ECS650/ECS789 Semi-Structured Data and Advanced Data Modelling

Coursework 2

MongoDB Semi-Structured Database Development

School of Electronic Engineering and Computer Science
Queen Mary University of London

Group 64 - MSc
Sarthak Agarwal - 220662176
Sayed Sohail Peerzade - 220541549

To fulfill the requirements of this coursework, a taxi booking system has been designed and built using MongoDb database management system. This report includes the assumptions made during the development, An ER diagram showing the different entities and relationships in the design, the description of how high-level design was mapped to MongoDB collections for the implementation, and 12 queries created to extract information from the system.

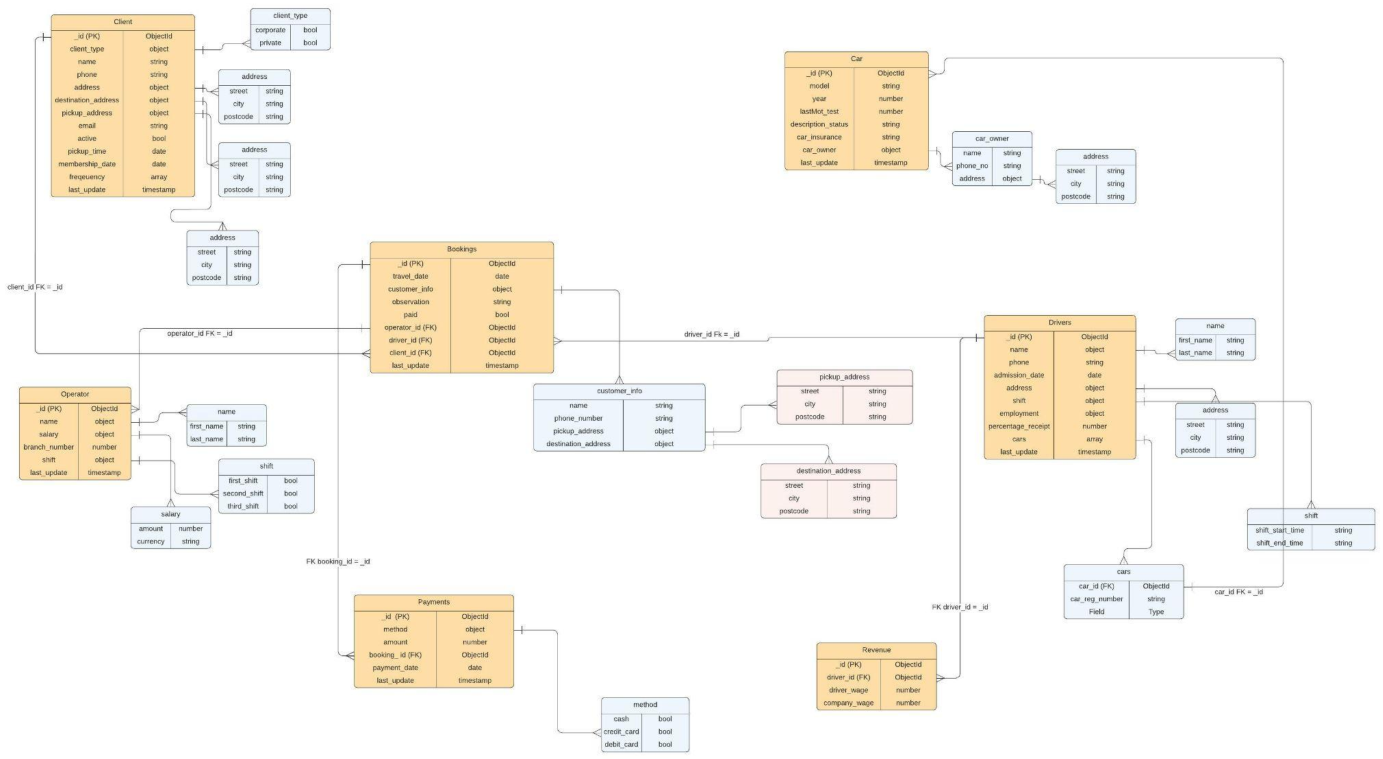
1. Assumptions

The following Assumptions were made for this implementation:

1. Cars are owned by individuals, who may or may not be Taxi Drivers.
2. Cars are then lent out to the Taxi Company.
3. Each Taxi Driver can have several cars.
4. Shift for each Taxi driver contains the start and end time when the driver is available for duty.
5. Clients are of two types Private and Corporate.
6. Operators work in shifts, First, Second and Third Shifts.
7. The Taxi Company receives 40% of the payments made for each booking.

2. Design

The UML diagram below describes the entities and their relationships for the taxi company.



3. MongoDb Database design and Implementation.

The above UML diagram was used to model the NoSQL database. We have divided our database into 7 collections. Each collection fulfills a key feature of the application as mentioned below.

1. Cars Collection

The Cars collection stores information regarding the car and the car owner. We have created a schema to validate the data before insert operation (Refer Appendix 1.1.1). For the Queries used to insert the data please refer Appendix 1.1.2

Cars collection contains the following information:

- Registration Number
- Model
- Year of Manufacturing
- Last MOT Test
- Car Insurance
- Car Owner

Name of the field	Data type	Comment
registration_number	string	
model	string	
year	number	
lastMot_test	number	
description_status	string	Status string should be either roadworthy, in for service or awaiting repair
car_insurance	string	
car_owner	object	This embedded Object field contains the following fields: name (string), phone_no (string), address (object). address object contains : street(string), city (string), postcode (string)
special_features	string	
last_update	timestamp	

2. Drivers Collection

The Drivers collection stores the information regarding the driver details, the employment type and the car he drives. We have created a schema validation to make sure the driver details entered are appropriate. Please refer to Appendix 1.2.1 for creation and schema code. And refer to Appendix 1.2.2 for the insert code.
Key information that can be found from this collection :

- Name
- Phone
- Email
- Admission Date
- Address
- Shift
- List of all cars

Name of the field	Data type	Comments
name	object	This embedded Object field contains the following

		fields: first_name (string), last_name (string)
phone	string	
email	string	This field is being validated and must be of the proper email format
admission_date	date	
address	object	This is a embedded Object field containing the following fields: street (string), city (string), postcode(string)
shift	object	This is an embedded Object field containing the following fields: shift_start_time (string), shift_end_time(string). We are using string values to store time, for correct comparison of those strings in queries these filed should be inserted with two values, ex for 00:00 , 08:00
employment	object	This is an embedded Object field containing the following fields: type (Object), due_date(date). type (Object) further contains: fixed (bool), percentage (bool)
percentage_receipt	number	
cars	array	This Array may contain several objects of the following type: car_id (ObjectId) car_reg_number (string). car_id field is a foreign key that stores the id of the cars which a driver can drive.
last_update	timestamp	

3. Operators Collection

The operator collection stores the information of operators that take bookings and allocate them to drivers. We have created schema validation to ensure only validated data is inserted into the collection, please refer to Appendix 1.3.1 for creation and 1.3.2 for insertion code.
Key information that can be found from this collection :

- Name
- Salary
- Branch Number
- Shift

Name of the field	Data Type	Comments
name	object	This is an embedded Object field containing the following fields: first_name (string), last_name(string)
salary	object	This is an embedded Object field containing the following fields: price (number), currency (string)
branch_number	number	
shift	object	This is an embedded Object field containing the following fields: first_shift (bool), second_shift(bool), third_shift(bool)
last_update	timestamp	

4. Clients Collection.

The clients collection stores the information of registered clients that have regular bookings. We have created schema validation to ensure only validated data is inserted into the collection, please refer to Appendix 1.4.1 for creation and 1.4.2 for insertion code.
Key information that can be found from this collection :

- Client Type – Private or Corporate
- Name
- Email
- Address
- Pickup Address
- Destination Address

- Pickup Time
- Membership Date
- Frequency of Travel

Name of the field	Data Type	Comments
client_type	object	This is an embedded Object field containing the following fields: corporate (bool), private(bool)
name	string	
phone	string	
address	object	This is an embedded Object field containing the following fields: street(string), city (string), postcode (string)
pickup_address	object	This is an embedded Object field containing the following fields: street(string), city (string), postcode (string)
destination_address	object	This is an embedded Object field containing the following fields: street(string), city (string), postcode (string)
pickup_time	date	
email	string	This field is being validated and must be of the proper email format
membership_date	date	
frequency	array	
last_update	timestamp	

5. Bookings Collection

The Bookings collection stores the details taken over phone by an operator while confirming the booking. We have created schema validation to ensure only validated data is inserted into the collection, please refer to Appendix 1.5.1 for creation and 1.5.2

for insertion code. Please insert Drivers and Operators data before proceeding with the Bookings Collection.

Bookings collection stores the following information:

- Contains details of all trips that have been completed or ongoing trips.
- Key information that can be found from this collection :
    - Travel date
    - Customer info
    - Paid or not
    - Operator details if any
    - Driver details.

Name of the field	Data Type	Comments
travel_date	date	
customer_info	object	This is an embedded Object field containing the following fields: name (object), phone_number(string), pickup_address(Object), destination_address(Object). name (object) field contains: first_name (string), last_name(string). pickup_address (object) and destination address (object) fields contain: street (string), city(string), postcode(string) fields.
observation	string	
paid	bool	
operator_id	ObjectId	This field stores the information of the operator who made the booking. It is an objectId field which acts as foreign key to operators collection
driver_id	ObjectId	This field stores the information of the driver for the booking. It is an objectId field which acts as foreign key to drivers

		collection
client_id	ObjectId	This field stores the information of the client for which the booking was made. It is an objectId field which acts as foreign key to Client collection
last_update	timestamp	

6. Payments Collection

The Payments collection stores the information about the payments made to driver for the corresponding bookings.We have created schema validation to ensure only validated data is inserted into the collection, please refer to Appendix 1.6.1 for creation and 1.6.2 for insertion code. Please insert Drivers , Operators and Bookings data before proceeding with the insertions to Payments Collection.
Key information that can be found from this collection :

- Mode of Payment
- Amount
- Date
- Booking Id

Name of the field	Data Type	Comments
method	object	This is an embedded Object field containing the following fields: cash (bool), credit_card (bool), debit_card(bool)
amount	number	
payment_date	date	
booking_Id	ObjectId	This field stores the information of the booking. It is an objectId field which acts as foreign key to Bookings collection
last_update	timestamp	

7. Revenue Collection

The Revenue collection stores the information of the driver and their revenue earned. This collection also stores the amount each driver pays to the company for operational costs.
Key information that can be found from this collection :

- Driver Id
- Driver Wage
- Company Wage

Name of the field	Data Type	Comments
driver_id	ObjectId	It is an objectId field which acts as foreign key to Drivers collection
driver_wage	number	
company_wage	number	

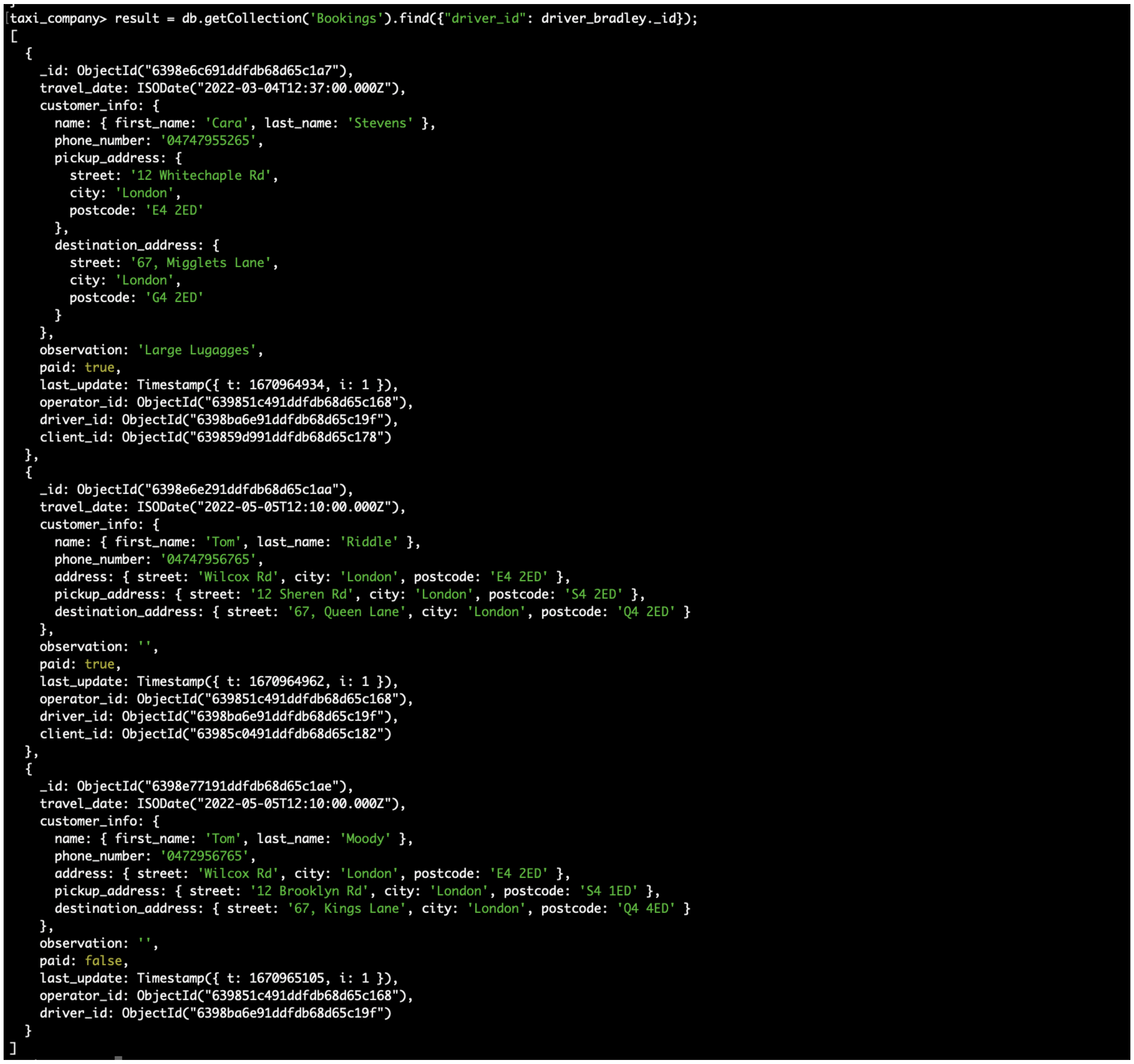
4. Queries.

The following queries were run to demonstrate that the database was designed appropriately for typical taxi company use cases.

1. To find all the bookings made for a specific taxi driver.
   Query:

```
// Query to return all trips from a certain driver
driver_bradley = db.Drivers.findOne({name: {first_name:"Bradley", last_name:"Greer"
}});
result = db.getCollection('Bookings').find({"driver_id": driver_bradley._id});
```

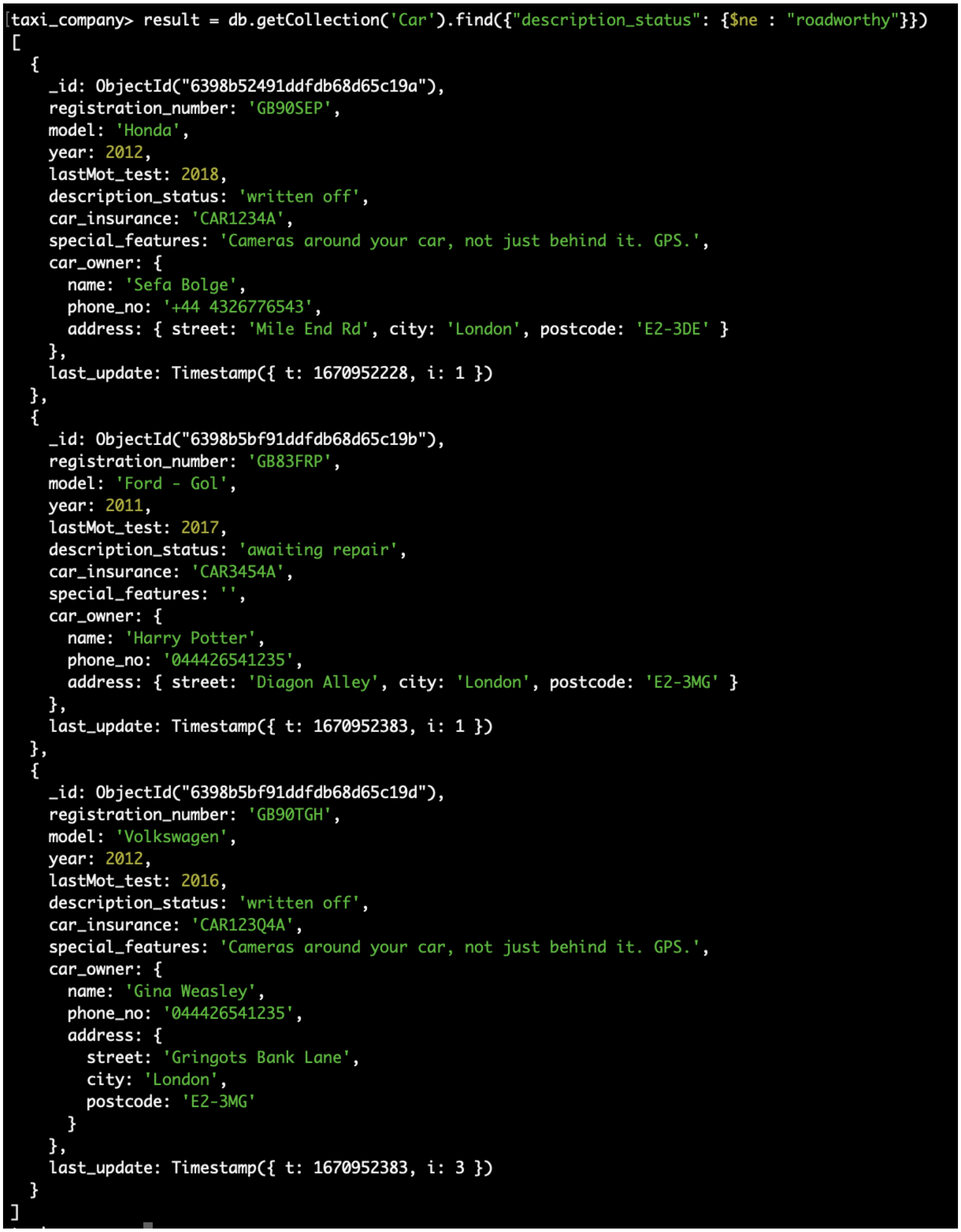
Output:


```
[taxi_company> result = db.getCollection('Bookings').find({"driver_id": driver_bradley._id});
[
  {
    _id: ObjectId("6398e6c691ddfdb68d65c1a7"),
    travel_date: ISODate("2022-03-04T12:37:00.000Z"),
    customer_info: {
      name: { first_name: 'Cara', last_name: 'Stevens' },
      phone_number: '04747955265',
      pickup_address: {
        street: '12 Whitechaple Rd',
        city: 'London',
        postcode: 'E4 2ED'
      },
      destination_address: {
        street: '67, Migglets Lane',
        city: 'London',
        postcode: 'G4 2ED'
      }
    },
    observation: 'Large Lugagges',
    paid: true,
    last_update: Timestamp({ t: 1670964934, i: 1 }),
    operator_id: ObjectId("639851c491ddfdb68d65c168"),
    driver_id: ObjectId("6398ba6e91ddfdb68d65c19f"),
    client_id: ObjectId("639859d991ddfdb68d65c178")
  },
  {
    _id: ObjectId("6398e6e291ddfdb68d65c1aa"),
    travel_date: ISODate("2022-05-05T12:10:00.000Z"),
    customer_info: {
      name: { first_name: 'Tom', last_name: 'Riddle' },
      phone_number: '04747956765',
      address: { street: 'Wilcox Rd', city: 'London', postcode: 'E4 2ED' },
      pickup_address: { street: '12 Sheren Rd', city: 'London', postcode: 'S4 2ED' },
      destination_address: { street: '67, Queen Lane', city: 'London', postcode: 'Q4 2ED' }
    },
    observation: '',
    paid: true,
    last_update: Timestamp({ t: 1670964962, i: 1 }),
    operator_id: ObjectId("639851c491ddfdb68d65c168"),
    driver_id: ObjectId("6398ba6e91ddfdb68d65c19f"),
    client_id: ObjectId("63985c0491ddfdb68d65c182")
  },
  {
    _id: ObjectId("6398e77191ddfdb68d65c1ae"),
    travel_date: ISODate("2022-05-05T12:10:00.000Z"),
    customer_info: {
      name: { first_name: 'Tom', last_name: 'Moody' },
      phone_number: '0472956765',
      address: { street: 'Wilcox Rd', city: 'London', postcode: 'E4 2ED' },
      pickup_address: { street: '12 Brooklyn Rd', city: 'London', postcode: 'S4 1ED' },
      destination_address: { street: '67, Kings Lane', city: 'London', postcode: 'Q4 4ED' }
    },
    observation: '',
    paid: false,
    last_update: Timestamp({ t: 1670965105, i: 1 }),
    operator_id: ObjectId("639851c491ddfdb68d65c168"),
    driver_id: ObjectId("6398ba6e91ddfdb68d65c19f")
  }
]
```

2. A Query to find all cars that are not roadworthy.
Query:

```
result = db.getCollection('Car').find({"description_status": {$ne : "roadworthy"}})
```

Output:



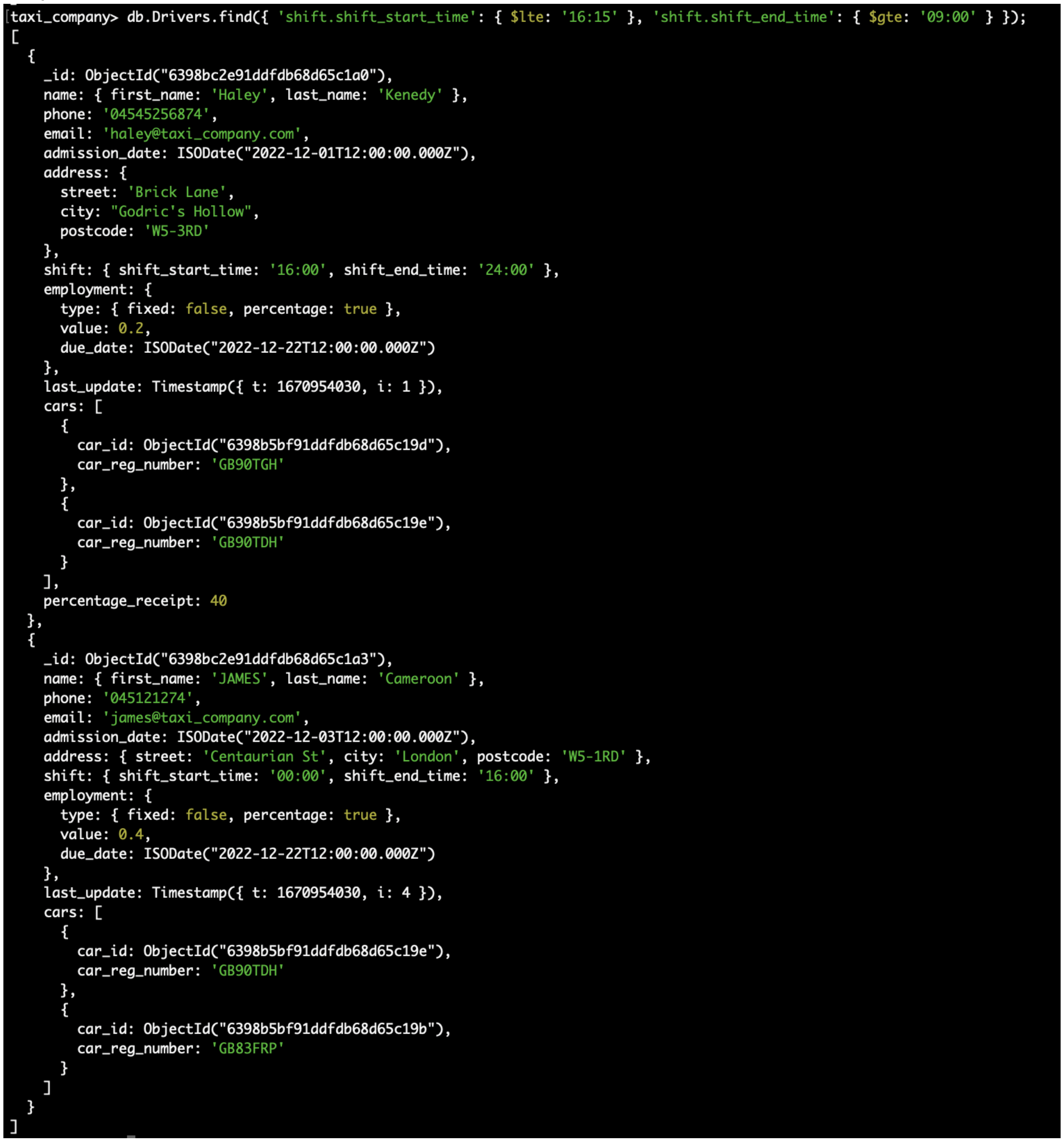
```
[taxi_company> result = db.getCollection('Car').find({"description_status": {$ne : "roadworthy"}})
[
  {
    _id: ObjectId("6398b52491ddfdb68d65c19a"),
    registration_number: 'GB90SEP',
    model: 'Honda',
    year: 2012,
    lastMot_test: 2018,
    description_status: 'written off',
    car_insurance: 'CAR1234A',
    special_features: 'Cameras around your car, not just behind it. GPS.',
    car_owner: {
      name: 'Sefa Bolge',
      phone_no: '+44 4326776543',
      address: { street: 'Mile End Rd', city: 'London', postcode: 'E2-3DE' }
    },
    last_update: Timestamp({ t: 1670952228, i: 1 })
  },
  {
    _id: ObjectId("6398b5bf91ddfdb68d65c19b"),
    registration_number: 'GB83FRP',
    model: 'Ford - Gol',
    year: 2011,
    lastMot_test: 2017,
    description_status: 'awaiting repair',
    car_insurance: 'CAR3454A',
    special_features: '',
    car_owner: {
      name: 'Harry Potter',
      phone_no: '044426541235',
      address: { street: 'Diagon Alley', city: 'London', postcode: 'E2-3MG' }
    },
    last_update: Timestamp({ t: 1670952383, i: 1 })
  },
  {
    _id: ObjectId("6398b5bf91ddfdb68d65c19d"),
    registration_number: 'GB90TGH',
    model: 'Volkswagen',
    year: 2012,
    lastMot_test: 2016,
    description_status: 'written off',
    car_insurance: 'CAR123Q4A',
    special_features: 'Cameras around your car, not just behind it. GPS.',
    car_owner: {
      name: 'Gina Weasley',
      phone_no: '044426541235',
      address: {
        street: 'Gringots Bank Lane',
        city: 'London',
        postcode: 'E2-3MG'
      }
    },
    last_update: Timestamp({ t: 1670952383, i: 3 })
  }
]
```

3. Query to find an available driver for a ride.

Query:

```
db.Drivers.find({ 'shift.shift_start_time': { $lte: '16:15' }, 'shift.shift_end_time':
{ $gte: '09:00' } });
```

Output:



```
taxi_company> db.Drivers.find({ 'shift.shift_start_time': { $lte: '16:15' }, 'shift.shift_end_time': { $gte: '09:00' } });
[
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a0"),
    name: { first_name: 'Haley', last_name: 'Kenedy' },
    phone: '04545256874',
    email: 'haley@taxi_company.com',
    admission_date: ISODate("2022-12-01T12:00:00.000Z"),
    address: {
      street: 'Brick Lane',
      city: "Godric's Hollow",
      postcode: 'W5-3RD'
    },
    shift: { shift_start_time: '16:00', shift_end_time: '24:00' },
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.2,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    },
    last_update: Timestamp({ t: 1670954030, i: 1 }),
    cars: [
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19d"),
        car_reg_number: 'GB90TGH'
      },
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19e"),
        car_reg_number: 'GB90TDH'
      }
    ],
    percentage_receipt: 40
  },
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a3"),
    name: { first_name: 'JAMES', last_name: 'Cameroon' },
    phone: '045121274',
    email: 'james@taxi_company.com',
    admission_date: ISODate("2022-12-03T12:00:00.000Z"),
    address: { street: 'Centaurian St', city: 'London', postcode: 'W5-1RD' },
    shift: { shift_start_time: '00:00', shift_end_time: '16:00' },
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.4,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    },
    last_update: Timestamp({ t: 1670954030, i: 4 }),
    cars: [
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19e"),
        car_reg_number: 'GB90TDH'
      },
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19b"),
        car_reg_number: 'GB83FRP'
      }
    ]
  }
]
```

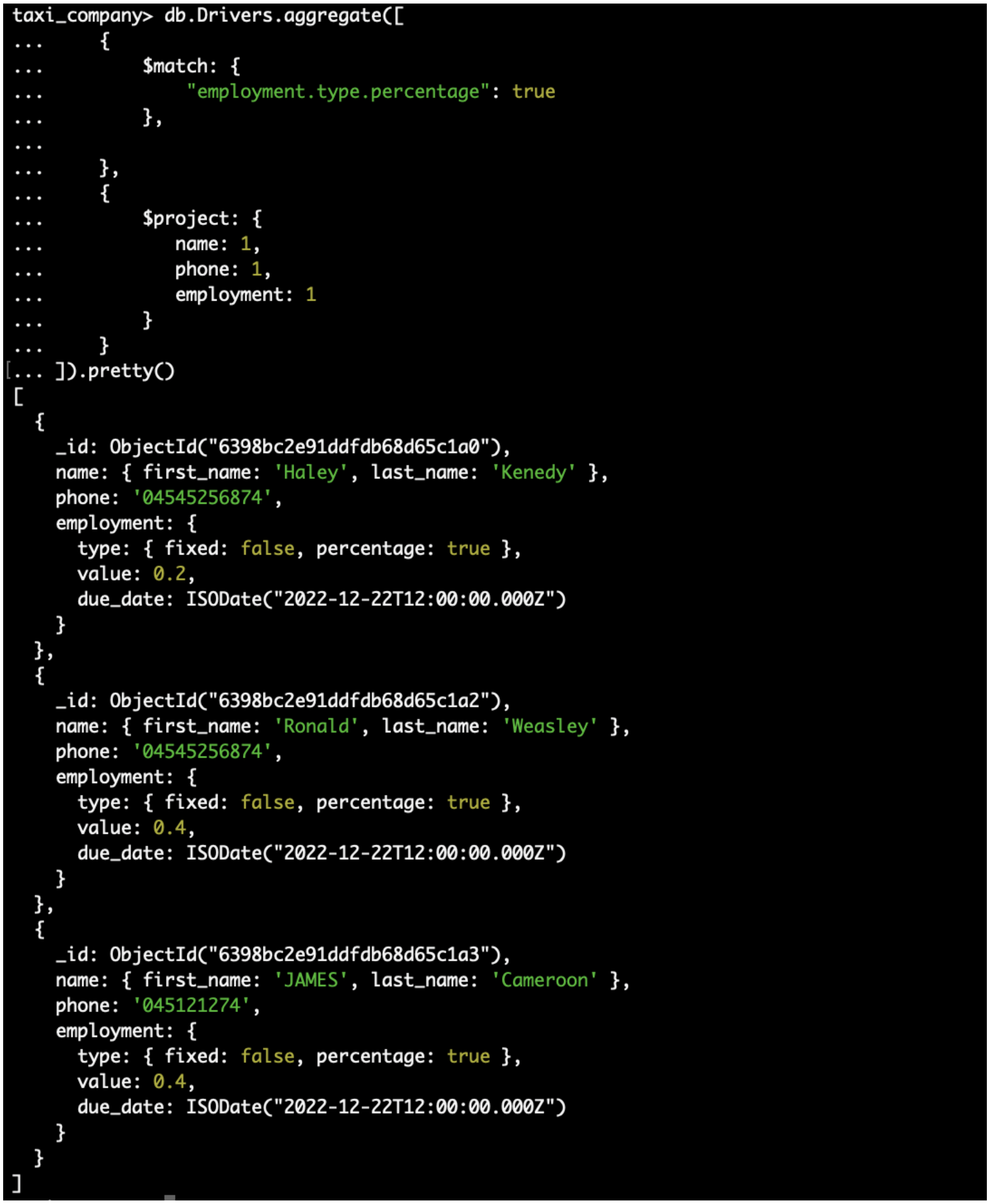
4. Find all drivers with employment type set as fixed and drivers with employment type as percentage.

Query:

```
db.Drivers.aggregate([
   {
       $match: {
           "employment.type.percentage": true
       },


   },
   {
       $project: {
          name: 1,
          phone: 1,
          employment: 1
       }
   }
]).pretty()
```

Output:

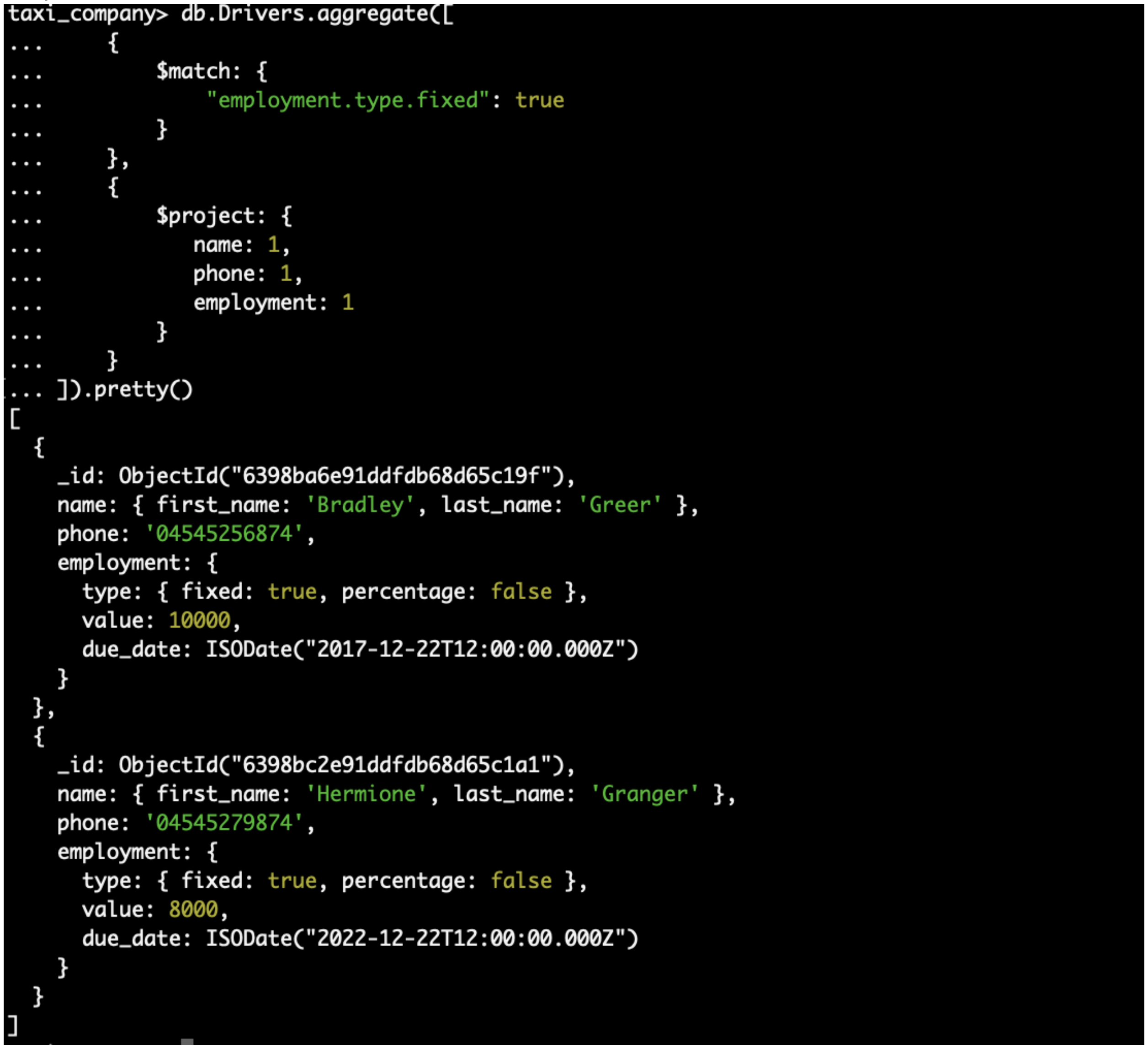


```
taxi_company> db.Drivers.aggregate([
...     {
...         $match: {
...             "employment.type.percentage": true
...         },
...
...     },
...     {
...         $project: {
...            name: 1,
...            phone: 1,
...            employment: 1
...         }
...     }
... ]).pretty()
[
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a0"),
    name: { first_name: 'Haley', last_name: 'Kenedy' },
    phone: '04545256874',
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.2,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    }
  },
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a2"),
    name: { first_name: 'Ronald', last_name: 'Weasley' },
    phone: '04545256874',
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.4,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    }
  },
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a3"),
    name: { first_name: 'JAMES', last_name: 'Cameroon' },
    phone: '045121274',
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.4,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    }
  }
]
```

Query:

```
db.Drivers.aggregate([
   {
       $match: {
           "employment.type.fixed": true
       }
   },
   {
       $project: {
          name: 1,
          phone: 1,
          employment: 1
       }
   }
]).pretty()
```

Output:



```
taxi_company> db.Drivers.aggregate([
...     {
...         $match: {
...             "employment.type.fixed": true
...         }
...     },
...     {
...         $project: {
...            name: 1,
...            phone: 1,
...            employment: 1
...         }
...     }
... ]).pretty()
[
  {
    _id: ObjectId("6398ba6e91ddfdb68d65c19f"),
    name: { first_name: 'Bradley', last_name: 'Greer' },
    phone: '04545256874',
    employment: {
      type: { fixed: true, percentage: false },
      value: 10000,
      due_date: ISODate("2017-12-22T12:00:00.000Z")
    }
  },
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a1"),
    name: { first_name: 'Hermione', last_name: 'Granger' },
    phone: '04545279874',
    employment: {
      type: { fixed: true, percentage: false },
      value: 8000,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    }
  }
]
```

5. Query to get the drivers that are using roadworthy cars. This query uses the aggregation framework to match the roadworthy cars and then performs a lookup on drivers to get the details of the driver driving roadworthy cars.

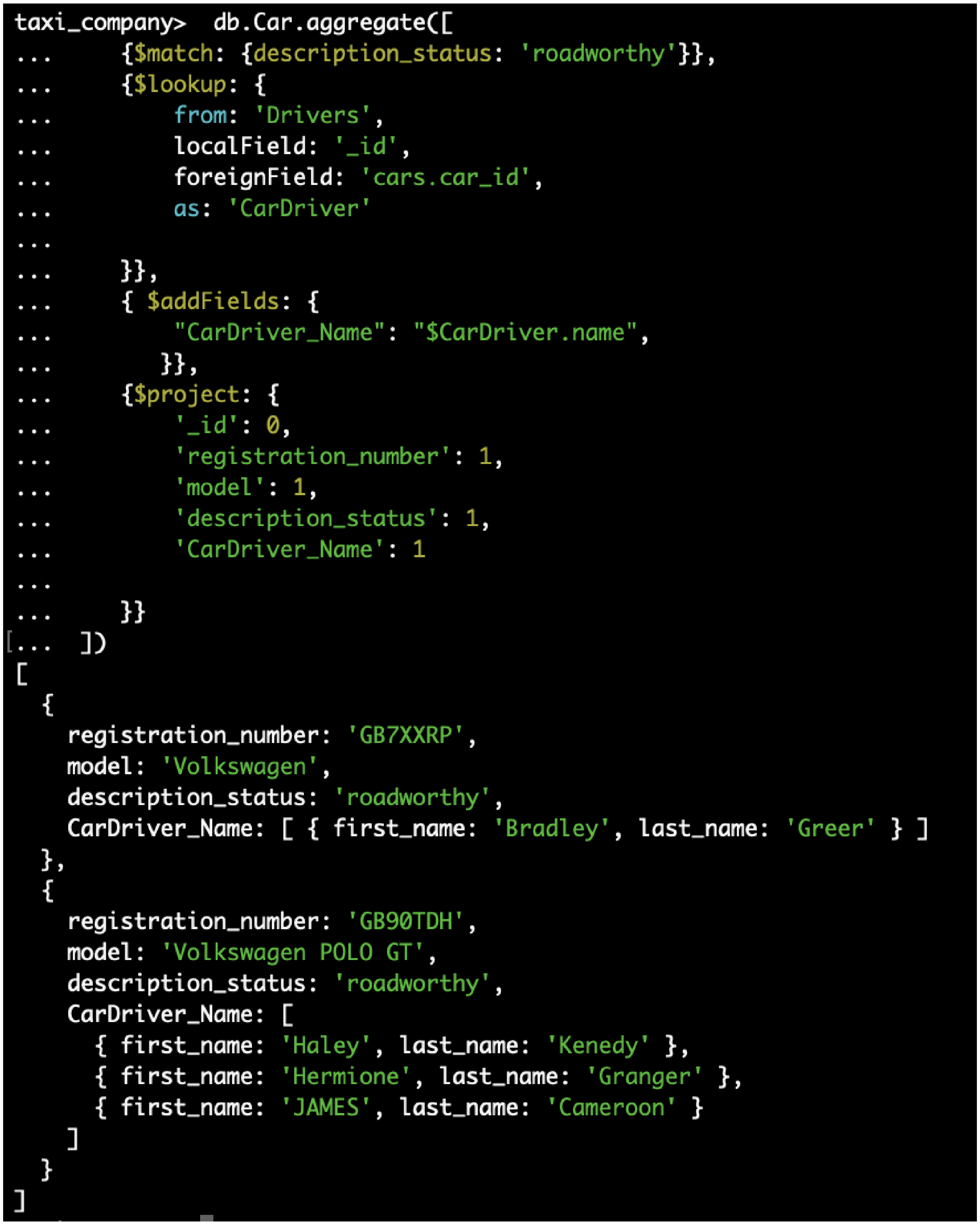
Query:

```
db.Car.aggregate([
   {$match: {description_status: 'roadworthy'}},
   {$lookup: {
       from: 'Drivers',
       localField: '_id',
       foreignField: 'cars.car_id',
       as: 'CarDriver'

   }},
   { $addFields: {
       "CarDriver_Name": "$CarDriver.name",
      }},
   {$project: {
       '_id': 0,
       'registration_number': 1,
       'model': 1,
       'description_status': 1,
       'CarDriver_Name': 1

   }}
])
```

Output:



```
taxi_company>  db.Car.aggregate([
...     {$match: {description_status: 'roadworthy'}},
...     {$lookup: {
...         from: 'Drivers',
...         localField: '_id',
...         foreignField: 'cars.car_id',
...         as: 'CarDriver'
...
...     }},
...     { $addFields: {
...         "CarDriver_Name": "$CarDriver.name",
...        }},
...     {$project: {
...         '_id': 0,
...         'registration_number': 1,
...         'model': 1,
...         'description_status': 1,
...         'CarDriver_Name': 1
...
...     }}
...  ])
[
  {
    registration_number: 'GB7XXRP',
    model: 'Volkswagen',
    description_status: 'roadworthy',
    CarDriver_Name: [ { first_name: 'Bradley', last_name: 'Greer' } ]
  },
  {
    registration_number: 'GB90TDH',
    model: 'Volkswagen POLO GT',
    description_status: 'roadworthy',
    CarDriver_Name: [
      { first_name: 'Haley', last_name: 'Kenedy' },
      { first_name: 'Hermione', last_name: 'Granger' },
      { first_name: 'JAMES', last_name: 'Cameroon' }
    ]
  }
]
```

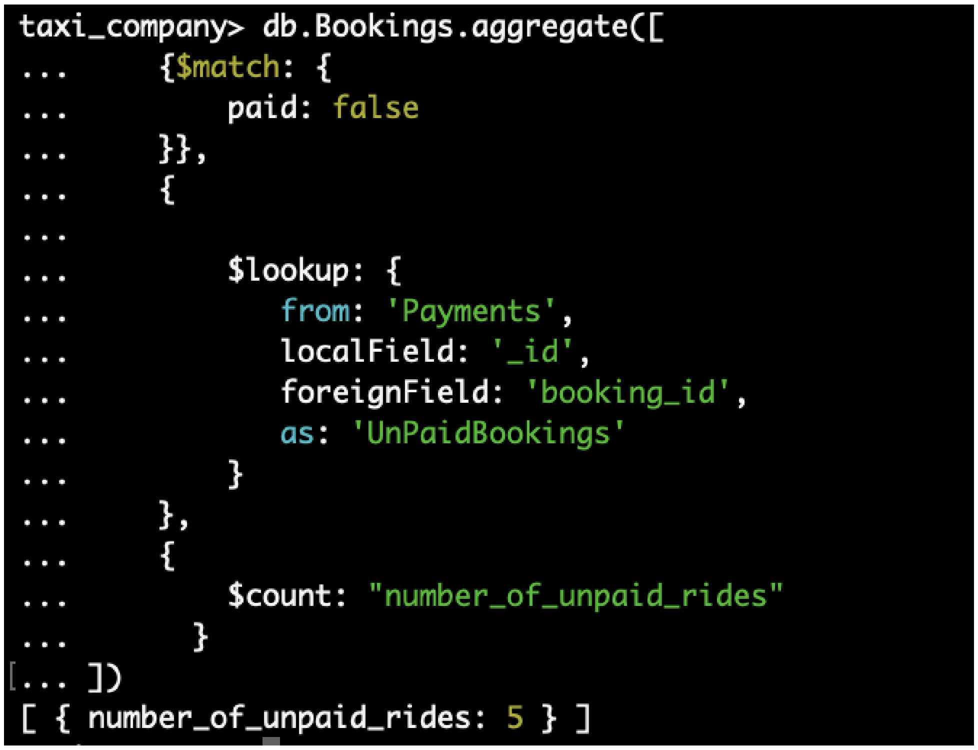
6. Query to find the Number of unpaid rides. For this query we are using the aggregation framework to match the booking.paid field to false and the creating a lookup to payment collection and then counting the number of results.

Query:

```
db.Bookings.aggregate([
   {$match: {
       paid: false
   }},
   {


       $lookup: {
          from: 'Payments',
          localField: '_id',
          foreignField: 'booking_id',
          as: 'UnPaidBookings'
       }
   },
   {
       $count: "number_of_unpaid_rides"
     }
])
```

Output:



```
taxi_company> db.Bookings.aggregate([
...     {$match: {
...         paid: false
...     }},
...     {
...
...         $lookup: {
...            from: 'Payments',
...            localField: '_id',
...            foreignField: 'booking_id',
...            as: 'UnPaidBookings'
...         }
...     },
...     {
...         $count: "number_of_unpaid_rides"
...       }
... ])
[ { number_of_unpaid_rides: 5 } ]
```

7. Query to find the money owed to the company for unpaid rides. This query uses the following aggregation framework functions: Lookup, addFields, unwind, project, group

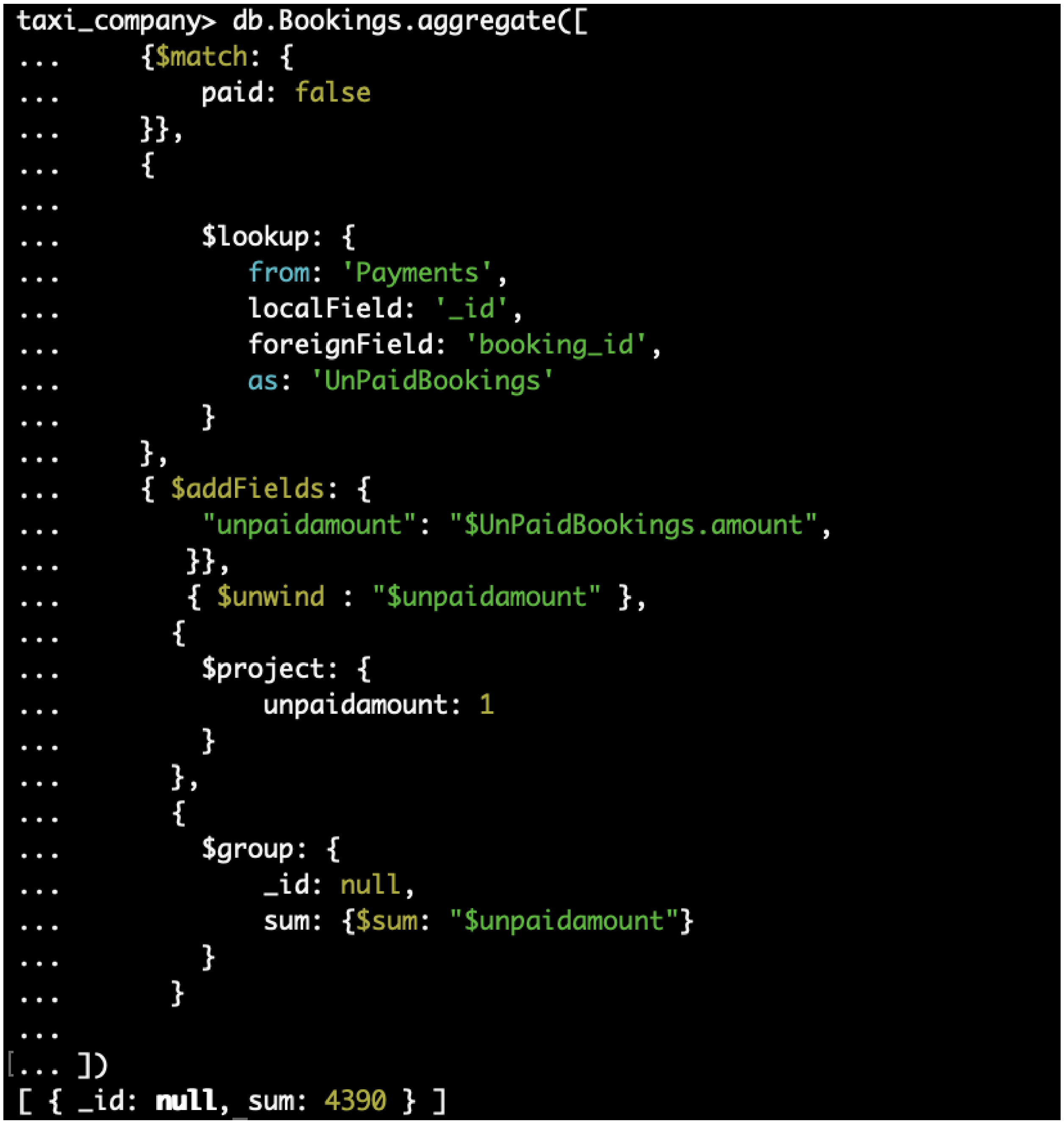
Query:

```
// Money Owed from unpaid rides
db.Bookings.aggregate([
   {$match: {
       paid: false
   }},
   {


       $lookup: {
          from: 'Payments',
          localField: '_id',
          foreignField: 'booking_id',
          as: 'UnPaidBookings'
       }
   },
   { $addFields: {
       "unpaidamount": "$UnPaidBookings.amount",
      }},
      { $unwind : "$unpaidamount" },
     {
       $project: {
           unpaidamount: 1
       }
     },
     {
       $group: {
           _id: null,
           sum: {$sum: "$unpaidamount"}
       }
     }

])
```

Output:

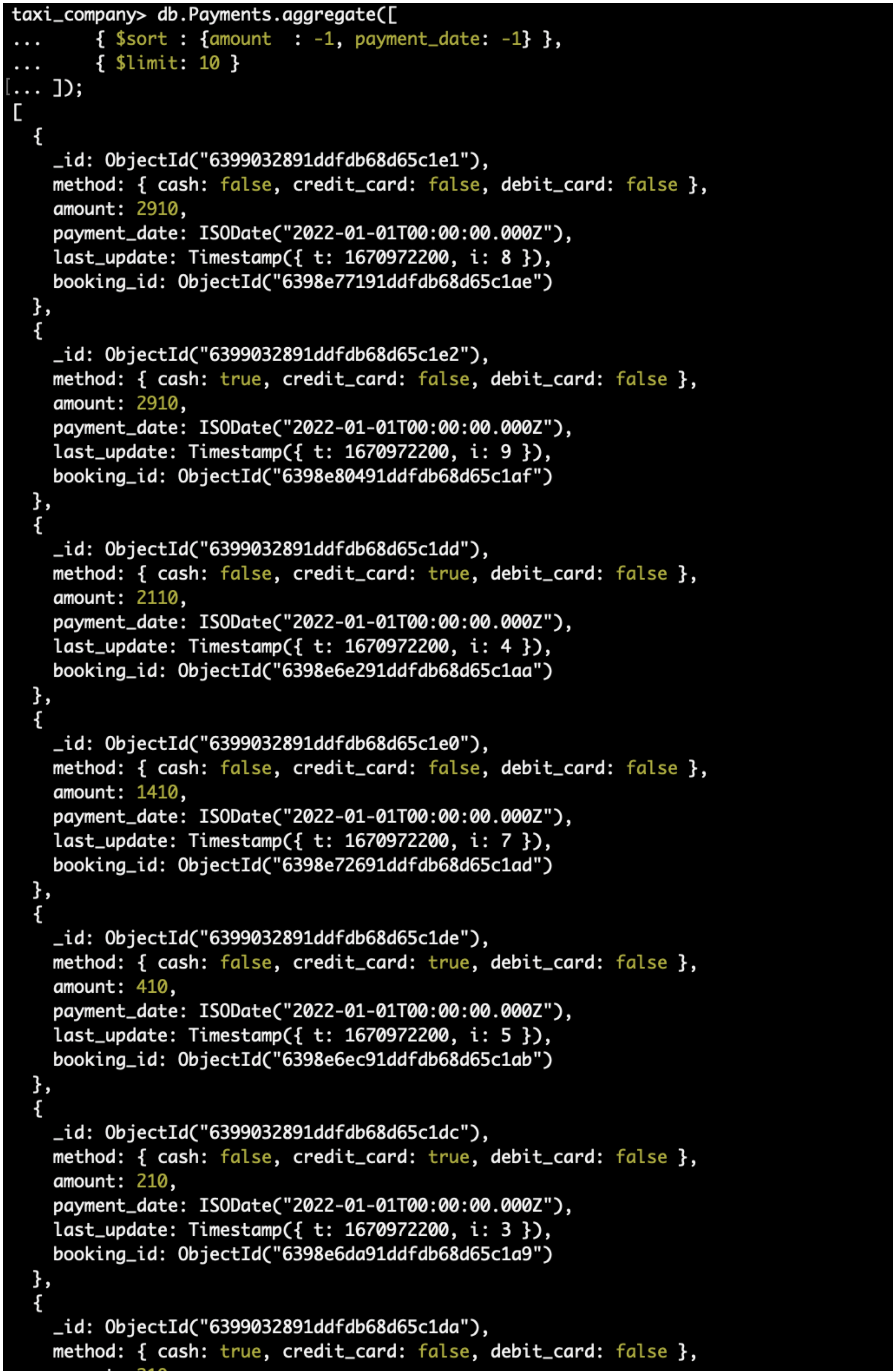

```
taxi_company> db.Bookings.aggregate([
...     {$match: {
...         paid: false
...     }},
...     {
...
...         $lookup: {
...            from: 'Payments',
...            localField: '_id',
...            foreignField: 'booking_id',
...            as: 'UnPaidBookings'
...         }
...     },
...     { $addFields: {
...         "unpaidamount": "$UnPaidBookings.amount",
...        }},
...        { $unwind : "$unpaidamount" },
...       {
...         $project: {
...             unpaidamount: 1
...         }
...       },
...       {
...         $group: {
...             _id: null,
...             sum: {$sum: "$unpaidamount"}
...         }
...       }
...
... ])
[ { _id: null, sum: 4390 } ]
```

8. Query to find the expensive payments / bookings and sorting the result with amount and payment_date. This query uses the sort and limit functionality of the aggregation framework.

Query:

```
db.Payments.aggregate([
   { $sort : {amount  : -1, payment_date: -1} },
   { $limit: 10 }
]);
```

Output:



```
taxi_company> db.Payments.aggregate([
...     { $sort : {amount  : -1, payment_date: -1} },
...     { $limit: 10 }
... ]);
[
  {
    _id: ObjectId("6399032891ddfdb68d65c1e1"),
    method: { cash: false, credit_card: false, debit_card: false },
    amount: 2910,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 8 }),
    booking_id: ObjectId("6398e77191ddfdb68d65c1ae")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1e2"),
    method: { cash: true, credit_card: false, debit_card: false },
    amount: 2910,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 9 }),
    booking_id: ObjectId("6398e80491ddfdb68d65c1af")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1dd"),
    method: { cash: false, credit_card: true, debit_card: false },
    amount: 2110,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 4 }),
    booking_id: ObjectId("6398e6e291ddfdb68d65c1aa")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1e0"),
    method: { cash: false, credit_card: false, debit_card: false },
    amount: 1410,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 7 }),
    booking_id: ObjectId("6398e72691ddfdb68d65c1ad")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1de"),
    method: { cash: false, credit_card: true, debit_card: false },
    amount: 410,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 5 }),
    booking_id: ObjectId("6398e6ec91ddfdb68d65c1ab")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1dc"),
    method: { cash: false, credit_card: true, debit_card: false },
    amount: 210,
    payment_date: ISODate("2022-01-01T00:00:00.000Z"),
    last_update: Timestamp({ t: 1670972200, i: 3 }),
    booking_id: ObjectId("6398e6da91ddfdb68d65c1a9")
  },
  {
    _id: ObjectId("6399032891ddfdb68d65c1da"),
    method: { cash: true, credit_card: false, debit_card: false },
```

9. Query to find the operator and driver for a specific booking

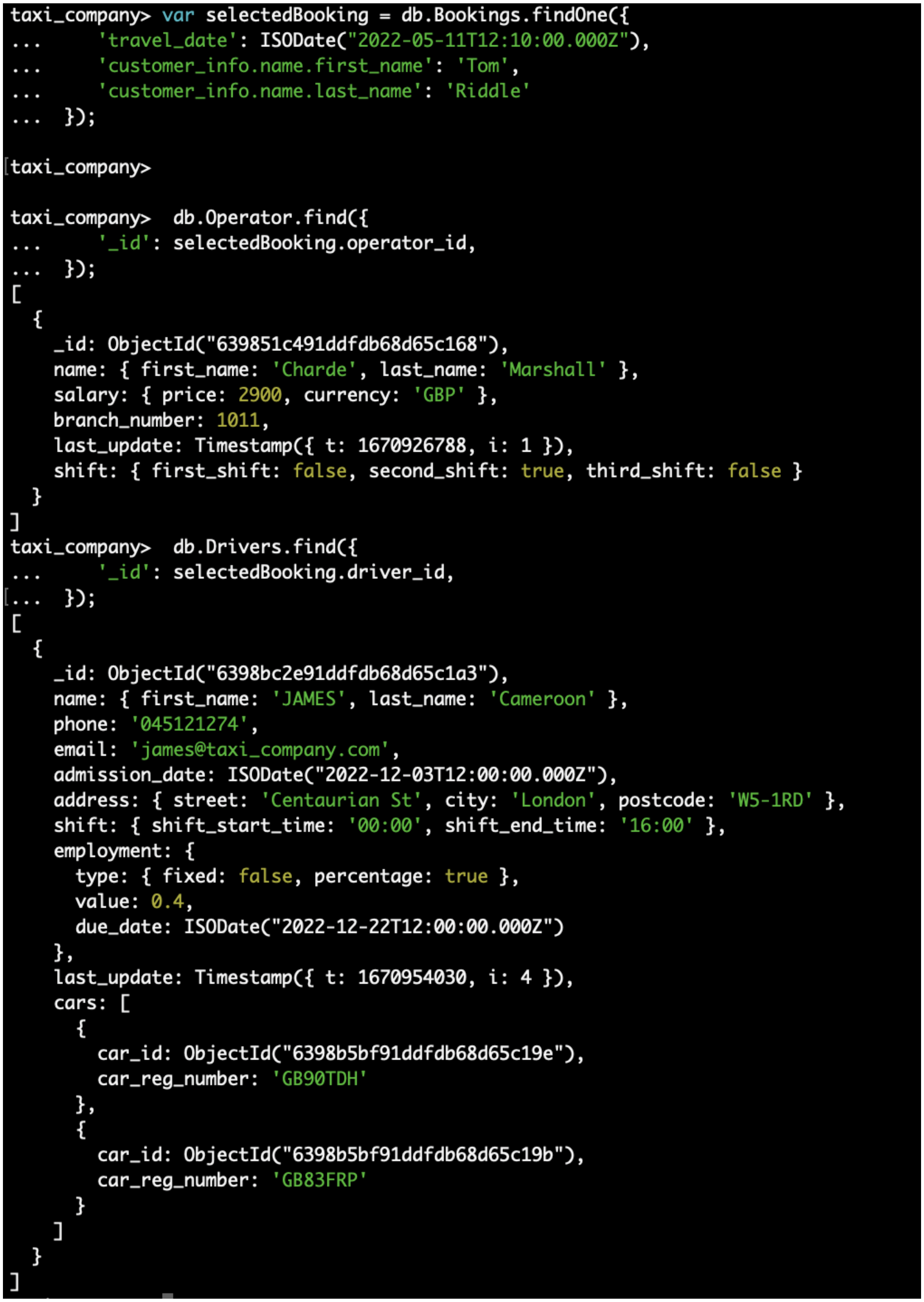
Query:

```
//Find a driver and operator for a booking
// For Example find booking made my Tom Riddle on 11/05/2022
var selectedBooking = db.Bookings.findOne({
   'travel_date': ISODate("2022-05-11T12:10:00.000Z"),
   'customer_info.name.first_name': 'Tom',
   'customer_info.name.last_name': 'Riddle'
});



db.Operator.find({
   '_id': selectedBooking.operator_id,
});
db.Drivers.find({
   '_id': selectedBooking.driver_id,
});
```

Output:



```
taxi_company> var selectedBooking = db.Bookings.findOne({
...     'travel_date': ISODate("2022-05-11T12:10:00.000Z"),
...     'customer_info.name.first_name': 'Tom',
...     'customer_info.name.last_name': 'Riddle'
...  });

taxi_company>

taxi_company>  db.Operator.find({
...     '_id': selectedBooking.operator_id,
...  });
[
  {
    _id: ObjectId("639851c491ddfdb68d65c168"),
    name: { first_name: 'Charde', last_name: 'Marshall' },
    salary: { price: 2900, currency: 'GBP' },
    branch_number: 1011,
    last_update: Timestamp({ t: 1670926788, i: 1 }),
    shift: { first_shift: false, second_shift: true, third_shift: false }
  }
]
taxi_company>  db.Drivers.find({
...     '_id': selectedBooking.driver_id,
...  });
[
  {
    _id: ObjectId("6398bc2e91ddfdb68d65c1a3"),
    name: { first_name: 'JAMES', last_name: 'Cameroon' },
    phone: '045121274',
    email: 'james@taxi_company.com',
    admission_date: ISODate("2022-12-03T12:00:00.000Z"),
    address: { street: 'Centaurian St', city: 'London', postcode: 'W5-1RD' },
    shift: { shift_start_time: '00:00', shift_end_time: '16:00' },
    employment: {
      type: { fixed: false, percentage: true },
      value: 0.4,
      due_date: ISODate("2022-12-22T12:00:00.000Z")
    },
    last_update: Timestamp({ t: 1670954030, i: 4 }),
    cars: [
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19e"),
        car_reg_number: 'GB90TDH'
      },
      {
        car_id: ObjectId("6398b5bf91ddfdb68d65c19b"),
        car_reg_number: 'GB83FRP'
      }
    ]
  }
]
```

10. Query to find the list of all drivers that completed their bookings(rides) and the total amount of payment received for each driver. This Query uses Aggregation framework.

Query:

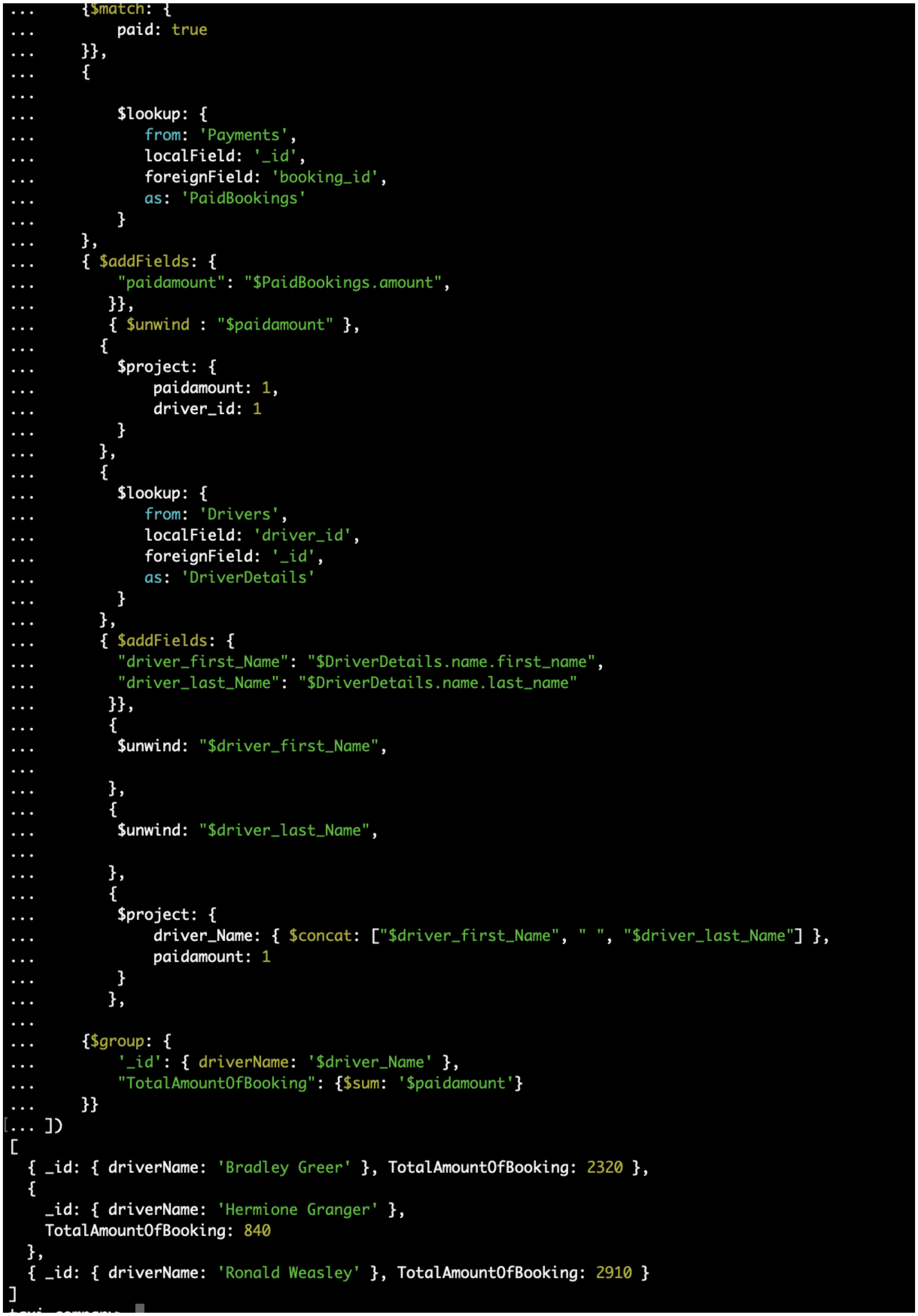
```
// Getting list of all drrivers and their payment
db.Bookings.aggregate([
   {$match: {
       paid: true
   }},
   {

       $lookup: {
          from: 'Payments',
          localField: '_id',
          foreignField: 'booking_id',
          as: 'PaidBookings'
       }
   },
   { $addFields: {
       "paidamount": "$PaidBookings.amount",
      }},
      { $unwind : "$paidamount" },
     {
       $project: {
           paidamount: 1,
           driver_id: 1
       }
     },
     {
       $lookup: {
          from: 'Drivers',
          localField: 'driver_id',
          foreignField: '_id',
          as: 'DriverDetails'
       }
     },
     { $addFields: {
       "driver_first_Name": "$DriverDetails.name.first_name",
       "driver_last_Name": "$DriverDetails.name.last_name"
      }},
      {
       $unwind: "$driver_first_Name",
      },
      {
       $unwind: "$driver_last_Name",

      },
      {
       $project: {
           driver_Name: { $concat: ["$driver_first_Name", " ", "$driver_last_Name"] },
           paidamount: 1
       }
      },

   {$group: {
       '_id': { driverName: '$driver_Name' },
       "TotalAmountOfBooking": {$sum: '$paidamount'}
   }}
])
```

Output:



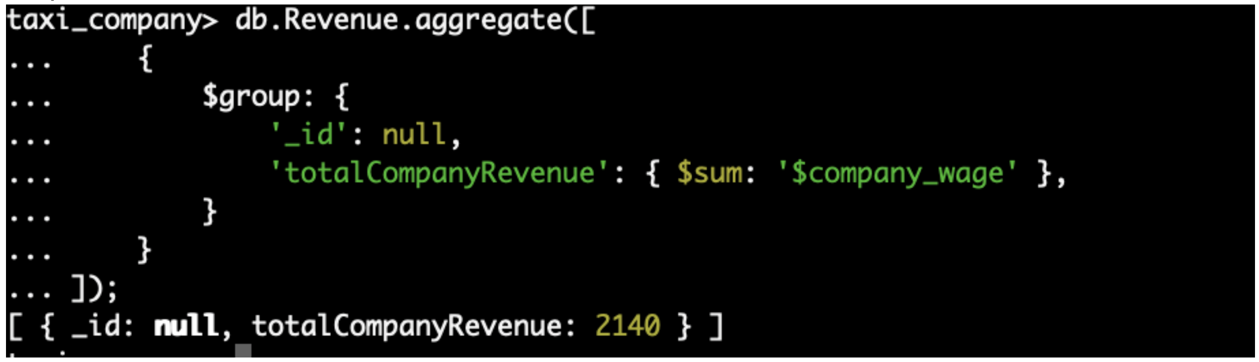
```
...     {$match: {
...         paid: true
...     }},
...     {
...
...         $lookup: {
...            from: 'Payments',
...            localField: '_id',
...            foreignField: 'booking_id',
...            as: 'PaidBookings'
...         }
...     },
...     { $addFields: {
...         "paidamount": "$PaidBookings.amount",
...        }},
...        { $unwind : "$paidamount" },
...       {
...         $project: {
...             paidamount: 1,
...             driver_id: 1
...         }
...       },
...       {
...         $lookup: {
...            from: 'Drivers',
...            localField: 'driver_id',
...            foreignField: '_id',
...            as: 'DriverDetails'
...         }
...       },
...       { $addFields: {
...         "driver_first_Name": "$DriverDetails.name.first_name",
...         "driver_last_Name": "$DriverDetails.name.last_name"
...        }},
...        {
...         $unwind: "$driver_first_Name",
...
...        },
...        {
...         $unwind: "$driver_last_Name",
...
...        },
...        {
...         $project: {
...             driver_Name: { $concat: ["$driver_first_Name", " ", "$driver_last_Name"] },
...             paidamount: 1
...         }
...        },
...
...     {$group: {
...         '_id': { driverName: '$driver_Name' },
...         "TotalAmountOfBooking": {$sum: '$paidamount'}
...     }}
[... ])
[
  { _id: { driverName: 'Bradley Greer' }, TotalAmountOfBooking: 2320 },
  {
    _id: { driverName: 'Hermione Granger' },
    TotalAmountOfBooking: 840
  },
  { _id: { driverName: 'Ronald Weasley' }, TotalAmountOfBooking: 2910 }
]
```

11. Total Revenue Earned by the company.This query uses the aggregation framework.

Query:

```
db.Revenue.aggregate([
    {
        $group: {
            '_id': null,
            'totalCompanyRevenue': { $sum: '$company_wage' },
        }
    }
]);
```

Output:



```
taxi_company> db.Revenue.aggregate([
...     {
...         $group: {
...             '_id': null,
...             'totalCompanyRevenue': { $sum: '$company_wage' },
...         }
...     }
... ]);
[ { _id: null, totalCompanyRevenue: 2140 } ]
```

12. Number of people who paid through credit cards. This query uses the aggregation framework.

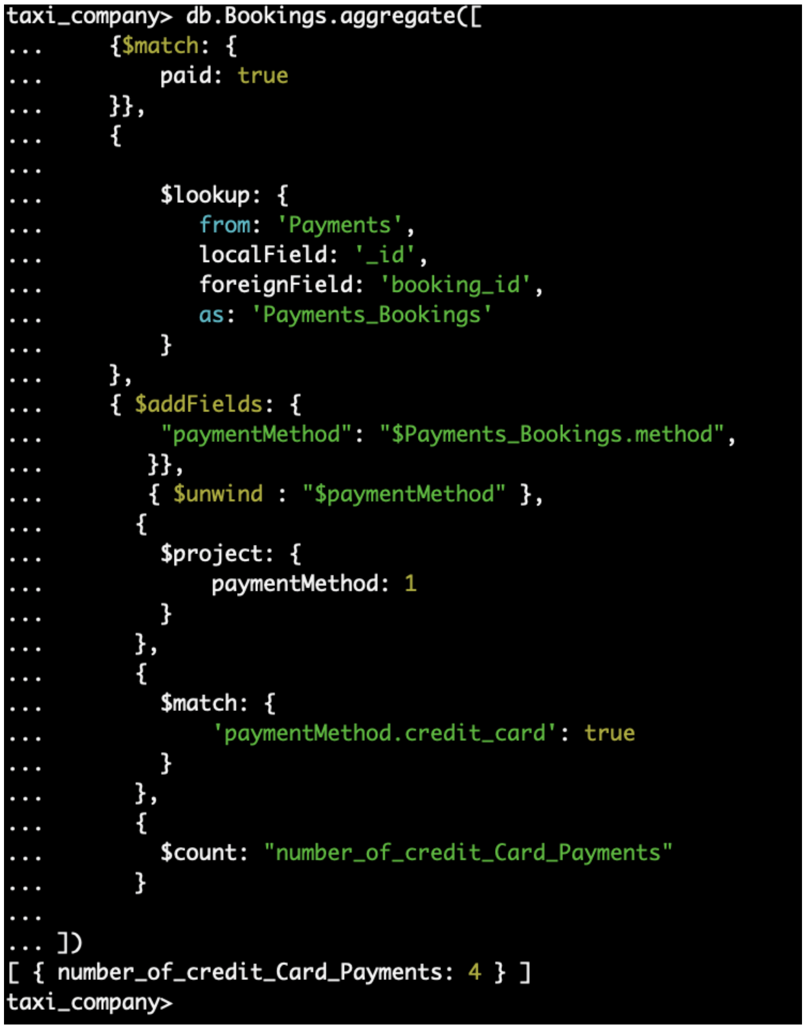
Query:

```
db.Bookings.aggregate([
    {$match: {
        paid: true
    }},
    {


        $lookup: {
           from: 'Payments',
           localField: '_id',
           foreignField: 'booking_id',
           as: 'Payments_Bookings'
        }
    },
    { $addFields: {
        "paymentMethod": "$Payments_Bookings.method",
      }},
      { $unwind : "$paymentMethod" },
     {
       $project: {
           paymentMethod: 1
       }
     },
     {
       $match: {
           'paymentMethod.credit_card': true
       }
     },
     {
       $count: "number_of_credit_Card_Payments"
     }

])
```

Output:

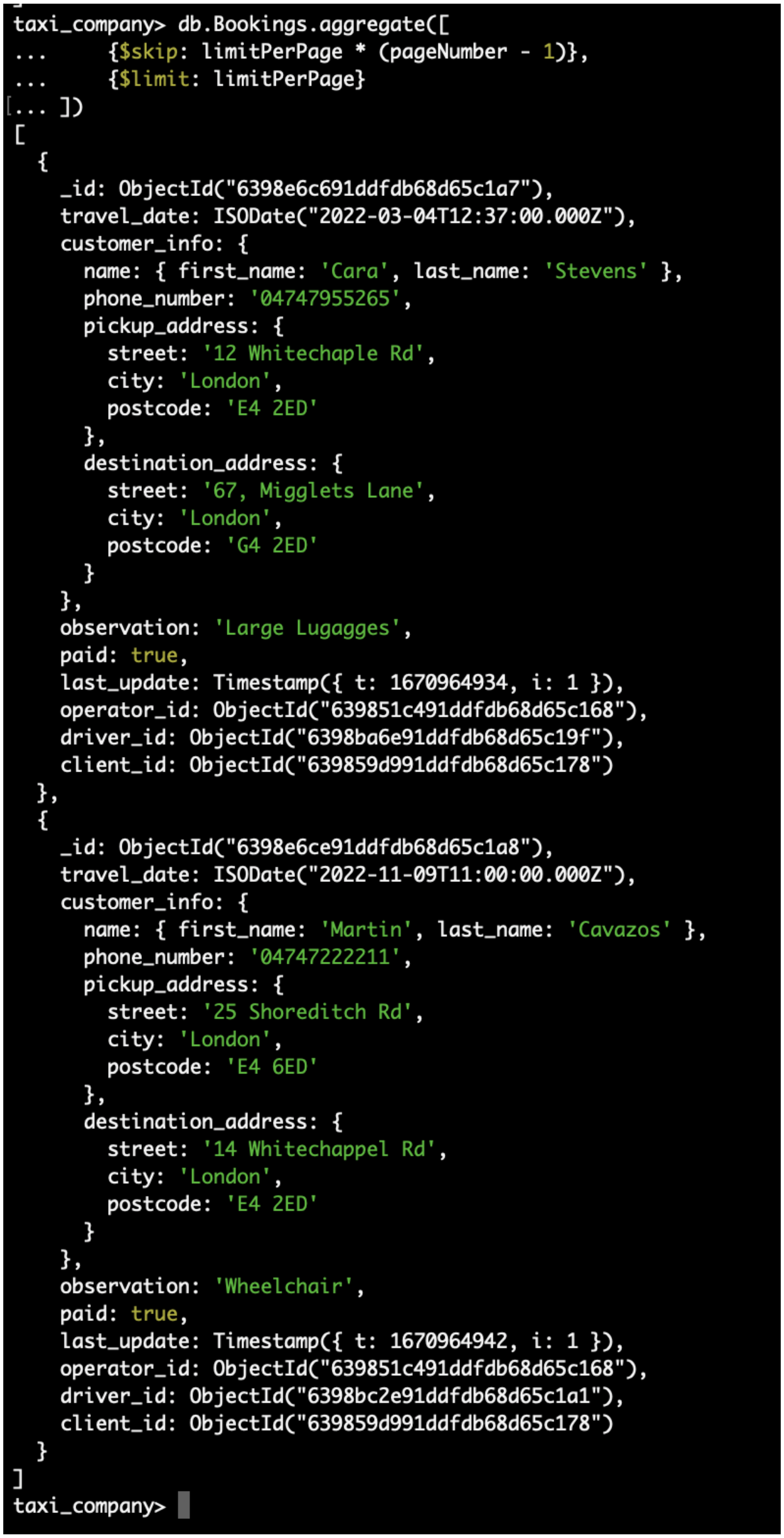

```
taxi_company> db.Bookings.aggregate([
...     {$match: {
...         paid: true
...     }},
...     {
...
...         $lookup: {
...            from: 'Payments',
...            localField: '_id',
...            foreignField: 'booking_id',
...            as: 'Payments_Bookings'
...         }
...     },
...     { $addFields: {
...         "paymentMethod": "$Payments_Bookings.method",
...        }},
...        { $unwind : "$paymentMethod" },
...       {
...         $project: {
...             paymentMethod: 1
...         }
...       },
...       {
...         $match: {
...             'paymentMethod.credit_card': true
...         }
...       },
...       {
...         $count: "number_of_credit_Card_Payments"
...       }
...
... ])
[ { number_of_credit_Card_Payments: 4 } ]
taxi_company>
```

13. Query to implement paging for bookings as booking gets populated with a lot of data.

Query:

```
var pageNumber = 1;
var limitPerPage = 2;
db.Bookings.aggregate([
   {$skip: limitPerPage * (pageNumber - 1)},
   {$limit: limitPerPage}
])
```

Output:



```
taxi_company> db.Bookings.aggregate([
...     {$skip: limitPerPage * (pageNumber - 1)},
...     {$limit: limitPerPage}
... ])
[
  {
    _id: ObjectId("6398e6c691ddfdb68d65c1a7"),
    travel_date: ISODate("2022-03-04T12:37:00.000Z"),
    customer_info: {
      name: { first_name: 'Cara', last_name: 'Stevens' },
      phone_number: '04747955265',
      pickup_address: {
        street: '12 Whitechaple Rd',
        city: 'London',
        postcode: 'E4 2ED'
      },
      destination_address: {
        street: '67, Migglets Lane',
        city: 'London',
        postcode: 'G4 2ED'
      }
    },
    observation: 'Large Lugagges',
    paid: true,
    last_update: Timestamp({ t: 1670964934, i: 1 }),
    operator_id: ObjectId("639851c491ddfdb68d65c168"),
    driver_id: ObjectId("6398ba6e91ddfdb68d65c19f"),
    client_id: ObjectId("639859d991ddfdb68d65c178")
  },
  {
    _id: ObjectId("6398e6ce91ddfdb68d65c1a8"),
    travel_date: ISODate("2022-11-09T11:00:00.000Z"),
    customer_info: {
      name: { first_name: 'Martin', last_name: 'Cavazos' },
      phone_number: '04747222211',
      pickup_address: {
        street: '25 Shoreditch Rd',
        city: 'London',
        postcode: 'E4 6ED'
      },
      destination_address: {
        street: '14 Whitechappel Rd',
        city: 'London',
        postcode: 'E4 2ED'
      }
    },
    observation: 'Wheelchair',
    paid: true,
    last_update: Timestamp({ t: 1670964942, i: 1 }),
    operator_id: ObjectId("639851c491ddfdb68d65c168"),
    driver_id: ObjectId("6398bc2e91ddfdb68d65c1a1"),
    client_id: ObjectId("639859d991ddfdb68d65c178")
  }
]
taxi_company> 
```

Appendix

1.1.1 Create Cars:

```
db.createCollection('Car', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required: ['registration_number', 'model', 'year', 'lastMot_test',
'description_status', 'car_insurance', 'car_owner'],
           properties: {
               registration_number: {
                   bsonType: 'string',
                   description: 'Registration Number is a required field'
               },
               model: {
                   bsonType: 'string',
                   description: 'Car model is a required field'
               },
               year: {
                   bsonType: 'number',
                   description: 'Car Manufacturing Year is a required field and must
be a number'
               },
               lastMot_test: {
                   bsonType: 'number',
                   description: 'Last MOT Test is a required field and must be a
number'
               },
               description_status:{
                   bsonType: 'string',
                  description: 'car status can be either roadworthy, in for service ,
awaiting repair, written off and must be string'
               },
               car_insurance: {
                   bsonType: 'string',
                   description: 'Car insurance is a required field'
               },
               special_features: {
                   bsonType: 'string'
               },
               car_owner: {
                   bsonType: 'object',
                   required: ['name', 'phone_no', 'address'],
                   description: 'Car owner details include Name , Phone_NO, Address',
                   properties: {
                       name: {
                           bsonType: 'string',
                           description: 'Car owner name is a required field'
                       },
                       phone_no: {
                           bsonType: 'string',
                           description: 'Car owner Phone Number is a required field
and is a string'
                       },
                       address: {
                           bsonType: 'object',
                           required: ['street', 'city', 'postcode'],
                           properties: {
                               street: {
                                   bsonType: 'string',
                                   description: 'Street must be a string'
                               },
                               city: {
                                   bsonType: 'string',
                                   description: 'City must be a string'
                               },
                               postcode: {
                                   bsonType: 'string',
                                   minLength: 1,
                                   maxLength: 6,
                                   description: 'Postcode must be a string'
                               }
                           }
                       }
                   }
               },
               last_update:{
                   bsonType: 'timestamp',
                   description: 'last_update must be a timestamp'
               }

           }
       }
```

```
   }
})
```

1.1.2 Insert Data Into Cars Collection:

```
db.Car.insertOne({
       "registration_number": "GB90SEP",
       "model": "Honda",
       "year": 2012,
       "lastMot_test": 2018,
       "description_status": "written off",
       "car_insurance": "CAR1234A",
       "special_features": "Cameras around your car, not just behind it. GPS.",
       "car_owner": {
           "name": "Sefa Bolge",
           "phone_no": "+44 4326776543",
           "address": {
               "street": "Mile End Rd",
               "city": "London",
               "postcode": "E2-3DE"
           }
       },
       "last_update": new Timestamp()
 })

db.Car.insertMany([
   {
       "registration_number": "GB83FRP",
       "model": "Ford - Gol",
       "year": 2011,
       "lastMot_test": 2017,
       "description_status": "awaiting repair",
       "car_insurance": "CAR3454A",
       "special_features": "",
       "car_owner": {
           "name": "Harry Potter",
           "phone_no": "044426541235",
           "address": {
               "street": "Diagon Alley",
               "city": "London",
               "postcode": "E2-3MG"
           }
       },
       "last_update": new Timestamp()
   },
   {
       "registration_number": "GB7XXRP",
       "model": "Volkswagen",
       "year": 2009,
       "lastMot_test": 2017,
       "description_status": "roadworthy",
       "car_insurance": "CAR1234QA",
       "special_features": "",
       "car_owner": {
           "name": "Jesus Garza",
           "phone_no": "044423141299",
           "address": {
               "street": "Sherren House",
               "city": "London",
               "postcode": "E1-5AF"
           }
       },
       "last_update": new Timestamp()
   },
   {
       "registration_number": "GB90TGH",
       "model": "Volkswagen",
       "year": 2012,
       "lastMot_test": 2016,
       "description_status":"written off",
       "car_insurance": "CAR123Q4A",
       "special_features": "Cameras around your car, not just behind it. GPS.",
       "car_owner": {
           "name": "Gina Weasley",
           "phone_no": "044426541235",
           "address": {
               "street": "Gringots Bank Lane",
               "city": "London",
               "postcode": "E2-3MG"
           }
       },
       "last_update": new Timestamp()
   },
   {
       "registration_number": "GB90TDH",
       "model": "Volkswagen POLO GT",
       "year": 2012,
       "lastMot_test": 2016,
       "description_status":"roadworthy",
       "car_insurance": "CARQ1234A",
       "special_features": "Cameras around your car, not just behind it. GPS.",
       "car_owner": {
           "name": "Gina Weasley",
           "phone_no": "044426541235",
           "address": {
               "street": "Gringots Bank Lane",
               "city": "London",
               "postcode": "E2-3MG"
           }
       },
       "last_update": new Timestamp()
   }


])
```

1.2.1 Create Drivers Collection:

```
db.createCollection('Drivers', {
   validator: {
       $jsonSchema: {
           bsonType: 'object',
           required: ['name', 'phone', 'email',  'admission_date', 'address', 'shift',
'last_update', 'cars'],
           properties: {
               name: {
                   bsonType: 'object',
                   required: ['first_name', 'last_name'],
                   properties: {
                       first_name: {
                           bsonType: 'string',
                           description: 'First Name is required and must be a string'
```

```
                        },
                        last_name: {
                            bsonType: 'string',
                            description: 'Last Name is required and must be a string'
                        }
                    }
                },
                phone: {
                    bsonType: 'string',
                    description: 'Phone Number is a required field and must be a
string'
                },
                email: {
                    bsonType: 'string',
                    minLength: 6,
                    maxLength: 40,
                    pattern:
'[a-z0-9._%+!$&*=^|~#%{}/-]+@([a-z0-9-]+.){1,}([a-z]{2,22})',
                    description: 'It is required and it must be a string with length
between 6 and 40 (regular expression pattern)'
                },
                admission_date: {
                    bsonType: 'date',
                    description: 'Admission must be a valid date'
                },
                address: {
                    bsonType: 'object',
                    required: ['street', 'city', 'postcode'],
                    properties: {
                        street: {
                            bsonType: 'string',
                            description: 'Street must be a string'
                        },
                        city: {
                            bsonType: 'string',
                            description: 'City must be a string'
                        },
                        postcode: {
                            bsonType: 'string',
                            minLength: 1,
                            maxLength: 6,
                            description: 'Postcode must be a string with length between
1 and 6'
                        }

                    }
                },
                shift: {
                    bsonType: 'string',
                    properties: {
                        shift_start_time: {
                            bsonType: 'string',
                            description: 'Shift start time is stored as a string'
                        },
                        shift_end_time: {
                            bsonType: 'string',
                            description: 'Shift end time is stored as a string'
                        }
                    }
                },
                employment: {
                    bsonType:['object', 'null'],
                    properties: {
                        type: {
                            bsonType: ['object'],
                            required: ['fixed', 'percentage'],
                            properties: {
                                fixed: {
                                    bsonType: 'bool',
                                    description: 'Can be true or false'
                                },
                                percentage: {
                                    bsonType: 'bool',
                                    description: 'Can be true or false'
                                }
                            }
                        },
                        due_date: {
                            bsonType: 'date',
                            description: 'Due Date must be a valid date'
                        }
                    }
                },
                last_update:{
                    bsonType: 'timestamp',
                    description: 'last_update must be a timestamp'
                },
                percentage_receipt:{
                    bsonType: ['number', 'null'],
                    description: 'Percentage the driver receives, if there is no such
field then the driver receives fixed salary'
                },
                cars: {
                    bsonType: 'array',
                    description: 'must be an array and is required',
                    items: {
                        bsonType: 'object',
                        required: ['car_id', 'car_reg_number'],
                        properties: {
                            car_id: {
                            bsonType: 'objectId',
                            description: 'must be a objectid and is required'
                          },
                          car_reg_number: {
                            bsonType: 'string',
                            description: 'must be an  string and is required'
                          }
                        }
                      }
                }
            }
        }
    },
  validationAction: 'warn'
});
```

1.2.2 Insert Data Into Drivers Collection:

```
var carsList = db.Car.find({},{_id:1,registration_number: 1}).toArray()
db.Drivers.insertOne({
   "name":
   {
       "first_name":"Bradley",
       "last_name":"Greer"
```

```
    },
    "phone":"04545256874",
    "email":"bradley@taxi_company.com",
    "admission_date":ISODate("2022-12-01T12:00:00"),
    "address":
    {
        "street":"Winterfell Lane",
        "city":"Godric's Hollow",
        "postcode":"W2-3RD"
    },
    "shift":
    {
        shift_start_time: '00:00',
        shift_end_time: '08:00'
        },
    "employment":{
        "type" : {
            "fixed": true,
            "percentage": false
        },
        "value" : 10000.000,
        "due_date" : ISODate("2022-12-22T12:00:00")
    },
    "last_update": new Timestamp(),
    "cars": [{
        "car_id":carsList[0]._id,
        "car_reg_number": carsList[0].registration_number
    },{
        "car_id": carsList[1]._id,
        "car_reg_number": carsList[1].registration_number
    }

        ]
})

db.Drivers.insertMany([
    {
        "name":
        {
          "first_name":"Haley",
          "last_name":"Kenedy"
        },
        "phone":"04545256874",
        "email":"haley@taxi_company.com",
        "admission_date":ISODate("2022-12-01T12:00:00"),
        "address":
        {
          "street":"Brick Lane",
          "city":"Godric's Hollow",
          "postcode":"W5-3RD"
        },
        "shift":
        {
            "shift_start_time": '16:00',
            "shift_end_time": '24:00'
        },
        "employment":
        {
            "type" :
            {
                "fixed": false,
                "percentage": true
            },
            "value" : 0.2,
            "due_date" : ISODate("2022-12-22T12:00:00")
        },
        "last_update": new Timestamp(),
        "cars": [{
            "car_id": carsList[1]._id,
            "car_reg_number": carsList[1].registration_number
        },{
            "car_id":  carsList[2]._id,
            "car_reg_number": carsList[2].registration_number
        }


        ],
        "percentage_receipt": 40
    },
      {
        "name":
        {
          "first_name":"Hermione",
          "last_name":"Granger"
        },
        "phone":"04545279874",
        "email":"hermione@taxi_company.com",
        "admission_date":ISODate("2022-12-03T12:00:00"),
        "address":
        {
          "street":"Muggles Rd",
          "city":"Godric's Hollow",
          "postcode":"G5-6TD"
        },
        "shift":
        {
            "shift_start_time": '8:00',
            "shift_end_time": '16:00'
        },
        "employment":
        {
            "type" :
            {
                "fixed": true,
                "percentage": false
            },
            "value" : 8000,
            "due_date" : ISODate("2022-12-22T12:00:00")
        },
        "last_update": new Timestamp(),
        "cars": [{
            "car_id": carsList[2]._id,
            "car_reg_number": carsList[2].registration_number
        },{
            "car_id": carsList[3]._id,
            "car_reg_number":carsList[3].registration_number
        }

        ],
        "percentage_receipt": 40

    },
      {
        "name":
        {
          "first_name":"Ronald",
          "last_name":"Weasley"
        },
        "phone":"04545256874",
        "email":"ronald@taxi_company.com",
        "admission_date":ISODate("2022-12-03T12:00:00"),
        "address":
        {
          "street":"Centaurian St",
          "city":"London",
          "postcode":"W5-3RD"
        },
        "shift":
        {
            "shift_start_time": '8:00',
            "shift_end_time": '16:00'
        },
        "employment":
        {
            "type" :
            {
                "fixed": false,
                "percentage": true
            },
            "value" : 0.4,
            "due_date" : ISODate("2022-12-22T12:00:00")
        },
        "last_update": new Timestamp(),
        "cars": [{
            "car_id": carsList[3]._id,
            "car_reg_number": carsList[3].registration_number
        },{
            "car_id": carsList[4]._id,
            "car_reg_number": carsList[4].registration_number
        }

        ]
    },
      {
        "name":
        {
          "first_name":"JAMES",
          "last_name":"Cameroon"
        },
        "phone":"045121274",
        "email":"james@taxi_company.com",
        "admission_date":ISODate("2022-12-03T12:00:00"),
        "address":
        {
          "street":"Centaurian St",
          "city":"London",
          "postcode":"W5-1RD"
        },
        "shift":
        {
            "shift_start_time": '00:00',
            "shift_end_time": '16:00'
        },
        "employment":
        {
            "type" :
            {
                "fixed": false,
                "percentage": true
            },
            "value" : 0.4,
            "due_date" : ISODate("2022-12-22T12:00:00")
        },
        "last_update": new Timestamp(),
        "cars": [{
            "car_id": carsList[4]._id,
            "car_reg_number": carsList[4].registration_number
        },{
            "car_id": carsList[1]._id,
            "car_reg_number": carsList[1].registration_number
        }

        ]
    }
 ])
```

1.3.1 Create Operators Collection:

```
db.createCollection('Operator', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required:['name', 'salary', 'branch_number', 'last_update', 'shift'],
           properties: {
               name: {
                   bsonType: 'object',
                   required: ['first_name', 'last_name'],
                   properties: {
                       first_name: {
                           bsonType: 'string',
                           description: 'First Name is required and must be a string'
                       },
                       last_name: {
                           bsonType: 'string',
                           description: 'Last Name is required and must be a string'
                       }
                   }
               },
               salary: {
                   bsonType: 'object',
                   required: ['price', 'currency'],
                   properties: {
                       price: {
                           bsonType: 'number',
                           description: 'Salary Amount is a required field and must be
a number'
                       },
                       currency: {
                           bsonType: 'string',
                           description: 'Currency is a required field and must be a
string'
                       }
                   }

               },
               branch_number: {
                   bsonType: 'number',
                   description: "Branch Number is a required field and must be a
number"
               },
               'last_update': {
                   bsonType: 'timestamp',
                   description: 'last_update must be a timestamp'
               },
               shift: {
                   bsonType: 'object',
                   properties: {
                       first_shift: {
                           bsonType: 'bool',
                           description: 'First Shift can only be true or false'
                       },
                       second_shift: {
                           bsonType: 'bool',
                           description: 'Second Shift can only be true or false'
                       },
                       third_shift: {
                           bsonType: 'bool',
                           description: 'Third Shift can only be true or false'
                       },

                   }
               }
           }
       }
}
}
)
```

1.3.2 Insert Data Into Operators Collection:

```
db.Operator.insertOne(
   {
       "name": {
           "first_name": "Charde",
           "last_name": "Marshall"
       },
       "salary": {
           "price": 2900,
           "currency": "GBP"
       },
       "branch_number": 1011,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":false,
         "second_shift":true,
         "third_shift": false
       },
   }
);

db.Operator.insertMany([
   {
       "name": {
           "first_name": "Garret",
           "last_name": "Winter"
       },
       "salary": {
           "price": 3500,
           "currency": "GBP"
       },
       "branch_number": 1012,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":true,
         "second_shift":false,
         "third_shift": false
       }
   },
   {
       "name": {
           "first_name": "Gavin",
           "last_name": "Cortez"
       },
       "salary": {
           "price": 3000,
           "currency": "GBP"
       },
       "branch_number": 1013,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":false,
         "second_shift":true,
         "third_shift": false
       },
   },
   {
       "name": {
           "first_name": "Finn",
           "last_name": "Rios"
       },
       "salary": {
           "price": 2300,
           "currency": "GBP"
       },
       "branch_number": 1014,
       "last_update":new Timestamp(),
       "shift":
       {
         "first_shift":false,
         "second_shift":false,
         "third_shift": true
       },
   },
   {
       "name": {
           "first_name": "Gavin",
           "last_name": "Joyce"
       },
       "salary": {
           "price": 2500,
           "currency": "GBP"
       },
       "branch_number": 1015,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":true,
         "second_shift":false,
         "third_shift": false
       },
   },
   {
       "name": {
           "first_name": "Gabriel",
           "last_name": "Sousa"
       },
       "salary": {
           "price": 2600,
           "currency": "GBP"
       },
       "branch_number": 1016,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":false,
         "second_shift":true,
         "third_shift": false
       },
   },
   {
       "name": {
           "first_name": "Josef",
           "last_name": "Haskell"
       },
       "salary": {
           "price": 2800,
           "currency": "GBP"
       },
       "branch_number": 1017,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":false,
         "second_shift":false,
         "third_shift": true
       },
   },
   {
       "name": {
           "first_name": "Jone",
           "last_name": "Weak"
       },
       "salary": {
           "price": 3100,
           "currency": "GBP"
```

```
       },
       "branch_number": 1012,
       "last_update": new Timestamp(),
       "shift":
       {
         "first_shift":true,
         "second_shift":false,
         "third_shift": false
       },
   }
]);
```

1.4.1 Create Clients Collection:

```
db.createCollection('Clients', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required: ['client_type', 'name', 'phone', 'address', 'pickup_address',
'destination_address', 'pickup_time', 'email', 'active', 'membership_date',
'frequency', 'last_update'],
           properties: {
               client_type: {
                   bsonType: 'object',
                   required: ['corporate', 'private'],
                   properties: {
                       corporate: {
                           bsonType: 'bool',
                           description: 'corporate is a required Boolean field'
                       },
                       private: {
                           bsonType: 'bool',
                           description: 'private is a required Boolean field'
                       },

                   }
               },
               name: {
                   bsonType: 'string',
                   description: 'Client Name Must be a string and is a required field'
               },
                phone: {
                    bsonType: 'string',
                    description: 'Phone Number is a required field and must be a 
string'
                },
                address: {
                    bsonType: 'object',
                    required: ['street', 'city', 'postcode'],
                    properties: {
                        street: {
                            bsonType: 'string',
                            description: 'Street must be a string'
                        },
                        city: {
                            bsonType: 'string',
                            description: 'City must be a string'
                        },
                        postcode: {
                            bsonType: 'string',
                            minLength: 1,
                            maxLength: 6,
                            description: 'Postcode must be a string with length between 
1 and 6'
                        }

                    }
                },
                pickup_address: {
                    bsonType: 'object',
                    required: ['street', 'city', 'postcode'],
                    properties: {
                        street: {
                            bsonType: 'string',
                            description: 'Street must be a string'
                        },
                        city: {
                            bsonType: 'string',
                            description: 'City must be a string'
                        },
                        postcode: {
                            bsonType: 'string',
                            minLength: 1,
                            maxLength: 6,
                            description: 'Postcode must be a string with length between
1 and 6'
                        }

                    }
                },
                destination_address: {
                    bsonType: 'object',
                    required: ['street', 'city', 'postcode'],
                    properties: {
                        street: {
                            bsonType: 'string',
                            description: 'Street must be a string'
                        },
                        city: {
                            bsonType: 'string',
                            description: 'City must be a string'
                        },
                        postcode: {
                            bsonType: 'string',
                            minLength: 1,
                            maxLength: 6,
                            description: 'Postcode must be a string with length between
1 and 6'
                        }

                    }
                },
                pickup_time: {
                    bsonType: 'date',
                    description: 'Pick Up time must be a valid ISO datetime'
                },
                email: {
                    bsonType: 'string',
                    minLength: 6,
                    maxLength: 40,
                    pattern:
'[a-z0-9._%+!$&*=^|~#%{}/-]+@([a-z0-9-]+.){1,}([a-z]{2,22})',
                    description: 'It is required and it must be a string with length
between 6 and 40 (regular expression pattern)'
                },
                active: {
                    bsonType: 'bool',
                    description: 'Is Active is a required Boolean field'
                },
                membership_date: {
                    bsonType: 'date',
                    description: 'Membership Date is a required field and must be a
valid ISO date'
                },
                frequency: {
                    bsonType: 'array',
                    description: 'Frequency must be an array and is required',
                },
                last_update:{
                    bsonType: 'timestamp',
                    description: 'last_update must be a timestamp'
                }

            }
        }}})
```

1.4.2 Insert Into Clients Collection:

```
db.Clients.insertOne( {
   "client_type":
   {
       "private": false,
       "corporate": true
   },
   "name":"ACME Ltda.",
   "phone":"07474566665",
   "address":
   {
       "street":"Westminester Rd",
       "city":"Briston",
       "postcode":"W2-3ED"
   },
   "pickup_address": {
       "street": "93 Eric Street",
       "city": "London",
       "postcode": "E4 2ED"
```

```
    },
    "destination_address": {
           "street": "67 Breadboard Lane",
           "city": "London",
           "postcode": "B4 2ED"
    },
   "pickup_time": ISODate("2017-01-03T00:00:00"),
   "email":"logistics@acme.com",
   "active":true,
   "membership_date": ISODate("2017-01-03T00:00:00"),
   "frequency" : [1,2,3,4,5,6,7],

   "last_update": new Timestamp()
});


db.Clients.insertMany([
   {
       "client_type":
         {
           "private": true,
           "corporate": false
         },
       "name": "Albert taylor",
       "phone":"09696523123",
       "address":
         {
           "street":"Groover St",
           "city":"London",
           "postcode":"W3-3ED"
         },
       "pickup_address": {
           "street": "93 Goblet Street",
           "city": "London",
           "postcode": "E4-2ED"
        },
       "destination_address": {
           "street": "67 Sweetplace Lane",
           "city": "London",
           "postcode": "B4-2ED"
        },
       "pickup_time": ISODate("2017-01-06T00:00:00"),
       "email":"albert@gmail.com",
       "active": true,
       "membership_date": ISODate("2017-01-01T00:00:00"),
       "frequency" : [5,6,7],
       "last_update": new Timestamp()
   },
   {
       "client_type":
         {
           "private": true,
           "corporate": false
         },
       "name": "Joseph Button",
       "phone":"09696523123",
       "address":
         {
           "street":"High St",
           "city":"London",
           "postcode":"E5-2ZL"
         },
       "pickup_address": {
           "street": "21/b Baker Street",
           "city": "London",
           "postcode": "E4-2ED"
        },
       "destination_address": {
           "street": "67 Sweetplace Lane",
           "city": "London",
           "postcode": "B4-2ED"
        },
       "pickup_time": ISODate("2017-01-01T00:00:00"),
       "email":"joseph@gmail.com",
       "active": true,
       "membership_date": ISODate("2017-01-01T00:00:00"),
       "frequency" : [5,6,7],
       "last_update": new Timestamp()
   }

]);
```

1.5.1 Create Bookings Collection:

```
db.createCollection('Bookings', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required: ['travel_date', 'customer_info', 'observation', 'paid',
'operator_id', 'driver_id', 'last_update'],
           properties: {
               travel_date: {
                   bsonType: 'date',
                   description: 'Travel Date is a required field and must be a valid
ISO date'
               },
               customer_info:{
                   bsonType: 'object',
                   required: ['name', 'phone_number', 'pickup_address',
'destination_address'],
                   properties: {
                       name: {
                           bsonType: 'object',
                           required: ['first_name', 'last_name'],
                           properties: {
                               first_name: {
                                   bsonType: 'string',
                                   description: 'First Name is required and must be a
string'
                               },
                               last_name: {
                                   bsonType: 'string',
                                   description: 'Last Name is required and must be a
string'
                               }
                           }
                       },
                       phone_number: {
                           bsonType: 'string',
                           description: 'Phone Number is a required field and must be
a string'
                       },
                       pickup_address: {
                           bsonType: 'object',
                           required: ['street', 'city', 'postcode'],
                           properties: {
                               street: {
                                   bsonType: 'string',
                                   description: 'Street must be a string'
                               },
                               city: {
                                   bsonType: 'string',
                                   description: 'City must be a string'
                               },
                               postcode: {
                                   bsonType: 'string',
                                   description: 'Postcode must be a string with length
between 1 and 6'
                               }

                           }
                       },
                       destination_address: {
                           bsonType: 'object',
                           required: ['street', 'city', 'postcode'],
                           properties: {
                               street: {
                                   bsonType: 'string',
                                   description: 'Street must be a string'
                               },
                               city: {
                                   bsonType: 'string',
                                   description: 'City must be a string'
                               },
                               postcode: {
                                   bsonType: 'string',
                                   description: 'Postcode must be a string with length
between 1 and 6'
                               }

                           }
                       }

                   }
               },
               observation: {
                   bsonType: 'string',
                   description: 'Observation is a required field and must be a string'
               },
               paid: {
                   bsonType: 'bool',
                   description: 'Paid is a required and a boolean field'
               },
               operator_id: {
                   bsonType: 'objectId',
                   description: 'Operator Id is a required field and must be of type 
objectid'
               },
               driver_id: {
                   bsonType: 'objectId',
                   description: 'Driver Id is a required field and must be of type 
objectid'
               },
               client_id: {
                   bsonType: 'objectId',
                   description: 'Client Id is a required field and must be of type 
objectid'
               },
               last_update:{
                   bsonType: 'timestamp',
                   description: 'last_update must be a timestamp'
               }
           }
       }},
       validationAction: 'warn'
   })
```

1.5.2 Insert Data Into Bookings Collection:

```
operator_charde = db.Operator.findOne({
   name: {
           first_name: "Charde",
           last_name: "Marshall"
         }
});
```

```
driver_bradley = db.Drivers.findOne({
   name: {
             first_name:"Bradley",
             last_name:"Greer"
         }
});

driver_hermione = db.Drivers.findOne({
   name: {
             first_name:"Hermione",
             last_name:"Granger"
         }
});

driver_james = db.Drivers.findOne({
   name: {
             first_name:"JAMES",
             last_name:"Cameroon"
         }
});

driver_ronald = db.Drivers.findOne({
   name: {
             first_name:"Ronald",
             last_name:"Weasley"
         }
});

client_ACME = db.Clients.findOne({
   name: "ACME Ltda." });

db.Bookings.insertOne(
   {
       "travel_date": ISODate("2022-03-04T12:37:00"),
       "customer_info": {
           "name":
           {
             "first_name":"Cara",
             "last_name":"Stevens"
           },
           "phone_number": "04747955265",
           "pickup_address": {
               "street": "12 Whitechaple Rd",
               "city": "London",
               "postcode": "E4 2ED"
           },
           "destination_address": {
               "street": "67, Migglets Lane",
               "city": "London",
               "postcode": "G4 2ED"
           }
       },
       "observation": "Large Lugagges",
       "paid": true,
       "last_update": new Timestamp(),
       "operator_id": operator_charde._id ,
       "driver_id": driver_bradley._id,
       "client_id": client_ACME._id
   });


db.Bookings.insertOne(
       {
           "travel_date": ISODate("2022-11-09T11:00:00"),
           "customer_info": {
               "name":
               {
                 "first_name":"Martin",
                 "last_name":"Cavazos"
               },
               "phone_number": "04747222211",
               "pickup_address": {
                   "street": "25 Shoreditch Rd",
                   "city": "London",
                   "postcode": "E4 6ED"
               },
               "destination_address": {
                   "street": "14 Whitechappel Rd",
                   "city": "London",
                   "postcode": "E4 2ED"
               }
           },
           "observation": "Wheelchair",
           "paid": true,
           "last_update": new Timestamp(),
           "operator_id": operator_charde._id ,
           "driver_id": driver_hermione._id,
           "client_id": client_ACME._id

       });


db.Bookings.insertOne(
           {
               "travel_date": ISODate("2022-05-05T12:10:00"),
               "customer_info": {
                   "name":
                   {
                     "first_name":"Tom",
                     "last_name":"Riddle"
                   },
                   "phone_number": "04747956765",
                   "address": {
                       "street": "Wilcox Rd",
                       "city": "London",
                       "postcode": "E4 2ED"
                   },
                   "pickup_address": {
                       "street": "12 Sheren Rd",
                       "city": "London",
                       "postcode": "S4 2ED"
                    },
                   "destination_address": {
                       "street": "67, Queen Lane",
                       "city": "London",
                       "postcode": "Q4 2ED"
                    }
               },
               "observation": "",
               "paid": true,
               "last_update": new Timestamp(),
               "operator_id": operator_charde._id ,
               "driver_id": driver_hermione._id,
               "client_id": client_ACME._id
           });
```

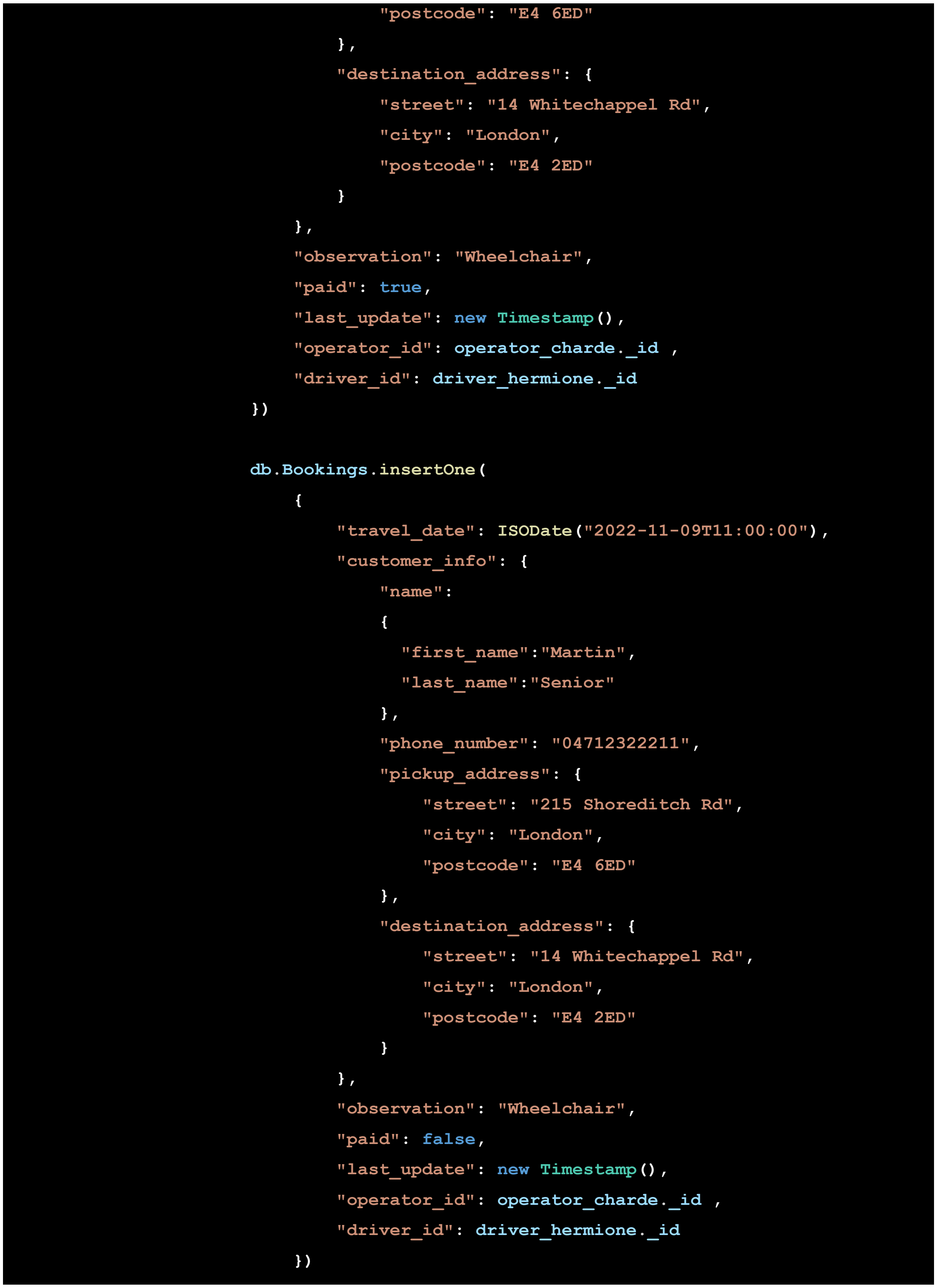
```
db.Bookings.insertOne(
    {
        "travel_date": ISODate("2022-05-05T12:10:00"),
        "customer_info": {
            "name":
            {
              "first_name":"Tom",
              "last_name":"Riddle"
            },
            "phone_number": "04747956765",
            "address": {
                "street": "Wilcox Rd",
                "city": "London",
                "postcode": "E4 2ED"
            },
            "pickup_address": {
                "street": "12 Sheren Rd",
                "city": "London",
                "postcode": "S4 2ED"
             },
            "destination_address": {
                "street": "67, Queen Lane",
                "city": "London",
                "postcode": "Q4 2ED"
             }
        },
        "observation": "",
        "paid": true,
        "last_update": new Timestamp(),
        "operator_id": operator_charde._id ,
        "driver_id": driver_bradley._id,
        "client_id": client_albert._id
    });


    db.Bookings.insertOne(
        {
            "travel_date": ISODate("2022-05-05T12:10:00"),
            "customer_info": {
                "name":
                {
```

```
          "first_name":"Tom",
          "last_name":"Riddle"
        },
        "phone_number": "04747956765",
        "address": {
            "street": "Wilcox Rd",
            "city": "London",
            "postcode": "E4 2ED"
        },
        "pickup_address": {
            "street": "12 Sheren Rd",
            "city": "London",
            "postcode": "S4 2ED"
         },
        "destination_address": {
            "street": "67, Queen Lane",
            "city": "London",
            "postcode": "Q4 2ED"
         }
    },
    "observation": "",
    "paid": true,
    "last_update": new Timestamp(),
    "operator_id": operator_charde._id ,
    "driver_id": driver_hermione._id
})



db.Bookings.insertOne(
    {
        "travel_date": ISODate("2022-11-09T11:00:00"),
        "customer_info": {
            "name":
            {
              "first_name":"Martin",
              "last_name":"Cavazos"
            },
            "phone_number": "04747222211",
            "pickup_address": {
                "street": "25 Shoreditch Rd",
                "city": "London",
```



```
            "postcode": "E4 6ED"
        },
        "destination_address": {
            "street": "14 Whitechappel Rd",
            "city": "London",
            "postcode": "E4 2ED"
        }
    },
    "observation": "Wheelchair",
    "paid": true,
    "last_update": new Timestamp(),
    "operator_id": operator_charde._id ,
    "driver_id": driver_hermione._id
})

db.Bookings.insertOne(
    {
        "travel_date": ISODate("2022-11-09T11:00:00"),
        "customer_info": {
            "name":
            {
              "first_name":"Martin",
              "last_name":"Senior"
            },
            "phone_number": "04712322211",
            "pickup_address": {
                "street": "215 Shoreditch Rd",
                "city": "London",
                "postcode": "E4 6ED"
            },
            "destination_address": {
                "street": "14 Whitechappel Rd",
                "city": "London",
                "postcode": "E4 2ED"
            }
        },
        "observation": "Wheelchair",
        "paid": false,
        "last_update": new Timestamp(),
        "operator_id": operator_charde._id ,
        "driver_id": driver_hermione._id
    })
```

```
db.Bookings.insertOne(
    {
        "travel_date": ISODate("2022-05-05T12:10:00"),
        "customer_info": {
            "name":
            {
              "first_name":"Tom",
              "last_name":"Moody"
            },
            "phone_number": "0472956765",
            "address": {
                "street": "Wilcox Rd",
                "city": "London",
                "postcode": "E4 2ED"
            },
            "pickup_address": {
                "street": "12 Brooklyn Rd",
                "city": "London",
                "postcode": "S4 1ED"
             },
            "destination_address": {
                "street": "67, Kings Lane",
                "city": "London",
                "postcode": "Q4 4ED"
             }
        },
        "observation": "",
        "paid": false,
        "last_update": new Timestamp(),
        "operator_id": operator_charde._id ,
        "driver_id": driver_bradley._id
    });


    db.Bookings.insertOne(
        {
            "travel_date": ISODate("2022-05-05T12:10:00"),
            "customer_info": {
                "name":
                {
                  "first_name":"Tom",
                  "last_name":"Riddle"
                },
                "phone_number": "04747956765",
                "address": {
                    "street": "Wilcox Rd",
                    "city": "London",
                    "postcode": "E4 2ED"
                },
                "pickup_address": {
                    "street": "12 Sheren Rd",
                    "city": "London",
                    "postcode": "S4 2ED"
                 },
                "destination_address": {
                    "street": "67, Queen Lane",
                    "city": "London",
                    "postcode": "Q4 2ED"
                 }
            },
            "observation": "",
            "paid": true,
            "last_update": new Timestamp(),
            "operator_id": operator_charde._id ,
            "driver_id": driver_ronald._id,
            "client_id": client_albert._id
        });

        db.Bookings.insertOne(
            {
                "travel_date":
        ISODate("2022-06-05T12:10:00"),
                "customer_info": {
                    "name":
                    {
                      "first_name":"Tom",
                      "last_name":"Riddle"
                    },
                    "phone_number": "04747956765",
                    "address": {
                        "street": "Wilcox Rd",
                        "city": "London",
                        "postcode": "E4 2ED"
```

```
                                                },
                                                "pickup_address": {
                                                    "street": "12 Sheren Rd",
                                                    "city": "London",
                                                    "postcode": "S4 2ED"
                                                 },
                                                "destination_address": {
                                                    "street": "67, Queen Lane",
                                                    "city": "London",
                                                    "postcode": "Q4 2ED"
                                                 }
                                            },
                                            "observation": "",
                                            "paid": false,
                                            "last_update": new Timestamp(),
                                            "operator_id": operator_charde._id ,
                                            "driver_id": driver_ronald._id,
                                            "client_id": client_ACME._id
                                        });


                                        db.Bookings.insertOne(
                                            {
                                                "travel_date":
ISODate("2022-05-15T12:10:00"),
                                                "customer_info": {
                                                    "name":
                                                    {
                                                      "first_name":"Tom",
                                                      "last_name":"Moody"
                                                    },
                                                    "phone_number": "0472956765",
                                                    "address": {
                                                        "street": "Wilcox Rd",
                                                        "city": "London",
                                                        "postcode": "E4 2ED"
                                                    },
                                                    "pickup_address": {
                                                        "street": "12 Brooklyn Rd",
                                                        "city": "London",
                                                        "postcode": "S4 1ED"
                                                     },
                                                    "destination_address": {
                                                        "street": "67, Kings Lane",
                                                        "city": "London",
                                                        "postcode": "Q4 4ED"
                                                     }
                                                },
                                                "observation": "",
                                                "paid": false,
                                                "last_update": new Timestamp(),
                                                "operator_id": operator_charde._id ,
                                                "driver_id": driver_ronald._id
                                            });


                db.Bookings.insertOne(
                    {
                        "travel_date": ISODate("2022-05-11T12:10:00"),
                        "customer_info": {
                            "name":
                            {
                              "first_name":"Tom",
                              "last_name":"Riddle"
                            },
                            "phone_number": "04747956765",
                            "address": {
                                "street": "Wilcox Rd",
                                "city": "London",
                                "postcode": "E4 2ED"
                            },
                            "pickup_address": {
                                "street": "12 Sheren Rd",
                                "city": "London",
                                "postcode": "S4 2ED"
                             },
                            "destination_address": {
                                "street": "67, Queen Lane",
                                "city": "London",
                                "postcode": "Q4 2ED"
                             }
                        },
                        "observation": "",
                        "paid": false,
                        "last_update": new Timestamp(),
                        "operator_id": operator_charde._id ,
                        "driver_id": driver_james._id
                    })
```

1.6.1 Create Payments Collection:

```
db.createCollection('Payments', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required: [ 'method', 'amount', 'payment_date', 'booking_id',
'last_update'],
           properties: {
               method: {
                   bsonType: 'object',
                   required: ['cash', 'credit_card', 'debit_card'],
                   properties: {
                       cash: {
                           bsonType: 'bool',
                           description: 'Cash is a required Boolean field'
                       },
                       credit_card: {
                           bsonType: 'bool',
                           description: 'Credit Card is a required Boolean field'
                       },
                       debit_card: {
                           bsonType: 'bool',
                           description: 'Debit Card is a required Boolean field'
                       }
                   }
               },
               amount: {
                   bsonType: 'number',
                   description: 'Payment Amount is a required field and must be a
number'
               },
               payment_date: {
                   bsonType: 'date',
                   description: 'Payment Date is a required field and must be a valid
date'
```

```
                },
                booking_id:{
                    bsonType: 'objectId',
                    description: 'Booking Id is a required field and must be of type
objectid'
                },
                last_update:{
                    bsonType: 'timestamp',
                    description: 'last_update must be a timestamp'
                },


            }
        }},
        validationAction: 'warn'
    })
```

1.6.2 Insert Data Into Payments Collection:

```
bookingIds = db.Bookings.find({},{paid: 1}).toArray();





   db.Payments.insertMany([
        {
            "method":
                {
                    "cash":true,
                    "credit_card":false,
                    "debit_card":false
                },
            "amount": 210,
            "payment_date": ISODate("2022-01-01T00:00:00"),
            "last_update": new Timestamp(),
            "booking_id": bookingIds[0]._id
        },
        {
            "method":
                {
```

```
                "cash":true,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 110,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id":  bookingIds[1]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":true,
                "debit_card":false
            },
        "amount": 210,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[2]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":true,
                "debit_card":false
            },
        "amount": 2110,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[3]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":true,
                "debit_card":false
            },
        "amount": 410,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[4]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":true,
                "debit_card":false
            },
        "amount": 110,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[5]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 1410,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[6]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 2910,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[7]._id
    },
    {
        "method":
            {
                "cash":true,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 2910,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[8]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 10,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[9]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 40,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[10]._id
    },
    {
        "method":
            {
                "cash":false,
                "credit_card":false,
                "debit_card":false
            },
        "amount": 20,
        "payment_date": ISODate("2022-01-01T00:00:00"),
        "last_update": new Timestamp(),
        "booking_id": bookingIds[11]._id
    },

]);
```

1.7.1 Create Revenue Collection:

```
db.createCollection('Revenue', {
   validator:{
       $jsonSchema: {
           bsonType: 'object',
           required: ['driver_id', 'driver_wage', 'company_wage'],
           properties: {
               driver_id: {
                   bsonType: 'objectId',
                   description: 'Driver Id is a required field and must be of type
objectid'
               },
               driver_wage: {
                   bsonType: 'number',
                   description: 'driver_wage is a required field and must be a number'
               },
               company_wage: {
                   bsonType: 'number',
                   description: 'company_wage is a required field and must be a
number'
               }
           }
       }}

   });
```

1.7.2 Insert Data Into Revenue Collection:

```

driver_ronlad = db.Drivers.findOne({
   "name.first_name": "Ronald",
   "name.last_name": "Weasley"
});
driver_hermoine = db.Drivers.findOne({
   "name.first_name": "Hermione",
   "name.last_name": "Granger"
})
driver_bradely= db.Drivers.findOne({
   "name.first_name": "Bradley",
   "name.last_name": "Greer"
});


db.Revenue.insertMany([
   {
       "driver_id": driver_ronlad._id,
       "driver_wage": 1314,
       "company_wage": 876
   },
   {
       "driver_id": driver_hermoine._id,
       "driver_wage": 504,
       "company_wage": 336
   },
   {
       "driver_id": driver_bradely._id,
       "driver_wage": 1392,
       "company_wage": 928
   },

])
```