

The following is a documentation for a taxi company build with MongoDB database management system which contains the data of drivers, cars, operators, bookings, payments, shifts, clients, and revenue. The system is initially designed with an Entity Relationship Diagram (ERD) which illustrates the relationship between the created entities for the company under NoSQL. The document consists of the assumption made for the database design, how the database has been mapped and a set of queries that extracts the data from the database system.

A. Assumptions

The design of the database system was implemented with the following assumptions:

- 1) The taxi company operates under one time-zone.
- 2) Staffs (drivers, operators) have multiple shifts (morning, afternoon, evening, night).
- 3) The revenue has been designed to be recorded monthly.
- 4) A membership number is given to customers who travel under a regular booking.
- 5) The payment currency is GBP.

B. Entity Relationship Diagram (ERD)

The database has been designed using the entity relationship diagram (figure 1) which contains the information of the collection's fields and their types. The table headers highlighted with green represents the collections created and yellow represents the objects embedded within each collection.

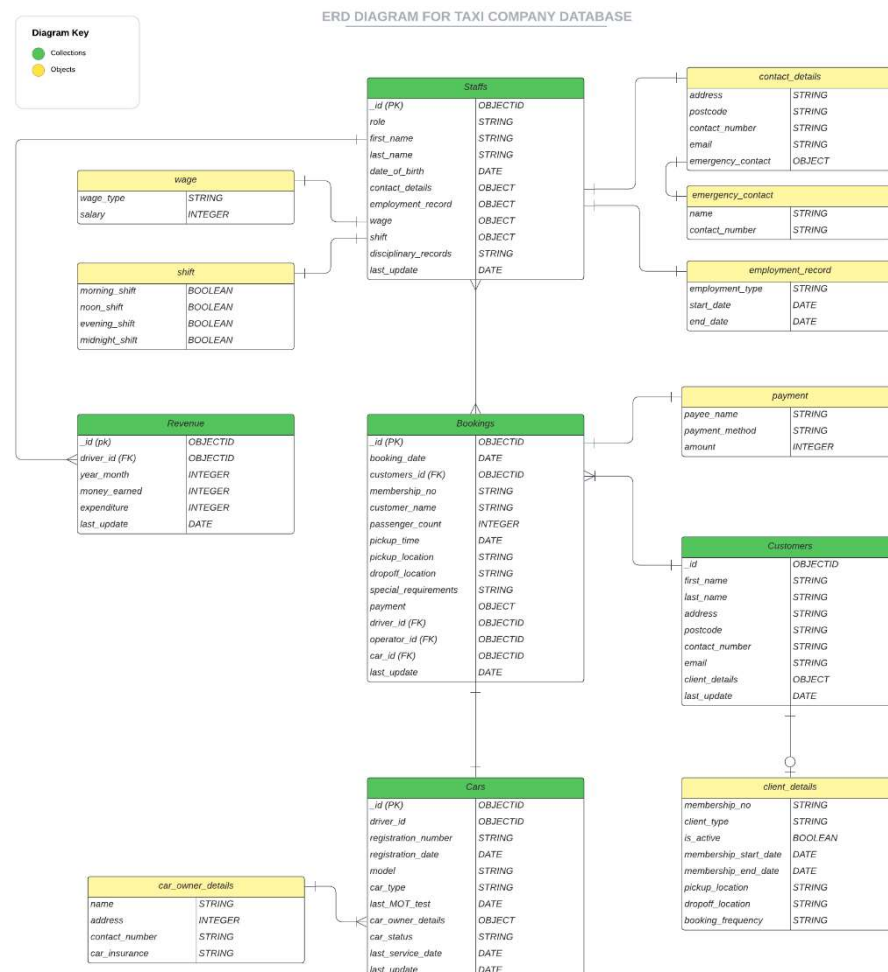


Figure 1. ERD for the taxi company

C. High-Level Design Mapping

The below tables represent how the collections are mapped and their relationship with their respective object data.

A. Collections:

1. Collection – Staff

Entity Relation: the staff collection has a many-to-many relation with bookings collection and a one-to-many relation to revenue collection.

No.	Field Name	Data Type	Remarks
1	<u>id</u> (Primary Key)	ObjectID	Staff ID
2	role	String	roles: driver, operator
3	first_name	String	Staff's first name
4	last_name	String	Staff's last name
5	date_of_birth	Date	Staff's date of birth
6	contact_details	Object	refer to: table 6; one-to-one relation
7	employment_record	Object	refer to: table 7; one-to-one relation
8	wage	Object	refer to: table 9; one-to-one relation
9	shift	Object	refer to: table 10; one-to-one relation
10	disciplinary_records	String	
11	last_update	Date	last update Date of collection

Table 1 Staff collection

2. Collection – Bookings

Entity Relation: the bookings collection has a one/many-to-one relation with customers collection, one-to-many relation to cars collection and a many-to-many relation with staff collection.

No.	Field Name	Data Type	Remarks
1	<u>id</u> (Primary Key)	ObjectID	Bookings ID
2	booking_date	Date	Date of booking
3	customers_id (Foreign Key)	ObjectID	Booking Customer's ID
4	membership_no	String	indicates the membership number for clients, i.e., for customers who travel on a regular basis
5	customer_name	String	Booking Customer's name
6	passenger_count	Integer	Passenger Count for the trip

No.	Field Name	Data Type	Remarks
7	pickup_time	Date	
8	pickup_location	String	
9	dropoff_location	String	
10	special_requirements	String	
11	payment	Object	refer to: table 11; one-to-one relation
12	driver_id (Foreign Key)	ObjectID	ID of the driver in-charge for the trip
13	operator_id (Foreign Key)	ObjectID	ID of the operator in-charge for the booking of the trip
14	car_id (Foreign Key)	ObjectID	ID of the assigned car for the booking
15	last_update	Date	last update Date of collection

Table 2 Bookings Collection

3. Collection – Customers

Entity Relation: the customers collection has a one-to-one/many relation with bookings collection.

No.	Field Name	Data Type	Remarks
1	_id (Primary Key)	ObjectID	Customer ID
2	first_name	String	Customer's first name
3	last_name	String	Customer's last name
4	address	String	Customer's address
5	postcode	String	Address postcode
6	contact_number	String	contact numbers are set to type String to insert their country codes
7	email	String	Customer's email
8	client_details	Object	refer to: table 12; one-to-zero/one relation
9	last_update	Date	last update Date of collection

Table 3 Customers Collection

4. Collection – Cars

Entity Relation: the cars collection has a one-to-one relation with bookings collection.

No.	Field Name	Data Type	Remarks
1	_id (Primary Key)	ObjectID	Car ID
2	driver_id	ObjectID	Driver ID
3	registration_number	String	car registration number
4	registration_date	Date	car registration date

No.	Field Name	Data Type	Remarks
5	model	String	represents the car brand
6	car_type	String	represents the car type (e.g., sedan, minivan, etc.) customers desire
7	last_MOT_test	Date	last transport test of cars
8	car_owner_details	Object	refer to: table 13; one/many-to-one relation
9	car_status	String	car statuses: roadworthy, in for service, awaiting repair, written off
10	last_service_date	Date	last car service date
11	last_update	Date	last update Date of collection

Table 4 Cars Collection

5. Collection – Revenue

Entity Relation: the customers collection has a many-to-one relation with staff collection.

No.	Field Name	Data Type	Remarks
1	_id (Primary Key)	ObjectID	Revenue ID
2	driver_id(Foreign Key)	ObjectID	Driver ID
3	year_month	Integer	Revenue of the corresponding month; in the format 'YYMM'
4	money_earned	Integer	Money earned in that corresponding month
5	expenditure	Integer	compromises the money spent on car maintenances, salaries, company costs, etc.)
6	last_update	Date	last update Date of collection

Table 5 Revenue Collection

B. Objects

1. Object - contact_details

The following object represents the contact details of staffs (drivers, operators)

No.	Field Name	Data Type	Remarks
1	address	String	
2	postcode	String	
3	contact_number	String	contact numbers are set to type String to

No.	Field Name	Data Type	Remarks
			insert their country codes
4	email	String	
5	emergency_contact	Object	refer to table: 7; one-to-one relation

Table 6 Object of contact details

2. Object - emergency_contact

The emergency_contact object has been generated to store emergency contact details of staffs working in the company.

No.	Field Name	Data Type	Remarks
1	name	String	
2	contact_number	String	

Table 7 Object of emergency contact

3. Object - employment_record

The following object stores the staff's employment records, such as their employment type, start date and end date for staffs who have left the company.

No.	Field Name	Data Type	Remarks
1	employment_type	String	employment types: permanent, temporary, and part-time
2	start_date	Date	
3	end_date	Date	

Table 8 Object of employment record

4. Object – wage

The wage object represents the wage types of each staff and their salary.

No.	Field Name	Data Type	Remarks
1	wage_type	String	Wage types: varied, fixed
2	salary	Integer	

Table 9 Object of wage

5. Object – shift

To cover 24-hour shifts, the shift object is injected to keep track of the staff's shifts.

No.	Field Name	Data Type	Remarks
1	morning_shift	Boolean	
2	noon_shift	Boolean	
3	evening_shift	Boolean	
4	midnight_shift	Boolean	

Table 10 Object of shift

6. Object – payment

Payment object is embedded within the bookings collection to record the payments received for each booking.

No.	Field Name	Data Type	Remarks
1	payee_name	String	
2	payment_method	String	payment methods: cash, card
3	amount	Integer	

Table 11 Object of payment

7. Object - client_details

Client details encloses the details of those customers that have a membership for regular bookings.

No.	Field Name	Data Type	Remarks
1	membership_no	String	
2	client_type	String	
3	is_active	Boolean	To keep track of the memberships that are active and inactive
4	membership_start_date	Date	
5	membership_end_date	Date	
6	pickup_location	String	Regular pick-up location
7	dropoff_location	String	Regular drop-off location
8	booking_frequency	String	Represents how frequent the booking is; booking frequencies: weekly, daily

Table 12 Object of client details

8. Object - car_owner_details

This following object consists of the details of car owners.

No.	Field Name	Data Type	Remarks
1	name	String	
2	address	Integer	
3	contact_number	String	
4	car_insurance	String	

Table 13 Object of car owner details

D. Queries

1. The query to get the top 10 driver names, who have earned the most amount of income in the month of December 2022. The query also concatenates the first_name and last_name to get the name field.

Query Code:

```
db.revenues.aggregate([
  { $match: { year_month: 2212 } },
  { $sort: { money_earned: -1 } },
  { $limit: 10 },
  {
```

```

    $lookup: {
      from: 'staffs',
      localField: 'driver_id',
      foreignField: '_id',
      as: 'driver',
    },
  },
  { $unwind: { path: '$driver' } },
  {
    $project: {
      name: {
        $concat: [
          '$driver.first_name',
          ' ',
          '$driver.last_name'
        ],
      },
      income: '$money_earned',
    },
  },
}
]);

```

Output screen print:

```

{ "_id" : ObjectId("639a4b683973f2a54147c3d7"), "name" : "Lura Streich", "income" : 4966 }
{ "_id" : ObjectId("639a4b683973f2a54147c609"), "name" : "Myrtie Stiedemann", "income" : 4957 }
{ "_id" : ObjectId("639a4b683973f2a54147c591"), "name" : "Reyes Rodriguez", "income" : 4957 }
{ "_id" : ObjectId("639a4b683973f2a54147c70f"), "name" : "Arch Jacobs", "income" : 4928 }
{ "_id" : ObjectId("639a4b673973f2a54147c243"), "name" : "Baron Shanahan", "income" : 4899 }
{ "_id" : ObjectId("639a4b673973f2a54147c0cd"), "name" : "Jorge Orn", "income" : 4876 }
{ "_id" : ObjectId("639a4b673973f2a54147c319"), "name" : "Stacey Rutherford", "income" : 4626 }
{ "_id" : ObjectId("639a4b683973f2a54147c3a5"), "name" : "Loyal Kuphal", "income" : 4625 }
{ "_id" : ObjectId("639a4b683973f2a54147c44f"), "name" : "Bianka Bauch", "income" : 4623 }
{ "_id" : ObjectId("639a4b683973f2a54147c3c1"), "name" : "Rosario Johnston", "income" : 4620 }
> 

```

- The query to get the total amount of revenue earned in each month, ordered in descending order of month. The integer month is converted into month in string and the year is separated from the year_month field.

Query Code:

```

db.revenues.aggregate([
  { $group: { _id: '$year_month', totalIncome: { $sum:
'$money_earned' } } },
  { $sort: { _id: -1 } },
  {
    $addFields: {
      month: {
        $let: {
          vars: {
            monthsInString: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
            monthInNum: { $subtract: [{ $mod: ['$_id', 100] },
1] },
          },
        },
      },
    },
  },
]

```

```

        in: { $arrayElemAt: ['$monthsInString',
'$monthInNum'] },
        },
        year: { $add: [{ $floor: { $divide: ['$_id', 100] } },
2000] },
        },
        },
        { $project: { _id: 0 } },
    ]);

```

Output screen print:

```

{ "totalIncome" : 297427, "month" : "Dec", "year" : 2022 }
{ "totalIncome" : 243621, "month" : "Nov", "year" : 2022 }
{ "totalIncome" : 243266, "month" : "Oct", "year" : 2022 }
{ "totalIncome" : 206126, "month" : "Sep", "year" : 2022 }
{ "totalIncome" : 203051, "month" : "Aug", "year" : 2022 }
{ "totalIncome" : 159172, "month" : "Jul", "year" : 2022 }
{ "totalIncome" : 139978, "month" : "Jun", "year" : 2022 }
{ "totalIncome" : 135745, "month" : "May", "year" : 2022 }
{ "totalIncome" : 109270, "month" : "Apr", "year" : 2022 }
{ "totalIncome" : 82752, "month" : "Mar", "year" : 2022 }
{ "totalIncome" : 54741, "month" : "Feb", "year" : 2022 }
{ "totalIncome" : 40007, "month" : "Jan", "year" : 2022 }
{ "totalIncome" : 29945, "month" : "Dec", "year" : 2021 }
{ "totalIncome" : 7117, "month" : "Nov", "year" : 2021 }
> 

```

3. The query to get the top 10 non-member customers with the most booking in descending order of booking count. A customer is a non-member when there exists a customer_id field present in the document of the booking.

Query Code:

```

db.bookings.aggregate([
    { $match: { customer_id: { $exists: true } } },
    { $group: { _id: '$customer_id', booking_count: { $sum: 1 } } },
    { $sort: { booking_count: -1 } },
    { $limit: 10 },
]);

```

Output screen print:

```

{ "_id" : ObjectId("639a4b683973f2a54147c85e"), "booking_count" : 30 }
{ "_id" : ObjectId("639a4b693973f2a54147cbcc"), "booking_count" : 30 }
{ "_id" : ObjectId("639a4b6a3973f2a54147d3c0"), "booking_count" : 30 }
{ "_id" : ObjectId("639a4b6a3973f2a54147d096"), "booking_count" : 29 }
{ "_id" : ObjectId("639a4b693973f2a54147cdde"), "booking_count" : 29 }
{ "_id" : ObjectId("639a4b693973f2a54147cd6a"), "booking_count" : 29 }
{ "_id" : ObjectId("639a4b693973f2a54147c9c2"), "booking_count" : 29 }
{ "_id" : ObjectId("639a4b693973f2a54147c9fe"), "booking_count" : 29 }
{ "_id" : ObjectId("639a4b693973f2a54147cea8"), "booking_count" : 28 }
{ "_id" : ObjectId("639a4b693973f2a54147cb1c"), "booking_count" : 28 }
> 

```


- The query to count the number of customers who have membership. A customer is determined to who a membership when the customer document contains the client_details field in it which is a subdocument about the membership.

Query Code:

```
db.customers.countDocuments({ client_details: { $exists: true }
});
```

Output screen print:

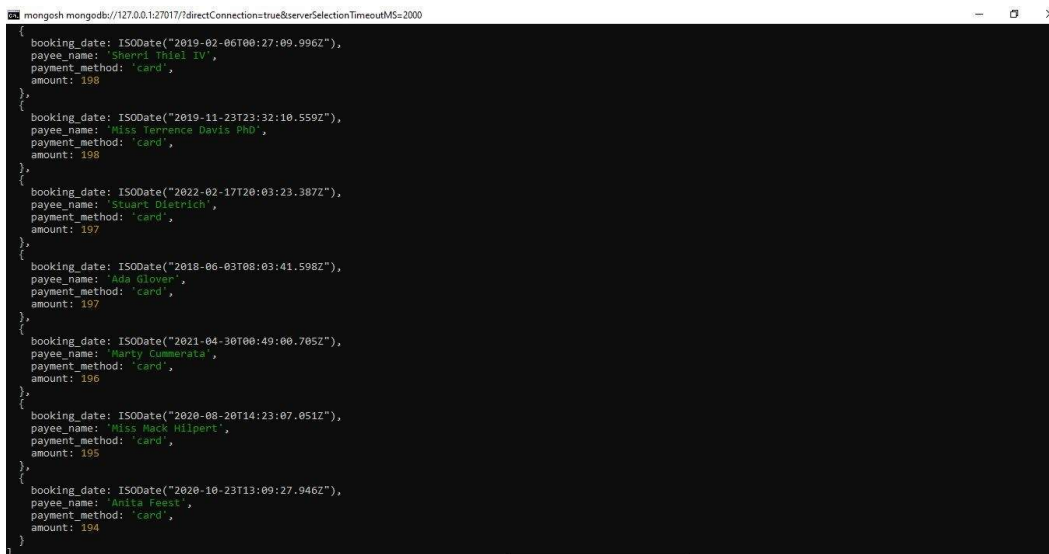
```
> db.customers.countDocuments({ client_details: { $exists: true } });
44
> []
```

- This query gets the top 20 payments for bookings using the card payment method and they are sorted by the highest amount. The result shows the booking_date, payee_name, payment_method and the payment amount sorted by the highest amount.

Query Code:

```
db.bookings.aggregate([{$match: { customer_id: { $exists: true,
}, "payment.payment_method": "card", }, }, {$sort: {
'payment.amount': -1, }, }, {$limit:20},{
$project:{booking_date:'$booking_date',payee_name:'$payment.payee_name',payment_method:'$payment.payment_method',amount:'$payment.amount', _id:0}}]);
```

Output screen print:



```
{
  booking_date: ISODate("2019-02-06T08:27:09.996Z"),
  payee_name: 'Sherri Thiel IV',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2019-11-23T23:32:10.559Z"),
  payee_name: 'Miss Terrence Davis PhD',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2022-02-17T20:03:23.387Z"),
  payee_name: 'Stuart Dietrich',
  payment_method: 'card',
  amount: 197
},
{
  booking_date: ISODate("2018-06-03T08:03:41.598Z"),
  payee_name: 'Ada Glover',
  payment_method: 'card',
  amount: 197
},
{
  booking_date: ISODate("2021-04-30T00:49:00.705Z"),
  payee_name: 'Marty Cuemeralta',
  payment_method: 'card',
  amount: 196
},
{
  booking_date: ISODate("2020-08-20T14:23:07.051Z"),
  payee_name: 'Miss Mack Hilpert',
  payment_method: 'card',
  amount: 195
},
{
  booking_date: ISODate("2020-10-23T13:09:27.946Z"),
  payee_name: 'Anita Feest',
  payment_method: 'card',
  amount: 194
}
```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  booking_date: ISODate("2018-09-26T13:57:40.913Z"),
  payee_name: 'Henrietta Leuschke',
  payment_method: 'card',
  amount: 199
},
{
  booking_date: ISODate("2022-02-15T19:43:310Z"),
  payee_name: 'Javier Ritchie DVM',
  payment_method: 'card',
  amount: 199
},
{
  booking_date: ISODate("2019-09-07T14:48:53.372Z"),
  payee_name: 'Bert Goodwin',
  payment_method: 'card',
  amount: 199
},
{
  booking_date: ISODate("2019-02-14T11:51:43.277Z"),
  payee_name: 'Daniel Block',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2022-05-05T08:32:28.863Z"),
  payee_name: 'Dennis Hermiston',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2018-12-08T06:03:53.767Z"),
  payee_name: 'Ms. Marguerite Oberbrunner',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2019-02-06T00:27:09.996Z"),
  payee_name: 'Sherri Thiel IV',
  payment_method: 'card',
  amount: 198
},
{
  booking_date: ISODate("2019-11-23T23:32:10.559Z"),

```

```

taxidatabase> db.bookings.aggregate([{$match: { customer_id: { $exists: true, }, "payment.payment_method": "card", }, }, {$sort: { 'payment.amount': -1, }, }, {$limit: 20}, {$project: {booking_date: $booking_date, payee_name: $payment.payee_name, payment_method: $payment.payment_method, amount: $payment.amount, _id: 0}}]);
{
  booking_date: ISODate("2019-01-24T23:38:13.995Z"),
  payee_name: 'Elaine Kub IV',
  payment_method: 'card',
  amount: 200
},
{
  booking_date: ISODate("2021-01-17T13:30:32.052Z"),
  payee_name: 'Oliver Schowalter',
  payment_method: 'card',
  amount: 200
},
{
  booking_date: ISODate("2022-04-07T14:22:01.234Z"),
  payee_name: 'Martin Baumbach',
  payment_method: 'card',
  amount: 200
},
{
  booking_date: ISODate("2018-06-09T01:22:55.196Z"),
  payee_name: 'Dean Keebler DVM',
  payment_method: 'card',
  amount: 200
},
{
  booking_date: ISODate("2018-05-04T13:32:38.546Z"),
  payee_name: 'Terence Bartell Jr.',
  payment_method: 'card',
  amount: 199
},
{
  booking_date: ISODate("2019-06-10T15:23:46.873Z"),
  payee_name: 'Brittany Thiel',
  payment_method: 'card',
  amount: 199
},
{
  booking_date: ISODate("2019-12-03T21:24:09.741Z"),
  payee_name: 'Geoffrey Kunze',
  payment_method: 'card',
  amount: 199
},

```

- This query gets the top 10 salary wages of staff whose role is a driver, employed permanently with the company sorted. The result displays a combined first_name & last_name, date_of_birth of the employee and the salary sorted by highest wage.

Query Code:

```

db.staffs.aggregate([{$match: {role:
'driver', 'employment_record.employment_type': 'permanent' }, },
{$sort: { 'wage.salary': -1, }, }, {$limit: 10}, {$project: {
name: { $concat: ['$first_name', ' ', '$last_name'] },
date_of_birth: 1, Salary : '$wage.salary', _id: 0} }]);

```

Output screen print:

```
taxidatabase> db.staffs.aggregate([{$match: {role: 'driver', employment_record.employment_type: 'permanent' }}, {$sort: { 'wage.salary': -1, }, {$limit:10},{ $project: { name: { $concat: [ '$first_name', ' ', '$last_name' ] }, date_of_birth: 1, Salary : '$wage.salary', _id:0 } }]);
[
  {
    date_of_birth: ISODate("1990-06-23T05:29:10.664Z"),
    name: 'Ross Medhurst',
    Salary: 9838
  },
  {
    date_of_birth: ISODate("1990-03-03T21:28:21.759Z"),
    name: 'Kathlyn Deckow',
    Salary: 9754
  },
  {
    date_of_birth: ISODate("1989-11-01T09:55:00.864Z"),
    name: 'Novella Rosenbaum',
    Salary: 9277
  },
  {
    date_of_birth: ISODate("1983-03-28T08:34:10.350Z"),
    name: 'Dewayne Towne',
    Salary: 9152
  },
  {
    date_of_birth: ISODate("1976-03-14T22:28:59.137Z"),
    name: 'Athena Streich',
    Salary: 8259
  },
  {
    date_of_birth: ISODate("1976-07-17T15:48:49.300Z"),
    name: 'Theresia Cumerata',
    Salary: 8159
  },
  {
    date_of_birth: ISODate("1996-11-06T20:33:50.997Z"),
    name: 'Abdullah McLaughlin',
    Salary: 7742
  },
  {
    date_of_birth: ISODate("1989-09-16T11:08:55.258Z"),
    name: 'Assunta Walter',
    Salary: 7567
  },
  {
    date_of_birth: ISODate("1973-10-31T19:27:18.171Z"),
    name: 'Event DuBuque',
    Salary: 7033
  },
  {
    date_of_birth: ISODate("1977-10-20T05:29:37.534Z"),
    name: 'Josie Buckridge',
    Salary: 6083
  }
]
```

```
{
  date_of_birth: ISODate("1996-11-06T20:33:50.997Z"),
  name: 'Abdullah McLaughlin',
  Salary: 7742
},
{
  date_of_birth: ISODate("1989-09-16T11:08:55.258Z"),
  name: 'Assunta Walter',
  Salary: 7567
},
{
  date_of_birth: ISODate("1973-10-31T19:27:18.171Z"),
  name: 'Event DuBuque',
  Salary: 7033
},
{
  date_of_birth: ISODate("1977-10-20T05:29:37.534Z"),
  name: 'Josie Buckridge',
  Salary: 6083
}
]
```

7. This query gets the top 10 list of cars which are not roadworthy sorted by last serviced date. The result displays the car registration number, car type, model, insurance number, car status and the last service date sorted by the longest duration for this service.

Query Code:

```
db.cars.aggregate([{$match: {car_status:{ $not: { $eq:
"roadworthy"}}}},{$skip:10},{ $limit:10},{ $project:{ Name:
"$car_owner_details.name",RegistrationNumber :
"$registration_number",CarType: "$car_type",Model:
"$model",InsuranceNumber:
"$car_owner_details.car_insurance",LastServiceDate:
"$last_service_date",CarStatus:'$car_status', _id:0}},{$sort:
{LastServiceDate: 1}}]);
```

Output screen print:

```
taxidatabase> db.cars.aggregate([{$match: {car_status: { $not: { $eq: "roadworthy" }}}},{$skip:10},{$limit:10},{ $project: { Name: "$car_owner_details.name",RegistrationNumber: "$Registration_number",CarType: "$Car_type",Model: "$Model",InsuranceNumber: "$car_owner_details.car_insurance",LastServiceDate: "$last_service_date",CarStatus: "$car_status",_id:0}},{$sort: (LastServiceDate: 1)}}]);
{
  Name: 'Stacey Hackett',
  RegistrationNumber: 'RMB42TW',
  CarType: 'Extended Cab Pickup',
  Model: 'Model V',
  InsuranceNumber: '358 757 8888',
  LastServiceDate: ISODate("2018-04-24T05:51:21.477Z"),
  CarStatus: 'in for service'
},
{
  Name: 'Neal Rowanquern',
  RegistrationNumber: 'KT04ACB',
  CarType: 'Passenger Van',
  Model: 'Explorer',
  InsuranceNumber: '111 717 2187',
  LastServiceDate: ISODate("2019-05-20T11:48:54.370Z"),
  CarStatus: 'awaiting repair'
},
{
  Name: 'Laurence Mueller',
  RegistrationNumber: 'HRB6RAF',
  CarType: 'Coupe',
  Model: '011',
  InsuranceNumber: '771 522 7291',
  LastServiceDate: ISODate("2020-08-17T22:35:00.630Z"),
  CarStatus: 'awaiting repair'
},
{
  Name: 'Bruce Simes',
  RegistrationNumber: 'CC3SUZC',
  CarType: 'SUV',
  Model: 'Accord',
  InsuranceNumber: '472 880 0631',
  LastServiceDate: ISODate("2020-10-31T14:04:06.032Z"),
  CarStatus: 'awaiting repair'
},
{
  Name: 'Samantha Torphy',
  RegistrationNumber: 'WQ36ZTA',
  CarType: 'Cargo Van',
  InsuranceNumber: '969 943 1938',
  LastServiceDate: ISODate("2021-12-14T01:06:23.957Z"),
  CarStatus: 'in for service'
},
{
  Name: 'Roberto Dare',
  RegistrationNumber: 'UV39PWD',
  CarType: 'Wagon',
  Model: 'Cruze',
  InsuranceNumber: '802 087 3580',
  LastServiceDate: ISODate("2022-05-05T06:11:34.262Z"),
  CarStatus: 'awaiting repair'
},
{
  Name: 'Kelly Shanahan',
  RegistrationNumber: 'WN77YGN',
  CarType: 'Sedan',
  Model: 'Fortwo',
  InsuranceNumber: '905 927 5528',
  LastServiceDate: ISODate("2022-05-07T10:46:58.761Z"),
  CarStatus: 'awaiting repair'
},
{
  Name: 'Derek Price',
  RegistrationNumber: 'WK85VIC',
  CarType: 'Crew Cab Pickup',
  Model: 'ATS',
  InsuranceNumber: '654 871 3635',
  LastServiceDate: ISODate("2022-07-06T07:53:01.544Z"),
  CarStatus: 'in for service'
},
{
  Name: 'Chad Thompson',
  RegistrationNumber: 'R036EQG',
  CarType: 'Minivan',
  Model: 'El Camino',
  InsuranceNumber: '951 426 2307',
  LastServiceDate: ISODate("2022-07-16T22:22:01.495Z"),
  CarStatus: 'written off'
}
```

8. This query gets the total no of bookings of private clients who travelled during the weekends. The result shows count for the total no of Bookings by private clients at weekends.

Query Code:

```
db.customers.aggregate([{$match: {$and:
[{"client_details.booking_frequency":
"weekends"}, {"client_details.client_type":
"private"}]}],{$count: "Total No of Private Bookings in
weekends"}]);
```

Output screen print:

```
taxidatabase>
taxidatabase>
taxidatabase> db.customers.aggregate([{$match: {$and: [{"client_details.booking_frequcy": "weekends"}, {"client_details.client_type": "private"}]}], {$count: "Total No o
Private Bookings in weekends"}}]
[ { 'Total No of Private Bookings in weekends': 6 } ]
taxidatabase>
```

9. This query gets the top 5 drivers' revenues in descending order for December 2022.

Query Code:

```
db.revenues.aggregate ([
  {
    $match: {
      year_month: 2212
    },
  },
  {
    $project: {
      _id: 1,
      revenue: {
        $subtract: ["$money_earned", "$expenditure"],
      },
    },
  },
  {
    $sort: {
      revenue: -1,
    },
  },
  {
    $limit: 5,
  },
])
```

Output screen print:

```
> db.revenues.aggregate([
...   {
...     $match: {
...       year_month: 2212,
...     },
...   },
...   {
...     $project: {
...       _id: 1,
...       revenue: {
...         $subtract: ["$money_earned", "$expenditure"],
...       },
...     },
...   },
...   {
...     $sort: {
...       revenue: -1,
...     },
...   },
...   {
...     $limit: 5,
...   },
... ])
... { "_id" : ObjectId("639a3154a812ec1939dde4a1"), "revenue" : 4741 }
... { "_id" : ObjectId("639a3154a812ec1939dde5c7"), "revenue" : 4672 }
... { "_id" : ObjectId("639a3154a812ec1939dde99d"), "revenue" : 4575 }
... { "_id" : ObjectId("639a3154a812ec1939dde6dd"), "revenue" : 4571 }
... { "_id" : ObjectId("639a3154a812ec1939dde7c9"), "revenue" : 4550 }
```

10. The query compares the Car type to the string “Minivan” using \$strcasecmp where the results with 0 are when 2 strings are matched as Minivan, results in 1 when the first letter of the resultant string is greater than the first letter of Minivan “M”, and -1 when the first letter of the resultant string is less than the first letter of Minivan “M”.

Query Code:

```
db.cars.aggregate ([
  {
    $project: {
      car_type: 1,
      cmpTominivan: {
        $strcasecmp: ["$car_type", "Minivan"],
      },
    },
  },
  {
    $limit: 10,
  },
],])
```

Output screen print:

```
db.cars.aggregate([
  {
    $project: {
      car_type: 1,
      cmpTominivan: {
        $strcasecmp: ["$car_type", "Minivan"],
      },
    },
  },
  {
    $limit: 10,
  },
])
{"_id": "ObjectId('639a3154a812ec1939dde3dc")", "car_type": "Minivan", "cmpTominivan": 0 }
{"_id": "ObjectId('639a3154a812ec1939dde3f4")", "car_type": "Passenger Van", "cmpTominivan": 1 }
{"_id": "ObjectId('639a3154a812ec1939dde3fc")", "car_type": "Sedan", "cmpTominivan": 1 }
{"_id": "ObjectId('639a3154a812ec1939dde41a")", "car_type": "Wagon", "cmpTominivan": 1 }
{"_id": "ObjectId('639a3154a812ec1939dde43c")", "car_type": "Sedan", "cmpTominivan": 1 }
{"_id": "ObjectId('639a3154a812ec1939dde456")", "car_type": "Coupe", "cmpTominivan": -1 }
{"_id": "ObjectId('639a3154a812ec1939dde464")", "car_type": "Hatchback", "cmpTominivan": -1 }
{"_id": "ObjectId('639a3154a812ec1939dde482")", "car_type": "Cargo Van", "cmpTominivan": -1 }
{"_id": "ObjectId('639a3154a812ec1939dde4a0")", "car_type": "Cargo Van", "cmpTominivan": -1 }
{"_id": "ObjectId('639a3154a812ec1939dde4c2")", "car_type": "Hatchback", "cmpTominivan": -1 }
```

11. This query is used to get the total earnings of drivers in the month of July in 2022.

Query Code:

```
db.revenues.aggregate ([
  {
    $match: {
      year_month: 2207,
    },
  },
  {
    $group: {
      _id: "$driver_id",
      total: {
        $sum: "$money_earned",
      },
    },
  },
],])
```

```

    },
  },
  {
    $sort: {
      total: -1,
    },
  },
}

])

```

Output screen print:

```

"_id" : ObjectId("639a3155a812ec1939dde9e6"), "total" : 4972 }
"_id" : ObjectId("639a3154a812ec1939dde626"), "total" : 4933 }
"_id" : ObjectId("639a3154a812ec1939dde480"), "total" : 4889 }
"_id" : ObjectId("639a3154a812ec1939dde7c6"), "total" : 4782 }
"_id" : ObjectId("639a3154a812ec1939dde652"), "total" : 4586 }
"_id" : ObjectId("639a3154a812ec1939dde814"), "total" : 4420 }
"_id" : ObjectId("639a3154a812ec1939dde866"), "total" : 4285 }
"_id" : ObjectId("639a3154a812ec1939dde3fa"), "total" : 4179 }
"_id" : ObjectId("639a3155a812ec1939ddeade"), "total" : 4114 }
"_id" : ObjectId("639a3154a812ec1939dde544"), "total" : 4047 }
"_id" : ObjectId("639a3155a812ec1939dde68"), "total" : 4044 }
"_id" : ObjectId("639a3154a812ec1939dde79a"), "total" : 4032 }
"_id" : ObjectId("639a3154a812ec1939dde990"), "total" : 4021 }
"_id" : ObjectId("639a3154a812ec1939dde9a0"), "total" : 3961 }
"_id" : ObjectId("639a3154a812ec1939dde750"), "total" : 3649 }
"_id" : ObjectId("639a3154a812ec1939dde638"), "total" : 3648 }
"_id" : ObjectId("639a3154a812ec1939dde582"), "total" : 3502 }
"_id" : ObjectId("639a3154a812ec1939dde5ca"), "total" : 3207 }
"_id" : ObjectId("639a3154a812ec1939dde7e8"), "total" : 3198 }
"_id" : ObjectId("639a3154a812ec1939dde4f8"), "total" : 3095 }

```

12. The Query to get the count of the bookings who book the taxi.

Query Code:

```
db.bookings.countDocuments({ payment: { $exists: true } })
```

Output screen print:

```

taxidb> db.bookings.countDocuments({ payment: { $exists: true } })
963

```

Appendix

The steps to import data into the database are as follows.

1. Open your terminal
2. Move into the 'data' directory using the 'cd' command.
3. Run the command 'mongoimport staffs.json -d taxidb -c staffs --jsonArray' to import the staff data into the database.
4. Run the command 'mongoimport revenues.json -d taxidb -c revenues --jsonArray' to import the staff data into the database.
5. Run the command 'mongoimport customers.json -d taxidb -c customers --jsonArray' to import the staff data into the database.
6. Run the command 'mongoimport cars.json -d taxidb -c cars --jsonArray' to import the staff data into the database.

7. Run the command ``mongoimport bookings.json -d taxidb -c bookings --jsonArray`` to import the staff data into the database.
8. Run ``mongosh`` in command line
9. Type ``use taxidb`` to switch the current db to taxidb
10. Run the queries present in the document above or from the ``queries.js`` file which contains all the queries with description.