budget" : "medium" }, "numberOfUers" : 2 }
{ "_id" : { "employement" : "Mexican", "budget" : "FALSE" }, "numberOfUers" : 2 }
{ "_id" : { "employement" : "Mexican", "budget" : "medium" }, "numberOfUers" : 1 }
{ "_id" : { "employement" : "student", "budget" : "" }, "numberOfUers" : 1 }
{ "_id" : { "employement" : "", "budget" : "high" }, "numberOfUers" : 1 }
{ "_id" : { "employement" : "working-class", "budget" : "medium" }, "numberOfUers" : 1 }
{ "_id" : { "employement" : "Family", "budget" : "low" }, "numberOfUers" : 1 }
```

5. display the number of people in each drink level in descending order, so that the MonR can decide whether to include drinks in restaurants, this query is in addition to the previous query about the average age of each drinking level, since apart from the consideration about the age, Number of customers in each level is also important to be considered regarding the profits.

```
db.userProfiles.aggregate([
        {
                $project:{
                        "personality.drinkLevel":1
                }
        },
        {
                $group:{
                        _id:"$personality.drinkLevel",
                        count:{$sum:1}
                }
        },
        {
                $sort:{
                        "count":-1
```

```
            }
        }
])
```

```
> db.userProfiles.aggregate([{
...         $project: {
...             "personality.drinkLevel": 1
...         }
...     },
...     {
...         $group: {
...             _id: "$personality.drinkLevel",
...             count: {
...                 $sum: 1
...             }
...         }
...     },
...     {
...         $sort: {
...             "count": -1
...         }
...     }
... ])
{ "_id" : "abstemious", "count" : 51 }
{ "_id" : "casual drinker", "count" : 47 }
{ "_id" : "social drinker", "count" : 40 }
```

Compound Index Creation:
For userProfiles:
create a compound index on:
[personality.drinkLevel,
otherDemographics.employment,favCuisines,preferences.budget]

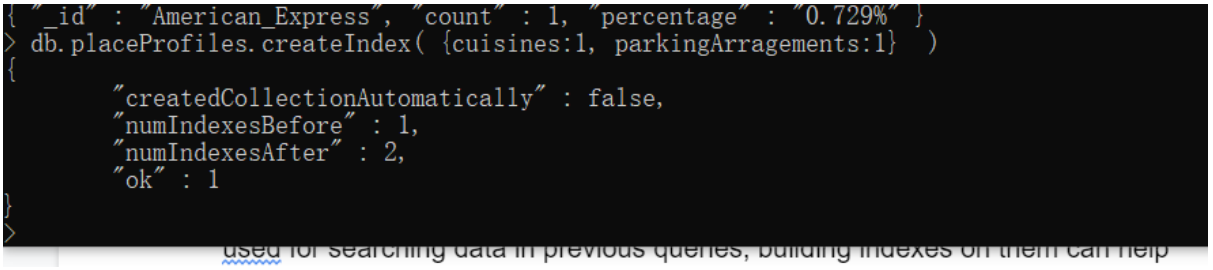
For placeProfiles:
[cuisines,parkingArragements]

Reason for selection:
Firstly, reason for creating index in MongoDB can help improve the efficiency of finding the data, this is because that without indexing, MongoDB have to iterate the whole collection to look for the wanted one, but with indexing can help reduce the number of docs system has to look through.

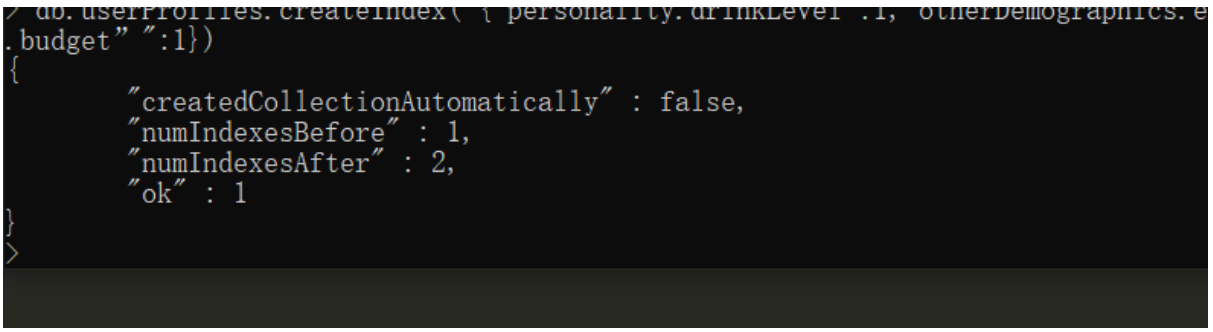
Next, the reason to select these two compound indexes is that the fields in the list is used for searching data in previous queries, building indexes on them can help improve the efficiency.

Syntax:
db.placeProfiles.createIndex({

```
  cuisines: 1,
  parkingArragements: 1
})
```




```
db.userProfiles.createIndex({
  "personality.drinkLevel": 1,
  "otherDemographics.employment": 1,
  "favCuisines": 1,
  "preferences.budget": 1
})
```



Task C.4. Summary Reports.

Summary Reports

1. Explanation of each database works.

Relational Database must define a schema before adding records to a database. It stores data in tables which are composed of columns and each column stores one type of data. this model is based on relational algebra and following ACID (Atomicity, Consistency, Isolation, Durability) principles

Document-Oriented Database and Column-Oriented Database belong to non-relational databases. They follow BASE (Basically Available, Soft state, Eventual consistency) principles and they are Schema-less data models.

For Document-Oriented Database, it is inherently a subclass of the key-value database. It does not need to predefine structure on the stored data, because each document can have its own structure. It stores data in collection and each collection stores one or many documents whose structure is key-value.

For Column-Oriented Database, it stores data tables as sections of columns of data, It also organizes data in key-value pairs. Data values are identified by row identifier, column name, and time stamp. All of the super columns which consist of a group of logical related columns are grouped together to be a column family. Column families are stored in a keyspace.

2. Comparison and real-word example

	Relational Database	Document-Oriented Database	Column-Oriented Database
strengths	1. Structural independence different tables could have different structures, changing the structure does not impact data access. 2. Tabular view improves conceptual simplicity It is earlier for designer to design, implementation, management, and use	1. flexible structure the structure of individual documents does not have to be consistent. Even large volumes of unstructured data can be accommodated in the database. 2. High scalability, it's easier to integrate new	1. High performance on aggregation queries it is faster in query processing and aggregation operations 2. Efficient storage and data compression it have true scalability and fast data loading

	the database 3. Query is based on SQL. SQL is easy in the relational database approach	information, the new information only needs to be included in just a few datasets in a document store 3. store data efficiently Key-value model improves storage efficiency.	for Big Data and/or partitioning 3.Simplified administration and configuration It is easier to administer the system because of fairly simple systems administration
weaknesses	1. Higher cost Such as join operation would need more physical storage consumption 2. Isolated Databases Complex relational database systems can lead to these databases becoming islands of information where the information cannot be shared easily from one large system to another. 3. Structured Limits When you design the database, you have to specify the amount of data you can fit into a field.	1. no relationship There is no relationship support such as foreign key 2. Complex programming For complex jobs you need Map-Reduce	1. inefficiently retrieving and join Queries with table joins can reduce high performance 2. no transactions Transactions are to be avoided or just not supported 3. Complex programming Effective partitioning/indexing schemes can be difficult to design
real-world example	Telenor has been using MySQL for fixed IP management since 2003	Adobe uses MongoDB to store petabytes of data in the large-scale content repositories	Facebook originally uses Cassandra to manage their Inbox Search

		underpinning the Experience Manager	

3. Database Selection

Overall, we would suggest using MongoDB to maintain the business requirements. The reason is that MongoDB can better help you analyze your data and do the decision making which is more important to your company. Besides, it also enables you to write in data with flexible data structures.

To be more specific, unlike social media companies such as Instagram and Facebooks, regarding the nature of a management institute, the main purpose to maintain a DB for you is to make queries to make the decisions instead of updating and scaling the db frequently. Moreover, the number of restaurants and users will not be increasing too frequently(like every seconds of a day).

Therefore, we would recommend MongoDB instead of Cassandra because while MongoDB can do the query more easily, it can also provide you with sufficient scalability and flexible data structures in your documents. And in contrast, Cassandra is more suitable for those companies who require huge data availability and scalability(updating the data very frequently).
Below is the detailed explanation:

(1) MongoDB is a better choice for frequently querying:

Firstly, MongoDB has built in aggregation but cassandra not, which means that MongoDB may be a better choice when it comes to a huge demand on small or medium-sized data traffic.

Besides, MongoDB has high-quality indexes functionality which makes query very efficient. But Cassandra's secondary index is less efficient and its queries are limited to single columns and equality comparisons. To be more specific, although Cassandra can make locating the data in a fast speed when querying on primary keys, the speed and efficiency will drop significantly when querying on secondary indexes. This is because that Cassandra partitions its data by its primary key, therefore if query on the PK, it can easily find and know where to get the data, however, when using secondary indexes, it has to go through every node.

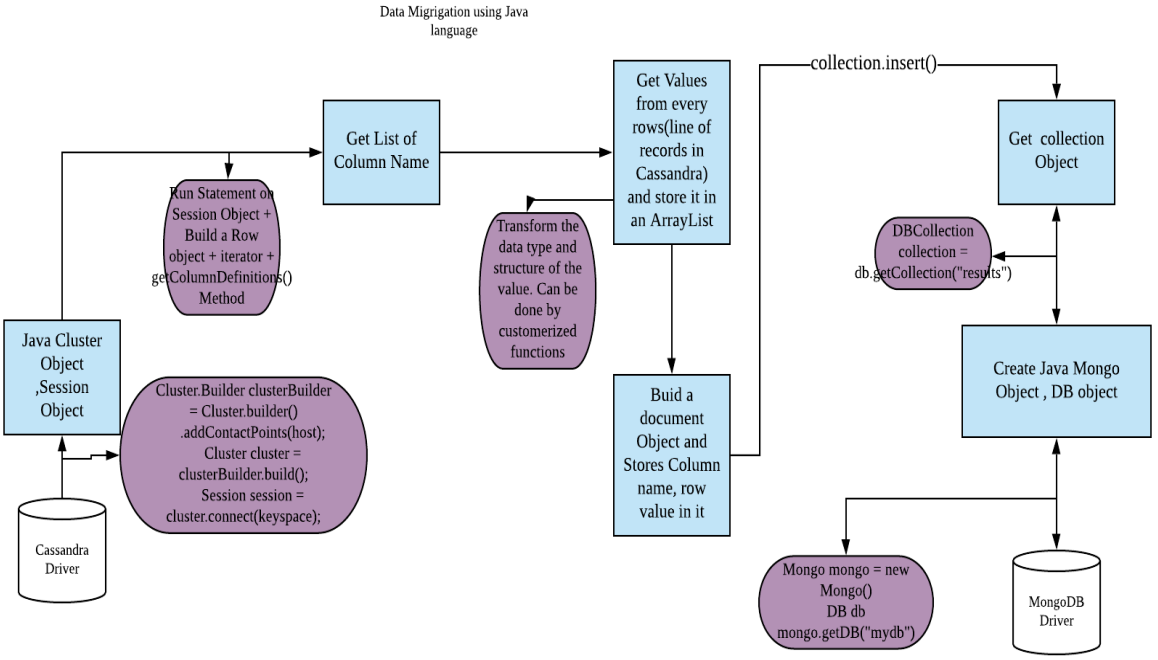
(2) MongoDB is better for your Data Model demands:

As a restaurant-management company, your data structure may have multiple forms.

From this aspect, MongoDB can represent any kind of object structures which can have properties or even be nested for multiple levels because of the nature of document-oriented databases. In comparison,Cassandra is a little more traditional than MongoDB although better than relational databases.Therefore, if you need a rich data model, MongoDB may be the better solution.

(3) Cassandra is more suitable for high availability and scalability which is not your main purpose.

4. Data Migration:



As shown in the flow chart above, documents in Cassandra can be migrated to MongoDB by using Mongo/Cassandra Driver in either Java/Python language. This flowchart shows the steps and some brief technical details within both steps.
To be concluded, there are steps including:

1. build objects for Cassandra and MongoDB including cluster objects, session objects ,statement and row objects for Cassandra; Mongo objects, DB objects and collection objects for MongoDB.
2. After we got the needed objects, we can first use the session object and the statement object to get data from Cassandra, then run the methods to get the column names from Cassandra as well as the values for each row in it by using the iterator. Column names and each row will then be stored in suitable data structure. Within this step, a customer function will be needed to adjust the data value such as filtering the data and convert to the data type we wanted.
3. After getting all data, storing it in a document object and inserting it to the collection object of MongoDB.

Task C.5. Connecting to Drivers

1. MongoDB Drivers

step:

1. make sure the files of sample data and .py in the same dictionary(under same file)
2. open terminal to start mongoDB server
3. open a new terminal window and use python3 mongodb_driver.py to run the mongodb script

```
#!/usr/bin/env python3

import pymongo
import json

client = pymongo.MongoClient('localhost', 27017)
# C.1.1
# create a data database
db = client['FIT5137A1MRDB']
# C.1.2
# create 2 collections
placeProfiles = db['placeProfiles']
userProfiles = db['userProfiles']
# C.1.3 a
with open('userProfile.json') as user_json:
   user_data = json.load(user_json)
insert_result = userProfiles.insert_many(user_data)
insert_result.acknowledged
# C.1.3 b
with open('placeProfiles.json') as place_json:
   place_data = json.load(place_json)
insert_result = placeProfiles.insert_many(place_data)
insert_result.acknowledged
# C.1.4 b
openingHours = db['openingHours']


def format_data(path):
   arr = []
   with open(path, "r") as files:
       next(files)
       for file in files:
           file = file.replace('\n', '')
           columns = file.split(",")
           obj = {
```

```
                "placeID": columns[0],
                "hours": columns[1],
                "days": columns[2]
            }
            arr.append(obj)


#     print(arr)
    return arr


def find(find_result):
    for result in find_result:
        print(result)


insert_result = openingHours.insert_many(format_data("openingHours.csv"))
insert_result.acknowledged

openingHours.aggregate([{
    "$group": {
        "_id": "$placeID",
        "openingHours": {
            "$addToSet": {
                "hours": "$hours",
                "days": "$days"
            }
        }
    }
}, {
    "$project": {
        "_id": 1,
        "openingHours": 1
    }
}, {
    "$out": "openingHours"
}])

placeProfiles.aggregate([{
    "$lookup": {
        "from": "openingHours",
        "localField": "_id",
        "foreignField": "_id",
        "as": "openInfo"
    }
```

```
}, {
   "$unwind": {
       "path": "$openInfo",
       "preserveNullAndEmptyArrays": True
   }
}, {
   "$project": {
       "_id": 1,
       "acceptedPaymentModes": 1,
       "address": 1,
       "cuisines": 1,
       "location": 1,
       "parkingArragements": 1,
       "placeFeatures": 1,
       "placeName": 1,
       "openingHours": "$openInfo.openingHours"
   }
}, {
   "$out": "placeProfiles"
}])

# C.2.1
placeProfiles.insert_one({
   "_id":
   "70000",
   "acceptedPaymentModes":
   "any",
   "address": {
       "city": "San Luis Potosi",
       "country": "Mexico",
       "state": "SLP",
       "street": "Carretera Central Sn"
   },
   "cuisines":
   "Mexican, Burgers",
   "parkingArragements":
   "none",
   "placeFeatures": {
       "accessibility": "completely",
       "alcohol": "No_Alcohol_Served",
       "area": "open",
       "dressCode": "informal",
       "franchise": "f",
       "otherServices": "Internet",
       "price": "medium",
```

```
        "smokingArea": "not permitted"
    },
    "placeName":
    "Taco Jacks",
    "openingHours": [{
        "hours": "09:00-20:00;",
        "days": "Mon;Tue;Wed;Thu;Fri;"
    }, {
        "hours": "12:00-18:00;",
        "days": "Sat;Sun;"
    }]
})

userProfiles.update_one({"_id": "1108"}, [{
    "$set": {
        "favCuisines": {
            "$replaceOne": {
                "input": "$favCuisines",
                "find": "Fast_Food, ",
                "replacement": ""
            }
        },
        "favPaymentMethod": {
            "$replaceOne": {
                "input": "$favPaymentMethod",
                "find": "cash",
                "replacement": "debit_cards"
            }
        }
    }
}, {
    "$set": {
        "favCuisines": {
            "$replaceOne": {
                "input": "$favCuisines",
                "find": "Fast_Food",
                "replacement": ""
            }
        }
    }
}])

userProfiles.delete_one({"_id": "1063"})

# C.3.1
```

```
print(
   f'How many users are there in the database? :
{userProfiles.estimated_document_count()}'
)

# C.3.2
print(
   f'How many places are there in the database? :
{placeProfiles.estimated_document_count()}'
)

# C.3.7
find_result = userProfiles.find(
   {
       "otherDemographics.employment": "student",
       "preferences.budget": "medium"
   }, {"_id": 1})
print("----C.3.7----")
find(find_result)

# C.3.8
Q8 = db['Q8']
userProfiles.aggregate([{
   "$match": {
       "favCuisines": {
           "$regex": "Bakery"
       }
   }
}, {
   "$project": {
       "_id": 0,
       "user": "$_id"
   }
}, {
   "$merge": {
       "into": "Q8"
   }
}])

placeProfiles.aggregate([{
   "$match": {
       "cuisines": {
           "$regex": "Bakery"
       }
   }
```

```
}, {
   "$project": {
       "cuisines": 1,
       "Restaruant": "$_id",
       "_id": 0
   }
}, {
   "$merge": {
       "into": "Q8"
   }
}])

find_result = Q8.find({})
print("----C.3.8----")
find(find_result)

# C.3.9
find_result = placeProfiles.aggregate([{
   "$match": {
       "cuisines": "International",
       "openingHours.days": "Sun;",
       "openingHours.hours": {
           "$nin": ["0:00-0:00;"]
       }
   }
}, {
   "$project": {
       "placeName": 1
   }
}])
print("----C.3.9----")
find(find_result)

# C.3.11
from datetime import datetime

find_result = userProfiles.aggregate([{
   "$project": {
       "birthYear": {
           "$convert": {
               "input": "$personalTraits.birthYear",
               "to": "int"
           }
       },
       "drinkLevel": "$personality.drinkLevel"
```

```
    }
}, {
    "$project": {
        "age": {
            "$subtract": [{
                "$year": datetime.now()
            }, "$birthYear"]
        },
        "drinkLevel": 1
    }
}, {
    "$group": {
        "_id": "$drinkLevel",
        "avgAge": {
            "$avg": "$age"
        }
    }
}])
print("----C.3.11----")
find(find_result)

# C.3.14
find_result = placeProfiles.aggregate([{
    "$project": {
        "cuisines": {
            "$split": ["$cuisines", ","]
        },
        "_id": 1
    }
}, {
    "$unwind": "$cuisines"
}, {
    "$group": {
        "_id": None,
        "uniqueCuisines": {
            "$addToSet": "$cuisines"
        }
    }
}, {
    "$project": {
        "uniqueCuisines": 1
    }
}])
print("----C.3.14----")
find(find_result)
```

```
# C.3.15
find_result = placeProfiles.aggregate([
{
 "$project":{
  "placeName":1,
  "cuisines":1,
  "Serving":{
   "$cond":{
    "if":{
     "$in":["$cuisines",["Mexico"]]
    },
    "then":"Mexican Served",
    "else":"Mexican Not Served"
   }
  }
 }
}
])
print("----C.3.15----")
find(find_result)

# Additional Query 2
find_result = placeProfiles.aggregate([
{
 "$project":{
  "parkingArragements":1,
 }
},
{
 "$group":{
  "_id":"$parkingArragements",
  "count":{
   "$sum":1
  }
 }
},
{
 "$project":{
  "count":1,
  "parkingArragements":1,
  "percentage":{
   "$concat":[
   {
    "$substr":
```

```
    [{
    "$multiply":[
   {
    "$divide":["$count",{"$literal": placeProfiles.estimated_document_count()}]
   },100]
   },0,5]
   },"%"
   ]
  }
  }
}
])
print("----Additional Query 2----")
find(find_result)

# Additional Query 3
find_result = userProfiles.aggregate([{
   "$project": {
       "favPaymentMethod": {
           "$split": ["$favPaymentMethod", ","]
       }
   }
}, {
   "$unwind": "$favPaymentMethod"
}, {
   "$group": {
       "_id": "$favPaymentMethod",
       "count": {
           "$sum": 1
       }
   }
}, {
   "$sort": {
       "count": -1
   }
}, {
   "$project": {
       "favPaymentMethod": 1,
       "count": 1,
       "percentage": {
           "$concat": [{
               "$substr": [{
                   "$multiply": [{
                       "$divide": [
                           "$count", {
```

```
                                "$literal":
                                userProfiles.estimated_document_count()
                            }
                        ]
                    }, 100]
                }, 0, 5]
            }, "%"]
        }
    }
}])
print("----Additional Query 3----")
find(find_result)

# Additional Query 4
find_result = userProfiles.aggregate([{
    "$project": {
        "employement": "$otherDemographics.employment",
        "budget": "$preferences.budget"
    }
}, {
    "$group": {
        "_id": {
            "employement": "$employement",
            "budget": "$budget"
        },
        "numberOfUers": {
            "$sum": 1
        }
    }
}, {
    "$sort": {
        "numberOfUers": -1
    }
}])
print("----Additional Query 4----")
find(find_result)

# Additional Query 5
find_result = userProfiles.aggregate([{
    "$project": {
        "personality.drinkLevel": 1
    }
}, {
    "$group": {
        "_id": "$personality.drinkLevel",
```

```
        "count": {
            "$sum": 1
        }
    }
}, {
    "$sort": {
        "count": -1
    }
}])
print("----Additional Query 5----")
find(find_result)
```