ECS789P - SEMI-STRUCTURED DATA AND ADVANCED DATA MODELLING - 2021/22

MongoDB Design and Implementation

GROUP 21: POST-GRADUATE

Sanjay Ramesh : 210811700
Ritika Gupta : 210487158
Aditya Ronak Shah: 210841431
Bineeta Kachhap : 210619025

DESCRIPTION:

The document specification is to the design and implement a mongodb database for an airline company. The system is required to store information about the flights it runs, journey bookings, passengers and air crew. The document has been prepared as part of coursework-2 for ECS789P Semi-Structured Data and Advanced Data Modelling - 2021/22

ASSUMPTIONS:

- 6 document collection has been created as Employees, Customers, Bookings, Flight Schedule, Employee Code, Planes. The details have been mentioned in the Schema section.
- Each collection has a primary identification key. For e.g., Employees collection has employee_id, Bookings has booking_id etc.
- Four types of employees are considered for this coursework: Pilots, Cabin Crew, Maintenance Staff, Booking Clerk.
- For the purpose of project and as experiment the day wise salary and hours of work has been added as 1: N embedded array within the employee detail.
- For this course the reference fields are not added in the validator schema.
- A standard salary has been maintained per employee type.
- All employees and customer have one address each.
- The flight schedules and travel plan are considered for 16th and 17th November 2021. For this course work, there are only two days that the airline was active for the month of November.
- All payments are done by Card.
- All the booking done by customer were done on 1st November 2021.
- Each plane has maximum capacity of 10. Flying range for the planes were measured in km.
- All the date time fields are in ISO format.
- Each Airport has a cost that comprises of its maintenance, fuel cost, cost for flights standby. For this project, the maintenance cost is born at the departure city for flights. Each flight at the time of departure halts for 2 hours each.

SOLUTION/DELIVERABLES:

Following files are part of deliverable that specifies the coursework.

- Gr21_210811700_210487158_210841431_210619025_Report.pdf :
- Group21_schemaValidator_cw2.js
- Group21_dbsetup_cw2.js
- Group21_queries_cw2.js
- Group21_profiler_cw2.js
- Group21_explain_indexes_cw2.js
- Data file: employees_CW2.json , bookings_CW2.json, customers_CW2.json, flightSchedule_CW2.json, employeeCode_CW2.json, planes_CW2.json.

The data files were shared among the group member for loading the data. The pdf file contains most of the query executed, in case of an issues the .js files are provided as well.

SCHEMA:

Following collection were created.

Employees:

The collection contains the list of current and old employees, along with their contact details. The field joining date and last working date are of in ISO format and contains employment details. For current employees the last working date is empty. To showcase the understanding of 1:N embedded array, the field work schedule has been created. The work schedule contains details of work per day and salary received per day. The work time is calculated in hours.

Field	Type	Comment
employee_id*	string	custom id for each employees.
firstName*	string	
lastName*	string	
email*	string	
contactNumber*	double	
address*	object	1:1 embedded array
joiningDate*	date	
lastWorkingDate*	date	
employeeType*	object	1:1 embedded array
employeeWorkSchedule*	array[1]	1:N embedded object

address	
buildingNo*	double
street*	string
city*	string
zipcode*	double

employeeWorkSchedule	
date*	date
hours*	double
salary*	double

employeeType	
eType*	string
fitToFly	bool

Customers:

The collection has list of customers, along with the contact details. The address is 1:1 array with building No, streets city and zip code stored in it.

Field	Type	Comments
customer_id*	string	custom id for each customer
title*	string	
firstName*	string	
lastName*	string	
email	string	
contactNumber*	double	
address*	object	1:1 embedded array

Bookings:

The booking collection stores the details of each booking done. The collection contains the customer who made the booking along with the flight details. The payment has been done by card and payment amount for each booking done by the customer.

Field	Type	Comments
book_id*	string	custom id for each booking
customer_id*	string	reference to customer_id from Customer collections
flight_id*	array	list of reference flight_id from Flight Schedule collection
paymentMode*	string	only card
bookingDate*	date	has been taken as 1st November 2021
bookingAmount*	double	

Airports:

The Airport contains the list of functioning aiport available in various cities, the airportCost contains details about the maintenance cost, refuel and hourly cost.

Field	type	Comment
cityID	string	
city	string	
airportCost	object	1:1 embedded array contains details about maintenance cost, refuel and hourly cost

airportCost	
refuelCost*	double
maintainanceCost*	double
hourlyStopRate*	double

Planes:

The Planes collection has details of all the planes that are available.

Field	Type	Comments
plane_id*	string	custom id for each type of planes
make*	string	
model*	string	
serviceTime	double	
state*	object	1:1 embedded array
capacity*	double	each has capacity of 10
flyingRange*	double	
unit*	string	accepted values of km, miles

state	
status	string
repairCost	double

List of accepted values:
working
in-repair
up-graded

Employee code:

The employee code collection stored the type of Employees available in the airline. The collection has been created to store the details of each employee type and salaries associated for the post.

Field	Type	Comments
code_id*	string	custom_id for each employee type
employeeType*	string	Pilots, Cabin Crew, Maintenance Staff, Booking Clerk
standardSalary*	double	

List of accepted values

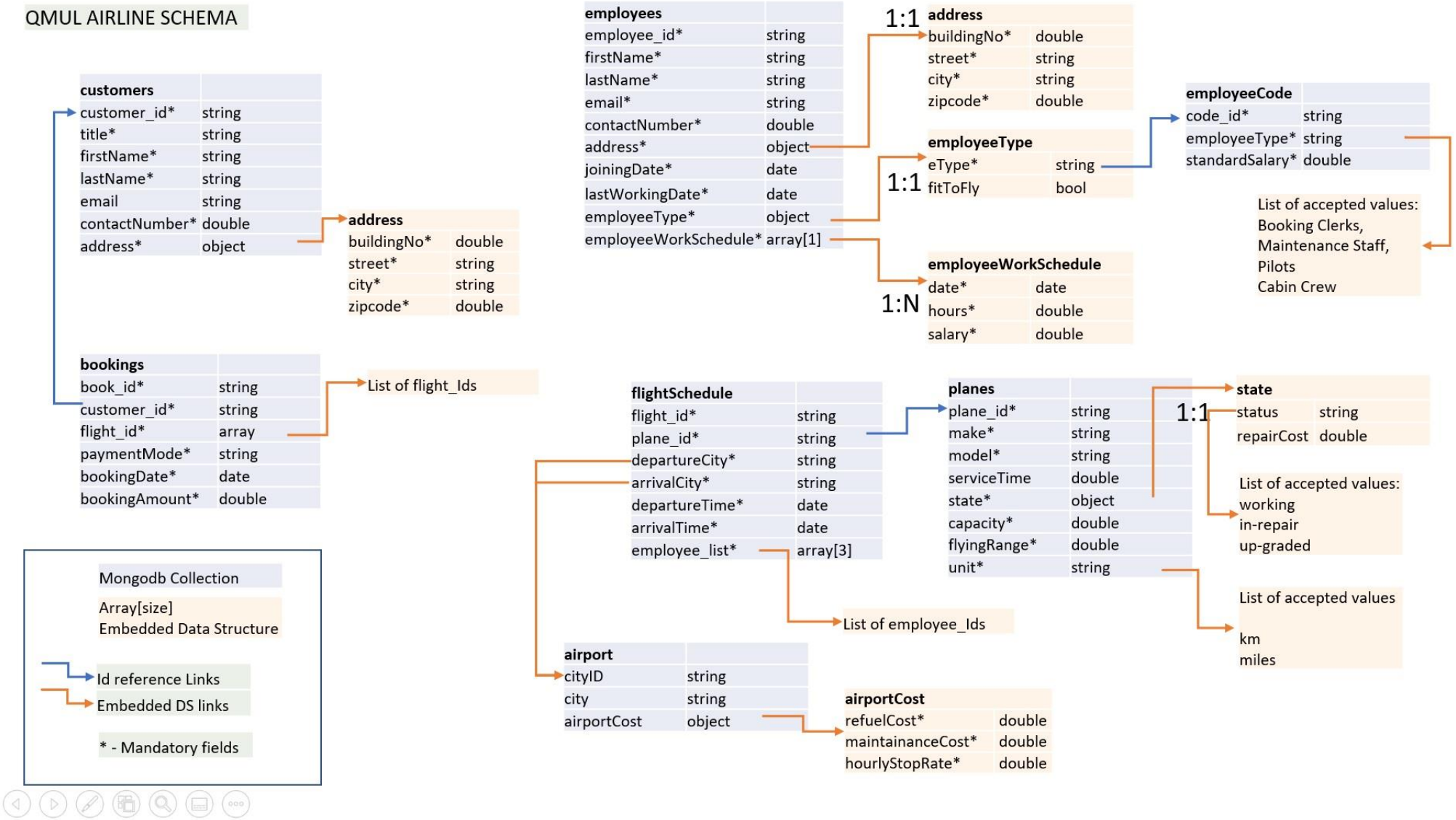
km
miles

Flight Schedule:

The flight schedule collection stores the list of all the flights that are schedules to fly. Each flight has details of the plane being used and the departure and arrival. time and city.

Field	Type	Comments
flight_id*	string	custom id for each flight schedules
plane_id*	string	reference to plane to be used
departure_city_id*	string	departure city
arrival_city_id*	string	arrival City
departureTime*	date	
arrivalTime*	date	
employee_list*	array[3]	

The below image outlines the complete schema and structure of the database:



SCHEMA VALIDATOR FOR EACH COLLECTION

1. *Schema validator for EmployeeCode Collection:*

```
db.createCollection("employeeCode", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "code_id", "employeeType", "standardSalary" ],
                properties: {
                        code_id: { bsonType: "string" },
                        employeeType: { bsonType: "string", enum:["Booking Clerks", "Maintenance Staff", "Pilot", "Cabin Crew"]},
                        standardSalary: { bsonType: "double" }
                }
        }
        }})
```

2. *Schema validator for Planes Collection:*

```
db.createCollection("planes", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "plane_id", "make", "model", "state", "capacity", "flyingRange", "unit"],
                properties: {
                        plane_id: { bsonType: "string" },
                        make: { bsonType: "string" },
                        model: { bsonType: "string" },
                        serviceTime: { bsonType: "double" },
                        state: {
                                bsonType: "object",
                                required: [ "status"],
                                properties:{
                                        status: { bsonType: "string", enum : ["working","in-repair","upgraded"] },
                                        repairCost: { bsonType: "double" }
                                }
                        },
                        capacity: { bsonType: "double" },
                        flyingRange: { bsonType: "double" },
```

```
                unit: { enum : ["km","miles"], bsonType: "string" }
            }
        }
        }})
```

3. *Schema validator for Bookings Collection:*

```
db.createCollection("bookings", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "book_id", "flight_id", "customer_id", "paymentMode", "bookingDate", "bookingAmount"],
                properties: {
                        book_id: { bsonType: "string" },
                        flight_id: { bsonType: "array", minItems: 1, items: { bsonType: "string" }},
                        customer_id: { bsonType: "string" },
                        paymentMode: { bsonType: "string" },
                        bookingDate: { bsonType: "date" },
                        bookingAmount: { bsonType: "double" }
                }
        }
        }})
```

4. *Schema validator for Customers Collection:*

```
db.createCollection("customers", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "customer_id", "firstName", "lastName", "contactNumber","address"],
                properties: {
                        customer_id: { bsonType: "string" },
                        title: { bsonType: "string" },
                        firstName: { bsonType: "string" },
                        lastName: { bsonType: "string" },
                        email: { bsonType: "string" },
                        contactNumber: { bsonType: "double" },
                        address: {
                                bsonType: "object",
                                required: [ "buildingNo", "street", "city" ,"zipcode"],
                                properties:{
                                        buildingNo: { bsonType: "double" },
```

```
                                        street: { bsonType: "string" },
                                        city: { bsonType: "string" },
                                        zipcode: { bsonType: "string" }
                                }
                        }
                }
        }} })
```

5. *5. Schema for Airports Collection*

```
db.createCollection("airports", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "city_id", "city", "airportCost"],
                properties: {
                        city_id: { bsonType: "string" },
                        city: { bsonType: "string" },
                        airportCost: {
                                bsonType: "object",
                                required: [ "refuelCost", "maintainanceCost", "hourlyStopRate"],
                                properties:{
                                        refuelCost: { bsonType: "double" },
                                        maintainanceCost: { bsonType: "double" },
                                        hourlyStopRate: { bsonType: "double" }
                                }
                        }
                }
        }
        }
})
```

6. *Schema validator for FlightSchedule Collection:*

```
db.createCollection("flightSchedule", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "flight_id", "plane_id", "departureCity", "arrivalCity", "departureTime", "arrivalTime","employee_list"],
                properties: {
                        flight_id: { bsonType: "string" },
                        plane_id: { bsonType: "string" },
                        departureCity: { bsonType: "string" },
```

```
                arrivalCity: { bsonType: "string" },
                departureTime: { bsonType: "date" },
                arrivalTime: { bsonType: "date" },
                employee_list: {
                        bsonType: "array",
                        minItems: 3,
                        items: { bsonType: "string" }
                }
        }
}} })
```

7. *Schema validator for Employees Collection:*

```
db.createCollection("employees", {
        validator: {
        $jsonSchema: {
                bsonType: "object",
                required: [ "employee_id", "firstName", "lastName", "email", "contactNumber","address","joiningDate", "employeeType"],
                properties: {
                        employee_id: { bsonType: "string" },
                        firstName: { bsonType: "string" },
                        lastName: { bsonType: "string" },
                        email: { bsonType: "string" },
                        contactNumber: { bsonType: "double" },
                        address: {
                                bsonType: "object",
                                required: [ "buildingNo", "street", "city" ,"zipcode"],
                                properties:{
                                        buildingNo: { bsonType: "double" },
                                        street: { bsonType: "string" },
                                        city: { bsonType: "string" },
                                        zipcode: { bsonType: "string" }
                                }
                        },
                        joiningDate: { bsonType: "date" },
                        lastWorkingDate: { bsonType: "date" },
                        employeeWorkSchedule: {
                                bsonType: "array",
                                items: {
                                        bsonType: "object",
```

```
                                        required: [ "date", "hours", "salary"],
                                        properties:{
                                                date: { bsonType: "date" },
                                                hours: { bsonType: "double" },
                                                salary: { bsonType: "double" }
                                        }
                                }
                        },
                        employeeType: {
                                bsonType: "object",
                                required: ["eType"],
                                properties:{
                                        eType: { bsonType: "string" },
                                        fitToFly: { bsonType: "bool" }
                                }
                        }
                }
        }} })
```

SET OF 12 QUERIES

Following set of queries have been written to showcase understanding about the queries and commands. Concepts and mongodb expressions were covered in the below queries: Screenshot with few details have been attached for the queries.

- showcase of aggregate and find utility method
- use of aggregate pipeline with $project, $match, $group, $unwind, $sort, $limit, $lookup.
- usage of utility methods line $gt, $eq, $concat, $where, $year, $month, $dayOfMonth, $sum, $subtract, $exist, $toString, $dateToString, $addToSet, $size, $isArray

1. *List of existing Employees name and contact number:*

Query by checking if last working day fields is empty.

```
db.employees.aggregate(

        {$match:{"lastWorkingDate":{$exists:false}}},
        {$project: {
        "_id":0,
        name: {$concat: [ "$firstName", " ", "$lastName"]},
        number: "$contactNumber",
        address: {$concat: [ {$toString:"$address.buildingNo"}, " ", "$address.street",", ","$address.city", "-", "$address.zipcode"]}}}
)
```

```
> db.employees.aggregate(
... {$match:{"lastWorkingDate":{$exists:false}}},
... {$project: {
... "_id":0,
... name: {$concat: [ "$firstName", " ", "$lastName"]},
... number: "$contactNumber",
... address: {$concat: [ {$toString:"$address.buildingNo"}, " ", "$address.street",", ","$address.city", "-", "$address.zipcode"]}}}
... )
{ "name" : "Ritika Gupta", "number" : 446785674563, "address" : "34 Abbey Road, Birmingham-E56GHB" }
{ "name" : "Aditya Ronak Shah", "number" : 446785674564, "address" : "56 Brick Lane, London-E57GHB" }
{ "name" : "Sanjay Ramesh", "number" : 446785674565, "address" : "98 Oxford Street, Liverpool-E58GHB" }
{ "name" : "Bineeta Kachhap", "number" : 446785674566, "address" : "43 Carnaby Street, London-E59GHB" }
{ "name" : "Antra Tripathi", "number" : 446785674567, "address" : "65 Piccadilly, Belfast-E60GHB" }
{ "name" : "Yunqing Gou", "number" : 446785674568, "address" : "72 Baker Street, Birmingham-E61GHB" }
{ "name" : "Yifeng Lin", "number" : 446785674569, "address" : "12 Downing Street , London-E62GHB" }
{ "name" : "Chuanning Liu", "number" : 446785674570, "address" : "78 Harley Street, Belfast-E63GHB" }
{ "name" : "Yunlong Mu", "number" : 446785674571, "address" : "43 Old Compton Street, Liverpool-E64GHB" }
{ "name" : "Weitian Ran", "number" : 446785674572, "address" : "27 Abbey Road, Belfast-E65GHB" }
{ "name" : "Yashi Gupta", "number" : 446785674573, "address" : "80 Brick Lane, Birmingham-E66GHB" }
{ "name" : "Alexander Sachkov", "number" : 446785674574, "address" : "20 Oxford Street, London-E67GHB" }
{ "name" : "Shiva Archith Siddhartha", "number" : 446785674575, "address" : "34 Carnaby Street, London-E68GHB" }
{ "name" : "Kang Simon", "number" : 446785674576, "address" : "59 Piccadilly, Birmingham-E69GHB" }
{ "name" : "Tong Liu", "number" : 446785674577, "address" : "30 Baker Street, Liverpool-E70GHB" }
{ "name" : "Rabia Alam", "number" : 446785674578, "address" : "23 Downing Street , Belfast-E71GHB" }
```

2. *Customers with doctoral degree:*

Query using find method and checking the title as Dr.

db.customers.find({title:"Dr."},{customer_id:true, firstName:true, lastName:true, email:true}).sort({firstName:1}).limit(10)

```
> db.customers.find({title:"Dr."},{_id:false, customer_id:true, firstName:true, lastName:true, email:true}).sort({firstName:1}).limit(10)
{ "customer_id" : "CUS001", "firstName" : "Anthony", "lastName" : "Stockman", "email" : "AnthonyStockman@customer.com" }
{ "customer_id" : "CUS004", "firstName" : "Emmanouil", "lastName" : "Benetos", "email" : "EmmanouilBenetos@passenger.com" }
{ "customer_id" : "CUS007", "firstName" : "Georgios", "lastName" : "Tzimiropoulos", "email" : "GeorgiosTzimiropoulos@customer.com" }
{ "customer_id" : "CUS002", "firstName" : "Jesus", "lastName" : "Requena-Carrion", "email" : "JesusRequenaCarrion@passenger.com" }
{ "customer_id" : "CUS005", "firstName" : "Jialun", "lastName" : "Hu", "email" : "JialunHu@customer.com" }
{ "customer_id" : "CUS008", "firstName" : "Qianni", "lastName" : "Zhang", "email" : "QianniZhang@passenger.com" }
{ "customer_id" : "CUS003", "firstName" : "Sukhpal", "lastName" : "Singh Gill", "email" : "SukhpalSinghGill@customer.com" }
{ "customer_id" : "CUS009", "firstName" : "Usman", "lastName" : "Naeem", "email" : "UsmanNaeem@customer.com" }
```

3. *Booking ids with more than one flight.*

Query by checking the flight list is more than 1

```
db.bookings.find({$where: "this.flight_id.length > 1"}).forEach(function(booking){
        print("Booking Ids: "+ booking.book_id+ ", "+"Flight List: "+ booking.flight_id);
});
```

```
> db.bookings.find({$where: "this.flight_id.length > 1"}).forEach(function(booking){
... print("Booking Ids: "+ booking.book_id+ ", "+"Flight List: "+ booking.flight_id);
... });
Booking Ids: BK010, Flight List: FL005,FL006,FL007
Booking Ids: BK011, Flight List: FL005,FL006,FL007
Booking Ids: BK013, Flight List: FL003,FL004
Booking Ids: BK014, Flight List: FL003,FL004
Booking Ids: BK015, Flight List: FL003,FL004
Booking Ids: BK016, Flight List: FL008,FL009
Booking Ids: BK017, Flight List: FL008,FL009
Booking Ids: BK018, Flight List: FL008,FL009
```

4. Available flight for dates in November (16/17)

Query by fetching all flight schedule and group by date.

```
db.flightSchedule.aggregate(
        { $project: {
          year: {$year: "$departureTime"},
          month: {$month: "$departureTime"},
          dayOfMonth: {$dayOfMonth: "$departureTime"},
          departureCity: "$departureCity",
          arrivalCity: "$arrivalCity"
        }},
        { $group: {
          _id: { date: {$concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] }},
          flights: { $push: { $concat: [ "$departureCity", " - ", "$arrivalCity",]}}}})
```

```
> db.flightSchedule.aggregate(
... { $project: {
...   year: {$year: "$departureTime"},
...   month: {$month: "$departureTime"},
...   dayOfMonth: {$dayOfMonth: "$departureTime"},
...   departureCity: "$departureCity",
...   arrivalCity: "$arrivalCity"
... }},
... { $lookup: {from: "airports", localField: "departureCity", foreignField: "city_id", as: "departureAirport"}},
... { $unwind: "$departureAirport"},
... { $lookup: {from: "airports", localField: "arrivalCity", foreignField: "city_id", as: "arrivalAirport"}},
... { $unwind: "$arrivalAirport"},
... { $group: {
...   _id: { date: {$concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] }},
...   flights: { $push: { $concat: [ "$departureAirport.city", " - ", "$arrivalAirport.city",]}}
... }
...   }
... )
{ "_id" : { "date" : "2021-11-17" }, "flights" : [ "London - Belfast", "Belfast - London", "Sheffield - Liverpool", "Liverpool - Southamton", "Southamton -
Sheffield" ] }
{ "_id" : { "date" : "2021-11-16" }, "flights" : [ "London - Manchester", "Manchester - London", "Sheffield - Manchester", "Manchester - Glasgow", "Glasgow
- Sheffield" ] }
>
```

5. *Top 5 employee with highest salary.*

Done by aggregating employees collection and fetching all data for employeeWorkSchedule and adding up the employeeWorkSchedule.salary field.

```
db.employees.aggregate(
	{$match: {employeeWorkSchedule: {$exists: true}}},
	{ $project: {
		name : {$concat:["$firstName"," ","$lastName"]},
		employeeWorkSchedule: 1 }},
	{$unwind: "$employeeWorkSchedule"},
	{ $group: {
		_id: "$name", salary: {$sum:"$employeeWorkSchedule.salary"}}},
	{ $sort: {"salary": -1}},
	{ $limit: 5}
)
```

```
> db.employees.aggregate(
... {$match: {employeeWorkSchedule: {$exists: true}}},
... { $project: {
... name : {$concat:["$firstName"," ","$lastName"]},
... employeeWorkSchedule: 1 }},
... {$unwind: "$employeeWorkSchedule"},
... { $group: {
... _id: "$name", salary: {$sum:"$employeeWorkSchedule.salary"}}},
... { $sort: {"salary": -1}},
... { $limit: 5}
... )
{ "_id" : "Sanjay Ramesh", "salary" : 1200 }
{ "_id" : "Aditya Ronak Shah", "salary" : 1200 }
{ "_id" : "Ritika Gupta", "salary" : 1200 }
{ "_id" : "Bineeta Kachhap", "salary" : 1200 }
{ "_id" : "Kang Simon", "salary" : 900 }
```

6. *List of employees group by city.*

Done by aggregating employees collection and fetching all data and grouping by address.city.

```
db.employees.aggregate(
	{$match: {address: {$exists: true}}},
	{ $project: {
		name : {$concat:["$firstName"," ","$lastName"]},
		address: 1 }},
	{ $group: {
		_id: "$address.city",  employeeList : {$push:"$name"} }}
)
```

```
> db.employees.aggregate(
... {$match: {address: {$exists: true}}},
... { $project: {
... name : {$concat:["$firstName"," ","$lastName"]},
... address: 1 }},
... { $group: {
... _id: "$address.city",  employeeList : {$push:"$name"} }}
... )
{ "_id" : "Birmingham", "employeeList" : [ "Ritika Gupta", "Yunqing Gou", "Yashi Gupta", "Kang Simon", "Remi Gul Bahar", "Tanishq Verma" ] }
{ "_id" : "Liverpool", "employeeList" : [ "Sanjay Ramesh", "Yunlong Mu", "Tong Liu", "Temesgen Daniel Teklebrhan" ] }
{ "_id" : "London", "employeeList" : [ "Aditya Ronak Shah", "Bineeta Kachhap", "Yifeng Lin", "Alexander Sachkov", "Shiva Archith Siddhartha", "Aditya Dubey"
, "David Stumbra" ] }
{ "_id" : "Belfast", "employeeList" : [ "Antra Tripathi", "Chuanning Liu", "Weitian Ran", "Rabia Alam" ] }
>
```

7. *Top 10 oldest employees by joining date(The find query has been written to do further for Indexes and explain utility method.)*

Done by find method with applying sort to the joining date, the find method has been used for further analysis of indexes. The aggregate method matched all data and sorts by joining date.

```
db.employees.find({"employee_id":{$exists:false}},{employee_id:true, firstName:true, lastName:true, joiningDate:true}).sort({joiningDate:1}).limit(10)
```

or

```
db.employees.aggregate(
        {$match: {joiningDate: {$exists: true}}},
        { $project: {
                name : {$concat:["$firstName"," ","$lastName"]},
                joiningDate: 1 }},
        { $sort: {"joiningDate": 1}},
        { $limit: 10})
```

```
> db.employees.aggregate(
... {$match: {joiningDate: {$exists: true}}},
... { $project: {
... name : {$concat:["$firstName"," ","$lastName"]},
... joiningDate: 1 }},
... { $sort: {"joiningDate": 1}},
... { $limit: 10}
... )
{ "_id" : ObjectId("61bf69523eaf4a068173c3e7"), "joiningDate" : ISODate("1998-06-24T00:00:00Z"), "name" : "Temesgen Daniel Teklebrhan" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3d8"), "joiningDate" : ISODate("1999-09-06T00:00:00Z"), "name" : "Antra Tripathi" }
{ "_id" : ObjectId("61bf69523eaf4a068173c3e8"), "joiningDate" : ISODate("2003-08-25T00:00:00Z"), "name" : "Tanishq Verma" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3dc"), "joiningDate" : ISODate("2005-10-13T00:00:00Z"), "name" : "Yunlong Mu" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3dd"), "joiningDate" : ISODate("2006-06-14T00:00:00Z"), "name" : "Weitian Ran" }
{ "_id" : ObjectId("61bf69523eaf4a068173c3e3"), "joiningDate" : ISODate("2007-06-20T00:00:00Z"), "name" : "Rabia Alam" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3d7"), "joiningDate" : ISODate("2008-06-08T00:00:00Z"), "name" : "Bineeta Kachhap" }
{ "_id" : ObjectId("61bf69523eaf4a068173c3e6"), "joiningDate" : ISODate("2009-06-23T00:00:00Z"), "name" : "David Stumbra" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3d4"), "joiningDate" : ISODate("2010-06-05T00:00:00Z"), "name" : "Ritika Gupta" }
{ "_id" : ObjectId("61bf69513eaf4a068173c3da"), "joiningDate" : ISODate("2010-06-11T00:00:00Z"), "name" : "Yifeng Lin" }
>
```

8. *Customers who travelled on 16/11/2021*

Used aggregate method to match all data in flight schedule collection for 16th Nov and then looked up bookings collection to match all the flight and booking_ids and customer_id. With the result of customer_id looked up customers collection to match all the customer.

```
db.flightSchedule.aggregate({
        $project: {
                year: {$year: "$departureTime"},
                month: {$month: "$departureTime"},
                dayOfMonth: {$dayOfMonth: "$departureTime"},
                flight: "$flight_id"
        }},
        { $match:{$and: [{"year":{$eq:2021}}, {"month":{$eq:11}}, {"dayOfMonth":{$eq:16}} ]}},
        { $group: {
                _id: { date: {$concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] }},
                flights: { $push: "$flight"}}},
        { $unwind: "$flights"},
        { $lookup: {from: "bookings", localField: "flights", foreignField: "flight_id", as: "bookings"}},
        { $unwind: "$bookings"},
        { $group: { _id: "$bookings.customer_id" }},
        { $lookup: {from: "customers", localField: "_id", foreignField: "customer_id", as: "customers"}},
        { $unwind: "$customers"},
        {
        $project: {
                _id: "$_id",
                customer: {$concat:["$customers.title"," ","$customers.firstName"," ","$customers.lastName"]}}})
```

```
> db.flightSchedule.aggregate({
... $project: {
... year: {$year: "$departureTime"},
... month: {$month: "$departureTime"},
... dayOfMonth: {$dayOfMonth: "$departureTime"},
... flight: "$flight_id"
... }},
... { $match:{$and: [{"year":{$eq:2021}}, {"month":{$eq:11}}, {"dayOfMonth":{$eq:16}} ]}},
... { $group: {
... _id: { date: {$concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] }},
... flights: { $push: "$flight"}}},
... { $unwind: "$flights"},
... { $lookup: {from: "bookings", localField: "flights", foreignField: "flight_id", as: "bookings"}},
... { $unwind: "$bookings"},
... { $group: { _id: "$bookings.customer_id" }},
... { $lookup: {from: "customers", localField: "_id", foreignField: "customer_id", as: "customers"}},
... { $unwind: "$customers"},
... {
... $project: {
... _id: "$_id",
... customer: {$concat:["$customers.title"," ","$customers.firstName"," ","$customers.lastName"]}
... }}
... )
{ "_id" : "CUS005", "customer" : "Dr. Jialun Hu" }
{ "_id" : "CUS003", "customer" : "Dr. Sukhpal Singh Gill" }
{ "_id" : "CUS001", "customer" : "Dr. Anthony Stockman" }
{ "_id" : "CUS004", "customer" : "Dr. Emmanouil Benetos" }
{ "_id" : "CUS006", "customer" : "Proff Norman Fenton" }
{ "_id" : "CUS008", "customer" : "Dr. Qianni Zhang" }
{ "_id" : "CUS010", "customer" : "Mr. Ercüment Ilhan" }
{ "_id" : "CUS007", "customer" : "Dr. Georgios Tzimiropoulos" }
{ "_id" : "CUS002", "customer" : "Dr. Jesus Requena-Carrion" }
{ "_id" : "CUS011", "customer" : "Mr. Iacopo Ghinassi" }
{ "_id" : "CUS009", "customer" : "Dr. Usman Naeem" }
{ "_id" : "CUS013", "customer" : "Mr. Woody Bayliss" }
>
```

9. *Total Earning of the November Month.*

Used aggregate method to sum up all the payment done for the month of November from booking collection and sum up all the salary given out from the customer collection and sum up any ongoing maintenance and repair work. The profit was calculated as total payment - salary given - maintenance work.

```
db.bookings.aggregate(
	{ $group:{
		_id: "$bookingDate",
		customerPayment: {$sum: "$bookingAmount"}
	}},
	{ $lookup: {from: "employees", pipeline: [{$match: {employeeWorkSchedule: {$exists: true}}}], as: "employees"} },
	{ $unwind: "$employees" },
	{ $group:{
		_id: "$customerPayment",
		totalSalary:{ $sum: {$sum:"$employees.employeeWorkSchedule.salary"}}}},
	{ $project:{
		_id: "Profit",
		profitEarned: {$subtract:["$_id","$totalSalary"]}}},
	{ $lookup: {from: "planes", pipeline: [{$match: {"state.status":{$in:["in-repair","upgraded"]}}}], as: "maintainanceCost"}},
```

```
        { $unwind: "$maintainanceCost" },
        { $group:{
                _id: "$profitEarned",
                maintainanceCost:{$sum:"$maintainanceCost.state.repairCost"}}},
        { $project:{
                _id: "$profitEarned",
                profitEarned: {$subtract:["$_id","$maintainanceCost"]}
        }},
        { $lookup: {from: "flightSchedule", pipeline: [{$match: {departureCity: {$exists: true}}}], as: "flightSchedule"}},
        { $unwind: "$flightSchedule" },
        { $lookup: {from: "airports", localField: "flightSchedule.departureCity", foreignField: "city_id", as: "airport"}},
        { $unwind: "$airport"},
        { $group: {
                _id: "$profitEarned",
                maintainanceCost: {$sum: {$add:["$airport.airportCost.refuelCost", "$airport.airportCost.maintainanceCost", {$multiply:["$airport.airportCost.hourlyStopRate",2]} ]}} }},
        { $project:{
                _id: "Profit",
                profitEarned: {$subtract:["$_id","$maintainanceCost"]} }}
)
```

```
> db.bookings.aggregate(
... { $group:{
... _id: "$bookingDate",
... customerPayment: {$sum: "$bookingAmount"}
... }},
... { $lookup: {from: "employees", pipeline: [{$match: {employeeWorkSchedule: {$exists: true}}}], as: "employees"} },
... { $unwind: "$employees" },
... { $group:{
... _id: "$customerPayment",
... totalSalary:{ $sum: {$sum:"$employees.employeeWorkSchedule.salary"}}}},
... { $project:{
... _id: "Profit",
... profitEarned: {$subtract:["$_id","$totalSalary"]}}},
... { $lookup: {from: "planes", pipeline: [{$match: {"state.status":{$in:["in-repair","upgraded"]}}}], as: "maintainanceCost"}},
... { $unwind: "$maintainanceCost" },
... { $group:{
... _id: "$profitEarned",
... maintainanceCost:{$sum:"$maintainanceCost.state.repairCost"}}},
... { $project:{
... _id: "$profitEarned",
... profitEarned: {$subtract:["$_id","$maintainanceCost"]}
... }},
... { $lookup: {from: "flightSchedule", pipeline: [{$match: {departureCity: {$exists: true}}}], as: "flightSchedule"}},
... { $unwind: "$flightSchedule" },
... { $lookup: {from: "airports", localField: "flightSchedule.departureCity", foreignField: "city_id", as: "airport"}},
... { $unwind: "$airport"},
... { $group: {
... maintainanceCost: {$sum: {$add:["$airport.airportCost.refuelCost", "$airport.airportCost.maintainanceCost", {$multiply:["$airpo
Rate",2]} ]}} }},ost: {$sum: {$add:["$airport.airportCost.refuelCost", "$airport.airportCost.maintainanceCost", {$multiply:["$airpo
... { $project:{
... _id: "Profit",
... profitEarned: {$subtract:["$_id","$maintainanceCost"]} }}
... )
{ "_id" : "Profit", "profitEarned" : 41516.03 }
```

10. Employees working on 17/11/2021

Use aggregate method to match all flight schedules and list of employee id associated to the flight for 17th November. Looked up the employee collection to match all the employee_id from the employees collection.

```
db.flightSchedule.aggregate({

        $project: {
                year: {$year: "$departureTime"},
                month: {$month: "$departureTime"},
                dayOfMonth: {$dayOfMonth: "$departureTime"},
                flight: "$flight_id",
                employee_list: "$employee_list"
        }},{
        $match:{ $and: [{"year":{$eq:2021}},{"month":{$eq:11}},{"dayOfMonth":{$eq:17}} ]}},
        { $unwind: "$employee_list" },
        {
        $group: {
                _id: { $concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] },
                employee: { $addToSet: "$employee_list"}}},
        { $unwind: "$employee"},
        { $lookup: {from: "employees", localField: "employee", foreignField: "employee_id", as: "employees"}},
        { $unwind: "$employees" },
        {$project: {"_id":"$_id", Name: {$concat: [ "$employees.firstName", " ", "$employees.lastName"]}, Number: "$employees.contactNumber"}})
```

```
> db.flightSchedule.aggregate({
... $project: {
... year: {$year: "$departureTime"},
... month: {$month: "$departureTime"},
... dayOfMonth: {$dayOfMonth: "$departureTime"},
... flight: "$flight_id",
... employee_list: "$employee_list"
... }},{
... $match:{ $and: [{"year":{$eq:2021}},{"month":{$eq:11}},{"dayOfMonth":{$eq:17}} ]}},
... { $unwind: "$employee_list" },
... {
... $group: {
... _id: { $concat: [ {$toString:"$year"}, "-", {$toString:"$month"},"-", {$toString:"$dayOfMonth"}] },
... employee: { $addToSet: "$employee_list"}}},
... { $unwind: "$employee"},
... { $lookup: {from: "employees", localField: "employee", foreignField: "employee_id", as: "employees"}},
... { $unwind: "$employees" },
... {$project: {"_id":"$_id", Name: {$concat: [ "$employees.firstName", " ", "$employees.lastName"]}, Number: "$employees.contactNumber"}}
... )
{ "_id" : "2021-11-17", "Name" : "Alexander Sachkov", "Number" : 446785674574 }
{ "_id" : "2021-11-17", "Name" : "Ritika Gupta", "Number" : 446785674563 }
{ "_id" : "2021-11-17", "Name" : "Weitian Ran", "Number" : 446785674572 }
{ "_id" : "2021-11-17", "Name" : "Kang Simon", "Number" : 446785674576 }
{ "_id" : "2021-11-17", "Name" : "Tong Liu", "Number" : 446785674577 }
{ "_id" : "2021-11-17", "Name" : "Rabia Alam", "Number" : 446785674578 }
{ "_id" : "2021-11-17", "Name" : "Yunlong Mu", "Number" : 446785674571 }
{ "_id" : "2021-11-17", "Name" : "Bineeta Kachhap", "Number" : 446785674566 }
{ "_id" : "2021-11-17", "Name" : "Shiva Archith Siddhartha", "Number" : 446785674575 }
{ "_id" : "2021-11-17", "Name" : "Sanjay Ramesh", "Number" : 446785674565 }
{ "_id" : "2021-11-17", "Name" : "Aditya Ronak Shah", "Number" : 446785674564 }
{ "_id" : "2021-11-17", "Name" : "Yashi Gupta", "Number" : 446785674573 }
```

11. Passenger List who travelled more than one flight.

From the booking collection get all the flight counts per booking, for any count which was greater than 1, the customer_ids were added to a set and looked up customers collection to get the customer details.

```
db.bookings.aggregate(
        { $project: {
                book_id: 1,
                customer_id: 1,
                flightCount: { $cond: { if: { $isArray: "$flight_id" }, then: { $size: "$flight_id" }, else: "NA"} }}},
        { $project :{
                book_id: 1,
                customer_id: 1,
                flightCount: 1,
                isMore: { $cond: {if: {$gt:["$flightCount",1] }, then: true, else: false}} }},
        { $group: {
                _id: "$isMore", customer_id: { $addToSet: "$customer_id"}}},
        { $match: { _id: true}},
```

```
        { $lookup: {from: "customers", localField: "customer_id", foreignField: "customer_id", as: "customers"}},
        { $unwind: "$customers"},
        {$project: {
                _id: "Flights",
                customer: {$concat:["$customers.firstName"," ","$customers.lastName"]}
        }}
)
```

```
> db.bookings.aggregate(
... { $project: {
... book_id: 1,
... customer_id: 1,
... flightCount: { $cond: { if: { $isArray: "$flight_id" }, then: { $size: "$flight_id" }, else: "NA"} }}},
... { $project :{
... book_id: 1,
... customer_id: 1,
... flightCount: 1,
... isMore: { $cond: {if: {$gt:["$flightCount",1] }, then: true, else: false}} }},
... { $group: {
... _id: "$isMore", customer_id: { $addToSet: "$customer_id"}}},
... { $match: { _id: true}},
... { $lookup: {from: "customers", localField: "customer_id", foreignField: "customer_id", as: "customers"}},
... { $unwind: "$customers"},
... {$project: {
... _id: "Flights",
... customer: {$concat:["$customers.firstName"," ","$customers.lastName"]}
... }}
... )
{ "_id" : "Flights", "customer" : "Ercüment Ilhan" }
{ "_id" : "Flights", "customer" : "Iacopo Ghinassi" }
{ "_id" : "Flights", "customer" : "Yeming Meng" }
{ "_id" : "Flights", "customer" : "Chia-Yen Chiang" }
{ "_id" : "Flights", "customer" : "Sebastian Berns" }
{ "_id" : "Flights", "customer" : "Linjie Xu" }
{ "_id" : "Flights", "customer" : "Abdulrahman Aloraini" }
{ "_id" : "Flights", "customer" : "Katarzyna Maria Adamska" }
>
```

12. Top 5 customers with highest expenditure.

Used aggregate method to sort all the bookings by the payment/booking amount, the top 5 list of customers were looked up in the customers collection for details.

```
db.bookings.aggregate(
        { $project: {
                book_id: 1,
                customer_id: 1,
                bookingAmount: 1}},
        { $sort: {"bookingAmount": -1}},
        { $limit: 5},
        { $lookup: {from: "customers", localField: "customer_id", foreignField: "customer_id", as: "customers"}},
```

```
        { $unwind: "$customers"},
        {$project: {
                _id: "Top5",
                customer: {$concat:["$customers.firstName"," ","$customers.lastName"]},
                address: {$concat: [ {$toString:"$customers.address.buildingNo"}, " ", "$customers.address.street",", ","$customers.address.city", "-", "$customers.address.zipcode"]}}}
)
```

```
> db.bookings.aggregate(
... { $project: {
... book_id: 1,
... customer_id: 1,
... bookingAmount: 1}},
... { $sort: {"bookingAmount": -1}},
... { $limit: 5},
... { $lookup: {from: "customers", localField: "customer_id", foreignField: "customer_id", as: "customers"}},
... { $unwind: "$customers"},
... {$project: {
... _id: "Top5",
... address: {$concat: [ {$toString:"$customers.address.buildingNo"}, " ", "$customers.address.street",", ","$customers.address.city", "-", "$customers.addr
ess.zipcode"]}}}oncat: [ {$toString:"$customers.address.buildingNo"}, " ", "$customers.address.street",", ","$customers.address.city", "-", "$customers.addr
... )
{ "_id" : "Top5", "customer" : "Peiling Yi", "address" : "27 Baker Street, Manchester-E79GHB" }
{ "_id" : "Top5", "customer" : "Parvathy Chittur Subramanianprasad", "address" : "2 Piccadilly, Brighton-E78GHB" }
{ "_id" : "Top5", "customer" : "Elona Shatri", "address" : "921 Carnaby Street, London-E77GHB" }
{ "_id" : "Top5", "customer" : "Saqib Iqbal", "address" : "59 Oxford Street, Brighton-E76GHB" }
{ "_id" : "Top5", "customer" : "Alvaro Ovalle Castañeda", "address" : "82 Brick Lane, Sheffield-E75GHB" }
>
```

13. Travel history of each passenger with date, price, flight and travel time details.

Used aggregate method to project all the customers, from the bookings collection mapped all the flight schedule and from the flight schedule collection mapped all the travel time, destination and arrival details. The details are shows together with $project.

```
db.customers.aggregate(
        { $project: {
                name : {$concat:["$title"," ","$firstName"," ","$lastName"]},
                customer_id: 1 }},
        { $lookup: {from: "bookings", localField: "customer_id", foreignField: "customer_id", as: "bookings"}},
        { $unwind: "$bookings" },
        { $project: {
                name : "$name",
                customer_id: 1,
                flight: "$bookings.flight_id",
                price: "$bookings.bookingAmount"}},
        { $unwind: "$flight"},
```

```
        { $lookup: {from: "flightSchedule", localField: "flight", foreignField: "flight_id", as: "schedule"}},
        { $unwind: "$schedule"},
        { $project: {
                _id: 0,
                name : 1,
                price: 1,
                travel: {$concat:["$schedule.departureCity","-","$schedule.arrivalCity"]},
                travelTime: {$concat:[{ $dateToString: { format: "%Y:%m:%d-%H:%M:%S", date: "$schedule.departureTime"}}," - ",{ $dateToString: { format: "%Y:%m:%d-%H:%M:%S",
date: "$schedule.arrivalTime"}}]}
                }}
)
```

```
> db.customers.aggregate(
... { $project: {
... name : {$concat:["$title"," ","$firstName"," ","$lastName"]},
... customer_id: 1 }},
... { $lookup: {from: "bookings", localField: "customer_id", foreignField: "customer_id", as: "bookings"}},
... { $unwind: "$bookings" },
... { $project: {
... name : "$name",
... customer_id: 1,
... flight: "$bookings.flight_id",
... price: "$bookings.bookingAmount"}},
... { $unwind: "$flight"},
... { $lookup: {from: "flightSchedule", localField: "flight", foreignField: "flight_id", as: "schedule"}},
... { $unwind: "$schedule"},
... { $lookup: {from: "airports", localField: "schedule.departureCity", foreignField: "city_id", as: "departureAirport"}},
... { $unwind: "$departureAirport"},
... { $lookup: {from: "airports", localField: "schedule.arrivalCity", foreignField: "city_id", as: "arrivalAirport"}},
... { $unwind: "$arrivalAirport"},
... { $project: {
... _id: 0,
... name : 1,
... price: 1,
... travelTime: {$concat:[{ $dateToString: { format: "%Y:%m:%d-%H:%M:%S", date: "$schedule.departureTime"}}," - ",{ $dateToString: { format: "%Y:%m:%d-%H:%M
:%S", date: "$schedule.arrivalTime"}}]}ng: { format: "%Y:%m:%d-%H:%M:%S", date: "$schedule.departureTime"}}," - ",{ $dateToString: { format: "%Y:%m:%d-%H:%M
... }}
... )
{ "name" : "Dr. Anthony Stockman", "price" : 1234.56, "travel" : "London-Manchester", "travelTime" : "2021:11:16-08:00:00 - 2021:11:16-11:00:00" }
{ "name" : "Dr. Jesus Requena-Carrion", "price" : 987.23, "travel" : "London-Manchester", "travelTime" : "2021:11:16-08:00:00 - 2021:11:16-11:00:00" }
{ "name" : "Dr. Sukhpal Singh Gill", "price" : 1500.34, "travel" : "London-Manchester", "travelTime" : "2021:11:16-08:00:00 - 2021:11:16-11:00:00" }
{ "name" : "Dr. Emmanouil Benetos", "price" : 1506.49, "travel" : "London-Manchester", "travelTime" : "2021:11:16-08:00:00 - 2021:11:16-11:00:00" }
{ "name" : "Dr. Jialun Hu", "price" : 1639.38, "travel" : "Manchester-London", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Proff Norman Fenton", "price" : 1772.27, "travel" : "Manchester-London", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Dr. Georgios Tzimiropoulos", "price" : 1905.16, "travel" : "Manchester-London", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Dr. Qianni Zhang", "price" : 2038.05, "travel" : "Manchester-London", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Dr. Usman Naeem", "price" : 2170.94, "travel" : "Manchester-London", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Mr. Ercüment Ilhan", "price" : 2303.83, "travel" : "Sheffield-Manchester", "travelTime" : "2021:11:16-07:00:00 - 2021:11:16-08:00:00" }
{ "name" : "Mr. Ercüment Ilhan", "price" : 2303.83, "travel" : "Manchester-Glasgow", "travelTime" : "2021:11:16-09:00:00 - 2021:11:16-11:00:00" }
{ "name" : "Mr. Ercüment Ilhan", "price" : 2303.83, "travel" : "Glasgow-Sheffield", "travelTime" : "2021:11:16-13:00:00 - 2021:11:16-16:00:00" }
{ "name" : "Mr. Iacopo Ghinassi", "price" : 2436.72, "travel" : "Sheffield-Manchester", "travelTime" : "2021:11:16-07:00:00 - 2021:11:16-08:00:00" }
```

PROFILER:

The profiler details are shown in the **Group21_profiler_cw2.js** as the result output contents were very huge.

EXPLAIN AND INDEXES:

Query number 7, 3, 2 are used as an example to showcase the analysis done on usage of indexes. As customers, bookings, employees collection had maximum amount of dummy data the indexes were created on this collection for their respective custom ids.

Query :7

1.. Explain and indexes for employees Collection on employee_id field. The below output has been produced for query no. 7 (Top 10 oldest employees by joining date)

Output a) without indexes:

- The highlighted "stage" : "COLLSCAN" shows that full column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 0 shows that it took less than 0 milli seconds to execute the whole query.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.

```
db.employees.find({"employee_id":{$exists:false}},{employee_id:true, firstName:true, lastName:true, joiningDate:true}).sort({joiningDate:1}).limit(10).explain("executionStats")
{
	"explainVersion" : "1",
	"queryPlanner" : {
		"namespace" : "qmulairline.employees",
		"indexFilterSet" : false,
		"parsedQuery" : {
			"employee_id" : {
				"$not" : {
					"$exists" : true
				}
			}
		},
		"maxIndexedOrSolutionsReached" : false,
		"maxIndexedAndSolutionsReached" : false,
		"maxScansToExplodeReached" : false,
		"winningPlan" : {
			"stage" : "PROJECTION_SIMPLE",
			"transformBy" : {
				"employee_id" : true,
				"firstName" : true,
```

```
                "lastName" : true,
                "joiningDate" : true
            },
            "inputStage" : {
                "stage" : "SORT",
                "sortPattern" : {
                    "joiningDate" : 1
                },
                "memLimit" : 104857600,
                "limitAmount" : 10,
                "type" : "simple",
                "inputStage" : {
                    "stage" : "COLLSCAN",
                    "filter" : {
                        "employee_id" : {
                            "$not" : {
                                "$exists" : true
                            }
                        }
                    },
                    "direction" : "forward"
                }
            }
        },
        "rejectedPlans" : [ ]
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 21,
        "executionStages" : {
            "stage" : "PROJECTION_SIMPLE",
            "nReturned" : 0,
            "executionTimeMillisEstimate" : 0,
            "works" : 24,
            "advanced" : 0,
            "needTime" : 23,
            "needYield" : 0,
```

```
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"transformBy" : {
        "employee_id" : true,
        "firstName" : true,
        "lastName" : true,
        "joiningDate" : true
},
"inputStage" : {
        "stage" : "SORT",
        "nReturned" : 0,
        "executionTimeMillisEstimate" : 0,
        "works" : 24,
        "advanced" : 0,
        "needTime" : 23,
        "needYield" : 0,
        "saveState" : 0,
        "restoreState" : 0,
        "isEOF" : 1,
        "sortPattern" : {
                "joiningDate" : 1
        },
        "memLimit" : 104857600,
        "limitAmount" : 10,
        "type" : "simple",
        "totalDataSizeSorted" : 0,
        "usedDisk" : false,
        "inputStage" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "employee_id" : {
                                "$not" : {
                                        "$exists" : true
                                }
                        }
                },
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 0,
                "works" : 23,
```

```
                    "advanced" : 0,
                    "needTime" : 22,
                    "needYield" : 0,
                    "saveState" : 0,
                    "restoreState" : 0,
                    "isEOF" : 1,
                    "direction" : "forward",
                    "docsExamined" : 21
                }
            }
        }
    },
    "command" : {
        "find" : "employees",
        "filter" : {
            "employee_id" : {
                "$exists" : false
            }
        },
        "limit" : 10,
        "singleBatch" : false,
        "sort" : {
            "joiningDate" : 1
        },
        "projection" : {
            "employee_id" : true,
            "firstName" : true,
            "lastName" : true,
            "joiningDate" : true
        },
        "$db" : "qmulairline"
    },
    "serverInfo" : {
        "host" : "DESKTOP-O2C1LRU",
        "port" : 27017,
        "version" : "5.0.4",
        "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
    },
    "serverParameters" : {
        "internalQueryFacetBufferSizeBytes" : 104857600,
```

```
            "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
            "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
            "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
            "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
            "internalQueryProhibitBlockingMergeOnMongoS" : 0,
            "internalQueryMaxAddToSetBytes" : 104857600,
            "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
```

Created indexes for customer collection for customer_id field.

```
db.employees.getIndexes();
db.employees.createIndex({"employee_id ":1});
db.employees.dropIndex("employee_id_1");
```

Output b) after indexes:

- The highlighted "stage" : "IXSCAN" shows that index column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 0 shows that it took less than 0 milli seconds to execute the whole query.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.

```
db.employees.find({"employee_id":{$exists:false}},{employee_id:true, firstName:true, lastName:true, joiningDate:true}).sort({joiningDate:1}).limit(10).explain("executionStats")
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "qmulairline.employees",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "employee_id" : {
                                "$not" : {
                                        "$exists" : true
                                }
                        }
                },
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "PROJECTION_SIMPLE",
                        "transformBy" : {
```

```
        "employee_id" : true,
        "firstName" : true,
        "lastName" : true,
        "joiningDate" : true
    },
    "inputStage" : {
        "stage" : "SORT",
        "sortPattern" : {
            "joiningDate" : 1
        },
        "memLimit" : 104857600,
        "limitAmount" : 10,
        "type" : "simple",
        "inputStage" : {
            "stage" : "FETCH",
            "filter" : {
                "employee_id" : {
                    "$not" : {
                        "$exists" : true
                    }
                }
            },
            "inputStage" : {
                "stage" : "IXSCAN",
                "keyPattern" : {
                    "employee_id" : 1
                },
                "indexName" : "employee_id_1",
                "isMultiKey" : false,
                "multiKeyPaths" : {
                    "employee_id" : [ ]
                },
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 2,
                "direction" : "forward",
                "indexBounds" : {
                    "employee_id" : [
                        "[null, null]"
```

```
                                ]
                            }
                        }
                    }
                }
            },
            "rejectedPlans" : [ ]
    },
    "executionStats" : {
            "executionSuccess" : true,
            "nReturned" : 0,
            "executionTimeMillis" : 0,
            "totalKeysExamined" : 0,
            "totalDocsExamined" : 0,
            "executionStages" : {
                    "stage" : "PROJECTION_SIMPLE",
                    "nReturned" : 0,
                    "executionTimeMillisEstimate" : 0,
                    "works" : 2,
                    "advanced" : 0,
                    "needTime" : 1,
                    "needYield" : 0,
                    "saveState" : 0,
                    "restoreState" : 0,
                    "isEOF" : 1,
                    "transformBy" : {
                            "employee_id" : true,
                            "firstName" : true,
                            "lastName" : true,
                            "joiningDate" : true
                    },
                    "inputStage" : {
                            "stage" : "SORT",
                            "nReturned" : 0,
                            "executionTimeMillisEstimate" : 0,
                            "works" : 2,
                            "advanced" : 0,
                            "needTime" : 1,
                            "needYield" : 0,
                            "saveState" : 0,
```

```
"restoreState" : 0,
"isEOF" : 1,
"sortPattern" : {
      "joiningDate" : 1
},
"memLimit" : 104857600,
"limitAmount" : 10,
"type" : "simple",
"totalDataSizeSorted" : 0,
"usedDisk" : false,
"inputStage" : {
      "stage" : "FETCH",
      "filter" : {
            "employee_id" : {
                  "$not" : {
                        "$exists" : true
                  }
            }
      },
      "nReturned" : 0,
      "executionTimeMillisEstimate" : 0,
      "works" : 1,
      "advanced" : 0,
      "needTime" : 0,
      "needYield" : 0,
      "saveState" : 0,
      "restoreState" : 0,
      "isEOF" : 1,
      "docsExamined" : 0,
      "alreadyHasObj" : 0,
      "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 0,
            "executionTimeMillisEstimate" : 0,
            "works" : 1,
            "advanced" : 0,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 0,
            "restoreState" : 0,
```

```
                        "isEOF" : 1,
                        "keyPattern" : {
                            "employee_id" : 1
                        },
                        "indexName" : "employee_id_1",
                        "isMultiKey" : false,
                        "multiKeyPaths" : {
                            "employee_id" : [ ]
                        },
                        "isUnique" : false,
                        "isSparse" : false,
                        "isPartial" : false,
                        "indexVersion" : 2,
                        "direction" : "forward",
                        "indexBounds" : {
                            "employee_id" : [
                                "[null, null]"
                            ]
                        },
                        "keysExamined" : 0,
                        "seeks" : 1,
                        "dupsTested" : 0,
                        "dupsDropped" : 0
                    }
                }
            }
        }
    },
    "command" : {
        "find" : "employees",
        "filter" : {
            "employee_id" : {
                "$exists" : false
            }
        },
        "limit" : 10,
        "singleBatch" : false,
        "sort" : {
            "joiningDate" : 1
        },
```

```
            "projection" : {
                "employee_id" : true,
                "firstName" : true,
                "lastName" : true,
                "joiningDate" : true
            },
            "$db" : "qmulairline"
        },
        "serverInfo" : {
            "host" : "DESKTOP-O2C1LRU",
            "port" : 27017,
            "version" : "5.0.4",
            "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
        },
        "serverParameters" : {
            "internalQueryFacetBufferSizeBytes" : 104857600,
            "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
            "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
            "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
            "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
            "internalQueryProhibitBlockingMergeOnMongoS" : 0,
            "internalQueryMaxAddToSetBytes" : 104857600,
            "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
>
>
```


Query:3

2. Explain and indexes for bookings collection on flight_id field. The below output has been produced for query no. 3.(Booking ids with more than one flight.)

Output a) without indexes:

- The highlighted "stage" : "COLLSCAN" shows that full column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 27 shows that it took 27 milli seconds to execute the whole query.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.
- In the executionStats, for executionStages COLLSCAN was performed, and it returned "nReturned" : 8 data based on the filter condition provided in the find query.

```
> db.bookings.find({ $and: [{"flight_id":{$exists:true}}, {$where: "this.flight_id.length > 1"}]}).explain("executionStats")
```

```
{
	"explainVersion" : "1",
	"queryPlanner" : {
		"namespace" : "qmulairline.bookings",
		"indexFilterSet" : false,
		"parsedQuery" : {
			"$where" : {
				"code" : "this.flight_id.length > 1"
			}
		},
		"maxIndexedOrSolutionsReached" : false,
		"maxIndexedAndSolutionsReached" : false,
		"maxScansToExplodeReached" : false,
		"winningPlan" : {
			"stage" : "COLLSCAN",
			"filter" : {
				"$where" : {
					"code" : "this.flight_id.length > 1"
				}
			},
			"direction" : "forward"
		},
		"rejectedPlans" : [ ]
	},
	"executionStats" : {
		"executionSuccess" : true,
		"nReturned" : 8,
		"executionTimeMillis" : 33,
		"totalKeysExamined" : 0,
		"totalDocsExamined" : 23,
		"executionStages" : {
			"stage" : "COLLSCAN",
			"filter" : {
				"$where" : {
					"code" : "this.flight_id.length > 1"
				}
			},
			"nReturned" : 8,
			"executionTimeMillisEstimate" : 27,
```

```
                "works" : 25,
                "advanced" : 8,
                "needTime" : 16,
                "needYield" : 0,
                "saveState" : 1,
                "restoreState" : 1,
                "isEOF" : 1,
                "direction" : "forward",
                "docsExamined" : 23
            }
        },
        "command" : {
            "find" : "bookings",
            "filter" : {
                "$where" : "this.flight_id.length > 1"
            },
            "$db" : "qmulairline"
        },
        "serverInfo" : {
            "host" : "DESKTOP-O2C1LRU",
            "port" : 27017,
            "version" : "5.0.4",
            "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
        },
        "serverParameters" : {
            "internalQueryFacetBufferSizeBytes" : 104857600,
            "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
            "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
            "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
            "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
            "internalQueryProhibitBlockingMergeOnMongoS" : 0,
            "internalQueryMaxAddToSetBytes" : 104857600,
            "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
>
```


Created indexes for bookings collection for flight_id field.

```
db.bookings.getIndexes();
```

```
db.bookings.createIndex({"flight_id":1});
db.bookings.dropIndex("flight_id_1");
```

Output b) after indexes:

- The highlighted "stage" : "IXSCAN" shows that index column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 4 shows that it took 4 milli seconds to execute the whole query.
  There was as significant decrease in the execution time after the indexes as compared to Output A which was "executionTimeMillisEstimate" : 27.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.
- In the executionStats, for executionStages IXSCAN was performed, and it returned "nReturned" : 23 data
  as there are overall 23 documents in bookings collection and then in FETCH stage it returned "nReturned" : 8 data based on the filter condition provided in the find query.

```
> db.bookings.find({ $and: [{"flight_id":{$exists:true}}, {$where: "this.flight_id.length > 1"}]}).explain("executionStats")
{
	"explainVersion" : "1",
	"queryPlanner" : {
		"namespace" : "qmulairline.bookings",
		"indexFilterSet" : false,
		"parsedQuery" : {
			"$and" : [
				{
					"flight_id" : {
						"$exists" : true
					}
				},
				{
					"$where" : {
						"code" : "this.flight_id.length > 1"
					}
				}
			]
		},
		"maxIndexedOrSolutionsReached" : false,
		"maxIndexedAndSolutionsReached" : false,
		"maxScansToExplodeReached" : false,
		"winningPlan" : {
			"stage" : "FETCH",
			"filter" : {
				"$and" : [
					{
						"flight_id" : {
```

```
                                "$exists" : true
                            }
                        },
                        {
                            "$where" : {
                                "code" : "this.flight_id.length > 1"
                            }
                        }
                    ]
                },
                "inputStage" : {
                    "stage" : "IXSCAN",
                    "keyPattern" : {
                        "flight_id" : 1
                    },
                    "indexName" : "flight_id_1",
                    "isMultiKey" : true,
                    "multiKeyPaths" : {
                        "flight_id" : [
                            "flight_id"
                        ]
                    },
                    "isUnique" : false,
                    "isSparse" : false,
                    "isPartial" : false,
                    "indexVersion" : 2,
                    "direction" : "forward",
                    "indexBounds" : {
                        "flight_id" : [
                            "[MinKey, MaxKey]"
                        ]
                    }
                }
            },
            "rejectedPlans" : [ ]
        },
        "executionStats" : {
            "executionSuccess" : true,
            "nReturned" : 8,
            "executionTimeMillis" : 101,
```

```
"totalKeysExamined" : 33,
"totalDocsExamined" : 23,
"executionStages" : {
        "stage" : "FETCH",
        "filter" : {
                "$and" : [
                        {
                                "flight_id" : {
                                        "$exists" : true
                                }
                        },
                        {
                                "$where" : {
                                        "code" : "this.flight_id.length > 1"
                                }
                        }
                ]
        },
        "nReturned" : 8,
        "executionTimeMillisEstimate" : 4,
        "works" : 34,
        "advanced" : 8,
        "needTime" : 25,
        "needYield" : 0,
        "saveState" : 0,
        "restoreState" : 0,
        "isEOF" : 1,
        "docsExamined" : 23,
        "alreadyHasObj" : 0,
        "inputStage" : {
                "stage" : "IXSCAN",
                "nReturned" : 23,
                "executionTimeMillisEstimate" : 0,
                "works" : 34,
                "advanced" : 23,
                "needTime" : 10,
                "needYield" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
```

```
                "keyPattern" : {
                        "flight_id" : 1
                },
                "indexName" : "flight_id_1",
                "isMultiKey" : true,
                "multiKeyPaths" : {
                        "flight_id" : [
                                "flight_id"
                        ]
                },
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 2,
                "direction" : "forward",
                "indexBounds" : {
                        "flight_id" : [
                                "[MinKey, MaxKey]"
                        ]
                },
                "keysExamined" : 33,
                "seeks" : 1,
                "dupsTested" : 33,
                "dupsDropped" : 10
        }
    }
},
"command" : {
        "find" : "bookings",
        "filter" : {
                "$and" : [
                        {
                                "flight_id" : {
                                        "$exists" : true
                                }
                        },
                        {
                                "$where" : "this.flight_id.length > 1"
                        }
                ]
```

```
            },
            "$db" : "qmulairline"
        },
        "serverInfo" : {
            "host" : "DESKTOP-O2C1LRU",
            "port" : 27017,
            "version" : "5.0.4",
            "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
        },
        "serverParameters" : {
            "internalQueryFacetBufferSizeBytes" : 104857600,
            "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
            "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
            "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
            "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
            "internalQueryProhibitBlockingMergeOnMongoS" : 0,
            "internalQueryMaxAddToSetBytes" : 104857600,
            "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
>
```


Query no:2

3. Explain and indexes for Customer Collection on customer_id field. The below output has been produced for query no. 2.(Customers with doctoral degree)

Output a) without indexes:

- The highlighted "stage" : "COLLSCAN" shows that full column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 0 shows that it took less than 0 milli seconds to execute the whole query.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.
- In the executionStats, for executionStages COLLSCAN was performed, and it returned "nReturned" : 8 data based on the filter condition provided in the find query.


```
> db.customers.find({ $and: [{"customer_id":{$exists:true}}, {title:"Dr."}]},{customer_id:true, firstName:true, lastName:true,
email:true}).sort({firstName:1}).limit(10).explain("executionStats")
{
    "explainVersion" : "1",
    "queryPlanner" : {
        "namespace" : "qmulairline.customers",
```

```
"indexFilterSet" : false,
"parsedQuery" : {
        "$and" : [
                {
                        "title" : {
                                "$eq" : "Dr."
                        }
                },
                {
                        "customer_id" : {
                                "$exists" : true
                        }
                }
        ]
},
"maxIndexedOrSolutionsReached" : false,
"maxIndexedAndSolutionsReached" : false,
"maxScansToExplodeReached" : false,
"winningPlan" : {
        "stage" : "PROJECTION_SIMPLE",
        "transformBy" : {
                "customer_id" : true,
                "firstName" : true,
                "lastName" : true,
                "email" : true
        },
        "inputStage" : {
                "stage" : "SORT",
                "sortPattern" : {
                        "firstName" : 1
                },
                "memLimit" : 104857600,
                "limitAmount" : 10,
                "type" : "simple",
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [
                                        {
                                                "title" : {
```

```
                                    "$eq" : "Dr."
                                }
                            },
                            {
                                "customer_id" : {
                                    "$exists" : true
                                }
                            }
                        ]
                    },
                    "direction" : "forward"
                }
            }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 8,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 25,
        "executionStages" : {
            "stage" : "PROJECTION_SIMPLE",
            "nReturned" : 8,
            "executionTimeMillisEstimate" : 0,
            "works" : 36,
            "advanced" : 8,
            "needTime" : 27,
            "needYield" : 0,
            "saveState" : 0,
            "restoreState" : 0,
            "isEOF" : 1,
            "transformBy" : {
                "customer_id" : true,
                "firstName" : true,
                "lastName" : true,
                "email" : true
            },
            "inputStage" : {
```

```
"stage" : "SORT",
"nReturned" : 8,
"executionTimeMillisEstimate" : 0,
"works" : 36,
"advanced" : 8,
"needTime" : 27,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"sortPattern" : {
        "firstName" : 1
},
"memLimit" : 104857600,
"limitAmount" : 10,
"type" : "simple",
"totalDataSizeSorted" : 2404,
"usedDisk" : false,
"inputStage" : {
        "stage" : "COLLSCAN",
        "filter" : {
                "$and" : [
                        {
                                "title" : {
                                        "$eq" : "Dr."
                                }
                        },
                        {
                                "customer_id" : {
                                        "$exists" : true
                                }
                        }
                ]
        },
        "nReturned" : 8,
        "executionTimeMillisEstimate" : 0,
        "works" : 27,
        "advanced" : 8,
        "needTime" : 18,
        "needYield" : 0,
```

```
                    "saveState" : 0,
                    "restoreState" : 0,
                    "isEOF" : 1,
                    "direction" : "forward",
                    "docsExamined" : 25
                }
            }
        }
    },
    "command" : {
        "find" : "customers",
        "filter" : {
            "$and" : [
                {
                    "customer_id" : {
                        "$exists" : true
                    }
                },
                {
                    "title" : "Dr."
                }
            ]
        },
        "limit" : 10,
        "singleBatch" : false,
        "sort" : {
            "firstName" : 1
        },
        "projection" : {
            "customer_id" : true,
            "firstName" : true,
            "lastName" : true,
            "email" : true
        },
        "$db" : "qmulairline"
    },
    "serverInfo" : {
        "host" : "DESKTOP-O2C1LRU",
        "port" : 27017,
        "version" : "5.0.4",
```

```
        "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
    },
    "serverParameters" : {
        "internalQueryFacetBufferSizeBytes" : 104857600,
        "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
        "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
        "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
        "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
        "internalQueryProhibitBlockingMergeOnMongoS" : 0,
        "internalQueryMaxAddToSetBytes" : 104857600,
        "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
    },
    "ok" : 1
}
>
```

Created indexes for employees collection for employee_id field.

```
db.customers.createIndex({"customer_id ":1});
db.customers.getIndexes();
db.customers.dropIndex("customer_id _1");
```

Output b) after indexes:

- The highlighted "stage" : "IXSCAN" shows that index column scan was done on the document collection to fetch all relevant data.
- The field "executionTimeMillisEstimate" : 0 shows that it took less than 0 milli seconds to execute the whole query.
- The "executionSuccess" : true shows that there are no logic or formatting issue and the query executed successfully.
- In the executionStats, for executionStages IXSCAN was performed, and it returned "nReturned" : 25 data
  as there are overall 25 documents in bookings collection and then in FETCH stage it returned "nReturned" : 8 data based on the filter condition provided in the find query.

```
> db.customers.find({ $and: [{"customer_id":{$exists:true}}, {title:"Dr."}]},{customer_id:true, firstName:true, lastName:true,
email:true}).sort({firstName:1}).limit(10).explain("executionStats")
{
    "explainVersion" : "1",
    "queryPlanner" : {
        "namespace" : "qmulairline.customers",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "$and" : [
                {
                    "title" : {
                        "$eq" : "Dr."
```

```
                }
            },
            {
                "customer_id" : {
                    "$exists" : true
                }
            }
        ]
},
"maxIndexedOrSolutionsReached" : false,
"maxIndexedAndSolutionsReached" : false,
"maxScansToExplodeReached" : false,
"winningPlan" : {
        "stage" : "PROJECTION_SIMPLE",
        "transformBy" : {
            "customer_id" : true,
            "firstName" : true,
            "lastName" : true,
            "email" : true
        },
        "inputStage" : {
            "stage" : "SORT",
            "sortPattern" : {
                "firstName" : 1
            },
            "memLimit" : 104857600,
            "limitAmount" : 10,
            "type" : "simple",
            "inputStage" : {
                "stage" : "FETCH",
                "filter" : {
                    "$and" : [
                        {
                            "customer_id" : {
                                "$exists" : true
                            }
                        },
                        {
                            "title" : {
                                "$eq" : "Dr."
```

```
                                }
                            }
                        ]
                    },
                    "inputStage" : {
                        "stage" : "IXSCAN",
                        "keyPattern" : {
                            "customer_id" : 1
                        },
                        "indexName" : "customer_id_1",
                        "isMultiKey" : false,
                        "multiKeyPaths" : {
                            "customer_id" : [ ]
                        },
                        "isUnique" : false,
                        "isSparse" : false,
                        "isPartial" : false,
                        "indexVersion" : 2,
                        "direction" : "forward",
                        "indexBounds" : {
                            "customer_id" : [
                                "[MinKey, MaxKey]"
                            ]
                        }
                    }
                }
            }
        },
        "rejectedPlans" : [ ]
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 8,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 25,
        "totalDocsExamined" : 25,
        "executionStages" : {
            "stage" : "PROJECTION_SIMPLE",
            "nReturned" : 8,
            "executionTimeMillisEstimate" : 0,
```

```
"works" : 35,
"advanced" : 8,
"needTime" : 26,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"transformBy" : {
        "customer_id" : true,
        "firstName" : true,
        "lastName" : true,
        "email" : true
},
"inputStage" : {
        "stage" : "SORT",
        "nReturned" : 8,
        "executionTimeMillisEstimate" : 0,
        "works" : 35,
        "advanced" : 8,
        "needTime" : 26,
        "needYield" : 0,
        "saveState" : 0,
        "restoreState" : 0,
        "isEOF" : 1,
        "sortPattern" : {
                "firstName" : 1
        },
        "memLimit" : 104857600,
        "limitAmount" : 10,
        "type" : "simple",
        "totalDataSizeSorted" : 2404,
        "usedDisk" : false,
        "inputStage" : {
                "stage" : "FETCH",
                "filter" : {
                        "$and" : [
                                {
                                        "customer_id" : {
                                                "$exists" : true
                                        }
```

```
            },
            {
                "title" : {
                    "$eq" : "Dr."
                }
            }
        ]
},
"nReturned" : 8,
"executionTimeMillisEstimate" : 0,
"works" : 26,
"advanced" : 8,
"needTime" : 17,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"docsExamined" : 25,
"alreadyHasObj" : 0,
"inputStage" : {
    "stage" : "IXSCAN",
    "nReturned" : 25,
    "executionTimeMillisEstimate" : 0,
    "works" : 26,
    "advanced" : 25,
    "needTime" : 0,
    "needYield" : 0,
    "saveState" : 0,
    "restoreState" : 0,
    "isEOF" : 1,
    "keyPattern" : {
        "customer_id" : 1
    },
    "indexName" : "customer_id_1",
    "isMultiKey" : false,
    "multiKeyPaths" : {
        "customer_id" : [ ]
    },
    "isUnique" : false,
    "isSparse" : false,
```

```
                        "isPartial" : false,
                        "indexVersion" : 2,
                        "direction" : "forward",
                        "indexBounds" : {
                            "customer_id" : [
                                "[MinKey, MaxKey]"
                            ]
                        },
                        "keysExamined" : 25,
                        "seeks" : 1,
                        "dupsTested" : 0,
                        "dupsDropped" : 0
                    }
                }
            }
        }
    },
    "command" : {
        "find" : "customers",
        "filter" : {
            "$and" : [
                {
                    "customer_id" : {
                        "$exists" : true
                    }
                },
                {
                    "title" : "Dr."
                }
            ]
        },
        "limit" : 10,
        "singleBatch" : false,
        "sort" : {
            "firstName" : 1
        },
        "projection" : {
            "customer_id" : true,
            "firstName" : true,
            "lastName" : true,
```

```
                "email" : true
            },
            "$db" : "qmulairline"
        },
        "serverInfo" : {
            "host" : "DESKTOP-O2C1LRU",
            "port" : 27017,
            "version" : "5.0.4",
            "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
        },
        "serverParameters" : {
            "internalQueryFacetBufferSizeBytes" : 104857600,
            "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
            "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
            "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
            "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
            "internalQueryProhibitBlockingMergeOnMongoS" : 0,
            "internalQueryMaxAddToSetBytes" : 104857600,
            "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
>
```