Design and Analysis of Transportation Management Database Using MongoDB: A Case Study

Ikmal Norhakim Norhaziman, Norhusnina Zulaikha Muhammad Rizwan, Nurul Athirah Abdul Rahman, Siti Fatehah Mahamad, Toh Min Xuan

Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia Pagoh Campus, KM 1, Jalan Panchor, 84600 Panchor, Johor, Malaysia.

KEYWORDS

MongoDB

NoSQL Database

Transport Management System (TMS)

Data Modelling

Data Analytics

Aggregation Pipeline

ABSTRACT

This study aims to design and analyze a transportation management database system using MongoDB, a NoSQL database. The project began with rigorous cleaning using Python and preprocess the synthetic data generated using Mockaroo to ensure data quality. During the data modeling phase, we built a multi-collection architecture that included core entities such as drivers, vehicles, routes, trip logs, bookings and maintenance and strategically used references to optimize data consistency and query flexibility. In order to improve system performance, key query fields were efficiently indexed. By using MongoDB's aggregation pipeline function, key operational indicators including route utilization, maintenance costs, driver performance and booking trends are analyzed deepedly. All analysis results were visualized through Python to provide intuitive insights. Finally, this report puts forward several practical suggestions based on data analysis, such as implementing predictive maintenance, optimizing route planning and improving driver scheduling strategies. This aims to help transportation organization to improve operational efficiency, reduce costs and optimize overall service quality.

Corresponding Author:

Dr. Logenthiran A/L Machap
Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology
Universiti Tun Hussein Onn Malaysia Pagoh Campus, KM 1, Jalan Panchor, 84600 Panchor, Johor, Malaysia.
Email: logen@uthm.edu.my

1.0 INTRODUCTION

In recent years, the growth of transportation systems has led to a significant increase in data generation, especially from daily operations such as trip scheduling, passenger bookings, vehicle maintenance and route management. To manage and analyze such complex and dynamic datasets, the use of non-relational databases has become increasingly important due to their flexibility and scalability.

This project focuses on the development of a NoSQL-based data analytics system for a fictional transportation company. The system is built using MongoDB to simulate real-world operations such as managing fleets, recording passenger data, tracking trip logs and analyzing maintenance activities.

By designing a well-structured MongoDB schema and performing advanced data queries, the objective is to gain actionable insights that can help the company make better decisions — such as optimizing routes, identifying peak travel times and reducing vehicle downtime.

This project also provides an opportunity to apply core NoSQL concepts such as document-based modeling, schema flexibility and aggregation pipelines to a real-world scenario. The data used for this project is synthetically generated using Mockaroo and spreadsheet editing tools, ensuring variety and realism while maintaining data integrity.

2.0 RELATED WORK

In complex data environments such as transportation management systems, MongoDB has been widely used in industries such as e-commerce and streaming. In the e-commerce field, large platforms such as Amazon and Flipkart use MongoDB to manage product catalogs, user activity logs and order information to cope with diverse data structures, frequent data updates, high concurrency read and write requirements (Rana et al., 2020; Singh & Sharma, 2022). MongoDB's flexible document model has also been adopted by companies such as Shopify and Zalando for real-time inventory management and user behavior analysis (Kapoor & Gupta, 2021; Kumar & Ghosh, 2023).

In the field of streaming media, YouTube uses MongoDB to manage user comments and channel metadata to improve data read and write efficiency and structural flexibility; Hulu also uses a similar NoSQL architecture to process unstructured user preference data in personalized recommendation services (Mehta & Shah, 2021; Natarajan et al., 2024). These application cases show that MongoDB is particularly suitable for systems that require high scalability, flexible data structures and real-time processing capabilities (Alflahi et al., 2023).

In contrast, traditional relational databases (such as MySQL and PostgreSQL) emphasize transaction consistency and the rigor of relational modeling and are suitable for business scenarios with stable structures and clear relationships, such as banking and financial systems (Ali et al., 2020; Chidambaram & Prakash, 2021). However, in transportation management systems, trip logs, booking information, vehicle maintenance records and driver data are semi-structured and the structural relationships change frequently (Ramesh & Kumar, 2021). If this type of data is modelled using MongoDB's document model, it will be more flexible, significantly simplify query logic and improve development efficiency (Bisht & Joshi, 2023). In addition, MongoDB natively supports horizontal sharding, multi-document transactions and aggregate analysis, has stronger data expansion capabilities and processing concurrency performance (Jain & Saxena, 2022; Alam & Rahman, 2020).

In summary, MongoDB has significant advantages over traditional relational databases in terms of structural flexibility, query efficiency and system scalability and is particularly suitable for transportation management systems for real-time analysis and big data processing (Patel et al., 2021; Gupta & Mahajan, 2024).

3.0 METHODS

3.1 DATABASE DESIGN

In terms of database modeling, we designed six core collections based on the business needs of the transportation management system, namely: drivers, vehicles, routes, trip_logs, bookings and maintenance. To avoid data redundancy and improve query efficiency, the system mainly adopts reference modeling, using driver_id, vehicle_id and route_id as foreign keys to establish data associations in Trip_logs. This design method facilitates cross-collection analysis, such as querying detailed information about drivers and vehicles involved in a trip.

For fields with relatively stable data structures and few changes (such as the origin, destination and distance of a route), we use document embedding to simplify the structure and improve query performance. In terms of performance optimization, we set up single-field indexes and composite indexes for commonly used query fields. For example, we create a composite index for {origin, destination} in the Routes collection and create indexes for vehicle_id, driver_id, etc. in Trip_logs to optimize the speed of filtering and aggregation operations.

3.2 TOOLS AND PLATFORM

In terms of data generation, Mockaroo platform is used to create structured and realistically simulated data samples. Each set generates about 100 records.

For data preprocessing, Python was used to unify the date and time formats, clean invalid values (such as unreasonable time fields) and convert the data into json format for import into the MongoDB database. In terms of database operations, we used the Compass graphical interface provided by MongoDB for collection management and index settings and complex queries were implemented through PyMongo.

In terms of data visualization, Python (Matplotlib, Seaborn) is used to generate a variety of charts and dashboards, such as route usage frequency, driver performance and maintenance cost statistics to help improve data interpretability and insight.

3.3 IMPLEMENTATION

3.3.1 CRUD OPERATION WITH MONGODB

The system incorporates the full range of CRUD (Create, Read, Update, Delete) operations to manage transportation data efficiently across multiple collections such as drivers, vehicles, routes, trip_logs, bookings and maintenance. Before each query is proceed, these operations are implemented to ensure to ensure data integrity, minimize redundancy, and uphold consistency across interconnected datasets.

These operations are governed by internal checks and constraints to ensure that business logic remains intact, particularly in cases where multiple collections interact. For example, updating a vehicle's availability status must reflect accurately in both vehicles and bookings collections.

i. Create

Adding new data, like inserting a new drivers record.

ii. Read

View or search data, such as checking driver info.

iii. Update

Edit existing data, for example, updating driver's experience years.

iv. Delete

Remove unwanted data, like deleting driver_id: 5 and deleting drivers with 5 years experience.

3.3.2 HANDLING REAL-TIME AND DYNAMIC REQUIREMENTS

Although MongoDB does not inherently support real-time updates, this project implemented several techniques to simulate dynamic responsiveness and ensure efficient querying of large datasets.

First, index optimization was applied by creating indexes on frequently queried fields such as driver_id, route_id and status. These indexes significantly improved the speed of filtering and aggregation operations, which are essential for analytical queries within the transportation management system.

Second, aggregation pipelines were used to support advanced and dynamic queries. Operators such as `$group`, `$match`, `$sort` and `$lookup` were employed to calculate insights like average trip durations, identify the busiest routes and determine vehicles with the highest maintenance frequency. These pipelines allowed for meaningful analysis across multiple collections without redundancy.

Furthermore, conditional handling using MongoDB operators like `$cond` and `$dateFromString` ensured that mixed data types, especially within time and date fields, were correctly processed during aggregations. This enabled more accurate calculations, such as measuring average durations or filtering based on specific time ranges.

Lastly, composite indexing strategies were used to further improve query efficiency. For instance, a compound index on {origin: 1, destination: 1} within the Routes collection facilitated complex queries involving both origin and destination filters. This proved essential in supporting queries aimed at analyzing travel patterns and optimizing route allocation.

4.0 RESULTS AND DISCUSSION

4.1 QUERY PERFORMANCE OBSERVATIONS

Query 1: Show how many routes have travel time above 7 hours

```
> db.Routes.aggregate([
    {
      $match: {
        estimated_time: { $gt: 7 }
      }
    },
    {
      $project: {
        route_id: 1,
        origin: 1,
        destination: 1,
        distance_km: 1,
        estimated_time: 1
      }
    }
  ]);
```

This query is to identify how many routes require more than 7 hours to travel. This is useful for detecting long journeys or for planning high duration logistics.

```
< {
    _id: ObjectId('685257337d757d7d87175e3a'),
    route_id: 18,
    origin: 'Zhaxi',
    destination: 'Binitayan',
    distance_km: 258,
    estimated_time: 8
  }
  {
    _id: ObjectId('685257337d757d7d87175e3d'),
    route_id: 21,
    origin: 'Juan de Acosta',
    destination: 'Gremyachinsk',
    distance_km: 472,
    estimated_time: 8
  }
  {
    _id: ObjectId('685257337d757d7d87175e41'),
    route_id: 25,
    origin: 'Pingshan',
    destination: 'Krapina',
    distance_km: 176,
```

Sample Output for Query 1: The Routes with Travel Time Above 7 Hours.

Query 2: Count of Routes in Distance Ranges

```
> db.Routes.aggregate([
    {
      $bucket: {
        groupBy: "$distance_km",
        boundaries: [0, 100, 200, 300, 400, 500, 600],
        default: "600+ km",
        output: {
          count: { $sum: 1 },
          avg_estimated_time: { $avg: "$estimated_time" }
        }
      }
    }
  ])
```

This purpose is to categorize routes into distance buckets (example: 0–99 km, 100–199 km, etc.) and show the number of routes. This helps in analyzing route distribution and understanding how distance affects trip planning and travel time.

```
< {
    _id: 0,
    count: 10,
    avg_estimated_time: 3.9
  }
  {
    _id: 100,
    count: 24,
    avg_estimated_time: 4.260869565217392
  }
  {
    _id: 200,
    count: 17,
    avg_estimated_time: 5.117647058823529
  }
  {
    _id: 300,
    count: 25,
    avg_estimated_time: 4.32
  }
  {
    _id: 400,
    count: 25,
    avg_estimated_time: 5
  }
```

Sample Output for Query 2: Number of Routes in Distance Ranges.

Query 3: Find Top 5 Longest Routes by Distance

```
> db.Routes.aggregate([
    {
      $sort: { distance_km: -1 }
    },
    {
      $limit: 5
    },
    {
      $project: {
        _id: 0,
        origin: 1,
        destination: 1,
        distance_km: 1,
        estimated_time: 1
      }
    }
  ])
```

Here is to identify the 5 longest routes based on distance. This is helpful for spotting extreme cases in the data which possibly for route optimization or logistics management.

```
< {
    origin: 'Rowokangkung',
    destination: 'Dobra',
    distance_km: 498,
    estimated_time: 5
  }
  {
    origin: 'Połomia',
    destination: 'Kusatsu',
    distance_km: 491,
    estimated_time: 4
  }
  {
    origin: 'Guadalupe',
    destination: 'Kiuruvesi',
    distance_km: 490,
    estimated_time: 4
  }
  {
    origin: 'Qingliu',
    destination: 'Taldyqorghan',
    distance_km: 488,
    estimated_time: 6
  }
  {
    origin: 'Brodek u Přerova',
    destination: 'Monte de Fralães',
    distance_km: 487,
```

Sample Output for Query 3: The Top 5 Longest Routes by Distance.

Query 4: The Total Cost and How Many Times It Has Been Maintained Based on Maintenance Type

This query analyzes maintenance costs by grouping records based on maintenance type, calculating the total cost and frequency of each type and sorting them from highest to lowest cost.

```
> db.maintenance.aggregate([
    {
      $group: {
        _id: "$type",
        totalCost: { $sum: "$cost" },
        typeCount: { $sum: 1 }
      }
    },
    { $sort: { totalCost: -1 } }
  ])
```

The output will shows the total cost for type of maintenance and the frequency that it had be maintenance.

```
>_MONGOSH
    _id: 'Brake Check',
    totalCost: 60484,
    count: 225
  }
  {
    _id: 'Air Filter',
    totalCost: 57105,
    count: 193
  }
  {
    _id: 'Tire Rotation',
    totalCost: 54665,
    count: 198
  }
  {
    _id: 'Oil Change',
    totalCost: 53004,
    count: 196
  }
  {
    _id: 'Battery',
    totalCost: 51197,
    count: 189
```

Sample Output for Query 4: Total Cost and Maintenance Frequency based on Each Maintenance Type.

Query 5: Total Cost and Maintenance Count Based on Type of Vehicle.

This query joins maintenance data with vehicle information to determine which vehicle types incur the highest maintenance costs, again sorted by total cost.

```
> db.maintenance.aggregate([
    {
      $lookup: {
        from: "vehicles",
        localField: "vehicle_id",
        foreignField: "vehicle_id",
        as: "vehicle_info"
      }
    },
    { $unwind: "$vehicle_info" },
    {
      $group: {
        _id: "$vehicle_info.type",
        totalCost: { $sum: "$cost" },
        maintenanceCount: { $sum: 1 }
      }
    },
    { $sort: { totalCost: -1 } }
  ])
```

The output will show the total cost maintenance and frequency for type vehicles.

```
[
  {
    _id: 'Van',
    totalCost: 119490,
    maintenanceCount: 423
  }
  {
    _id: 'Minibus',
    totalCost: 98499,
    maintenanceCount: 365
  }
  {
    _id: 'Bus',
    totalCost: 58466,
    maintenanceCount: 213
  }
```

Sample Output for Query 5: Total Cost and Maintenance Count based on Type of Vehicles.

Query 6: Drivers and Maintenance Records

This query links maintenance records to drivers, showing the total number of tasks, total and average costs and driver experience, helping to identify which drivers are associated with the highest maintenance expenses.

```
db.maintenance.aggregate([
  {
    $lookup: {
      from: "drivers",
      localField: "vehicle_id",
      foreignField: "assigned_vehicle_id",
      as: "driver_info"
    }
  },
  {
    $unwind: "$driver_info"
  },
  {
    $group: {
      _id: {
        driver_name: "$driver_info.name",
        vehicle_id: "$vehicle_id"
      },
      total_maintenance_tasks: { $sum: 1 },
      total_cost: { $sum: "$cost" },
      avg_cost: { $avg: "$cost" },
      experience_years: { $first: "$driver_info.experience_years" } }
  },
  {
    $project: {
      _id: 0,
      driver_name: "$_id.driver_name",
      vehicle_id: "$_id.vehicle_id",
      total_maintenance_tasks: 1,
      total_cost: 1,
      experience_years: 1,
      avg_cost: { $round: ["$avg_cost", 2] }
    }
  },
  {
    $sort: { total_cost: -1 }}])
```

The output will show total maintenances, total cost for all maintenances and average cost per maintenance for each driver and included drivers details.

```
>_MONGOSH
    total_maintenance_tasks: 16,
    total_cost: 4847,
    experience_years: 11,
    driver_name: 'Brennen Van der Kruys',
    vehicle_id: 97,
    avg_cost: 302.94
  }
  {
    total_maintenance_tasks: 15,
    total_cost: 4691,
    experience_years: 19,
    driver_name: 'Phelia Gooda',
    vehicle_id: 36,
    avg_cost: 312.73
  }
  {
    total_maintenance_tasks: 15,
    total_cost: 4691,
    experience_years: 3,
    driver_name: 'Waldemar Staines',
    vehicle_id: 36,
    avg_cost: 312.73
  }
```

Sample Output for Query 6: Drivers and Maintenance Records.

Query 7: Managing Vehicle Maintenance and Status (entities: Vehicle)

The purpose of this query is to identify which vehicle types contribute most to the total active fleet capacity, supporting effective operational planning.

```
[
  {
    $match: { status: "Active" } // Only include active vehicles
  },
  {
    $group: {
      _id: "$type", // Group by vehicle type
      totalCapacity: { $sum: "$capacity" },
      vehicleCount: { $sum: 1 }
    }
  },
  {
    $sort: { totalCapacity: -1 } // Sort by total capacity descending
  },
  {
    $project: {
      _id: 0,
      vehicleType: "$_id",
      totalCapacity: 1,
      vehicleCount: 1
    }
  }
]
```

```
totalCapacity : 349
vehicleCount : 11
vehicleType : "Minibus"

totalCapacity : 333
vehicleCount : 11
vehicleType : "Van"

totalCapacity : 169
vehicleCount : 6
vehicleType : "Bus"
```

Coding and Sample Pipeline Output for Query 7: Find top vehicle types by total fleet capacity using aggregate. (Only Active Vehicles)

Query 8: Tracking Delayed Vehicle Types

```
> db.vehicles.createIndex({ status: 1, type: 1 })
< status_1_type_1
> db.trip_logs.createIndex({ status: 1, vehicle_id: 1 });

< status_1_vehicle_id_1
> db.trip_logs.createIndex({ route_id: 1 });
< route_id_1
```

```
> route_id_1          REGULAR  20.5 KB   0 (since Mon Jun 23 2025)   READY
> status_1_type_1     REGULAR  20.5 KB   0 (since Mon Jun 23 2025)   COMPOUND
> status_1_vehicle_id_1  REGULAR  20.5 KB   9 (since Mon Jun 23 2025)   COMPOUND
```

Coding and output in Query 8: Create compund indexes for performance query.

To enhance query efficiency in transport analytics, indexes are applied to key fields. In the **vehicles** collection, {status: 1, type: 1} speeds up maintenance-related queries. In **trip_logs**, {status: 1, vehicle_id: 1} improves tracking of trip outcomes, while { route_id: 1} enables fast route-based analysis. These indexes reduce query time and support efficient data processing.

The aim is to identify delay-prone vehicle types to help optimize allocation, maintenance and scheduling, improving overall service reliability.

```
[
  // Step 1: Filter only delayed trips
  { $match: { status: "Delayed" } },

  // Step 2: Join vehicle details from the 'vehicles' collection using vehicle_id
  {
    $lookup: {
      from: "vehicles",           // target collection to join
      localField: "vehicle_id",   // field in trip_logs
      foreignField: "vehicle_id", // field in vehicles
      as: "vehicle_info"          // output array field
    }
  },

  // Step 3: Flatten the joined array (assumes one-to-one match)
  { $unwind: "$vehicle_info" },

  // Step 4: Group by vehicle type (from joined vehicle info)
  {
    $group: {
      _id: "$vehicle_info.type",   // group by vehicle type
      total_delays: { $sum: 1 }    // count the number of delayed trips per type
    }
  },

  // Step 5: Sort types by number of delays descending (most problematic on top)
  { $sort: { total_delays: -1 } }
]
```

PIPELINE OUTPUT

Sample of 3 documents

```
total_delays : 17
type : "Van"

total_delays : 11
type : "Minibus"

total_delays : 4
type : "Bus"
```

Coding and Sample Pipeline Output for Query 8: Find the most delayed vehicle types.

Query 9: Driver Insights for Transport Systems (entities: Drivers)

The purpose is to highlight variation in driver experience within each vehicle type, as a higher standard deviation indicates inconsistent skill levels, helping to inform balanced training.

```
[
  {
    $lookup: {
      from: "vehicles",
      localField: "assigned_vehicle_id",
      foreignField: "vehicle_id",
      as: "vehicle"
    }
  },
  { $unwind: "$vehicle" },
  {
    $group: {
      _id: "$vehicle.type",
      avgExp: { $avg: "$experience_years" },
      drivers: { $push: "$experience_years" }
    }
  },
  {
    $project: {
      _id: 0,
      vehicle_type: "$_id",
      avgExp: 1,
      stdDevExp: {
        $sqrt: {
          $avg: {
            $map: {
              input: "$drivers",
              as: "exp",
              in: { $pow: [{ $subtract: ["$$exp", "$avgExp"] }, 2] }
            }
          }
        }
      }
    }
  }
]
```

```
{
  avgExp: null,
  vehicle_type: 'Truck',
  stdDevExp: null
}
{
  avgExp: 14.447368421052632,
  vehicle_type: 'Minibus',
  stdDevExp: 8.780053256105877
}
{
  avgExp: 15.653846153846153,
  vehicle_type: 'Bus',
  stdDevExp: 7.893241060418073
}
{
  avgExp: 13.666666666666666,
  vehicle_type: 'Van',
  stdDevExp: 8.216098435648973
}
```

Coding and Sample Pipeline Output for Query 9: Calculate Standard Deviation of Driver Experience for Each Vehicle Type.

Based on this output, it shows the minibus category shows a notably high standard deviation of 8.78 years, highlighting significant variation in driver experience. This suggests the need for targeted training programs, mentorship strategies, or experience-based assignments to ensure a more balanced and capable driver pool. By identifying vehicle types with the greatest experience gaps, organizations can take proactive steps to improve safety, efficiency and service quality.

Query 10: Top 5 Drivers by Completed Trips

This query identifies the drivers with the most completed trips and adds their details from the drivers collection. It helps highlight high-performing staff and can be used for recognition or performance management.

Coding and Sample Pipeline Output for Query 10: Top 5 Drivers by Completed Trips.

Query 11: Average Trip Duration by Route

This query calculates the average trip duration for each route. The result can be compared against estimated durations to assess whether actual trips are running on time or often delayed.

```
project> db.trip_logs.aggregate([
  {
    $project: {
      route_id: 1,
      departure: {
        $cond: [
          { $eq: [{ $type: "$departure_time" }, "string"] },
          { $dateFromString: { dateString: "$departure_time" } },
          "$departure_time"
        ]
      },
      arrival: {
        $cond: [
          { $eq: [{ $type: "$arrival_time" }, "string"] },
          { $dateFromString: { dateString: "$arrival_time" } },
          "$arrival_time"
        ]
      }
    }
  },
  {
    $project: {
      route_id: 1,
      duration_minutes: {
        $divide: [
          { $subtract: ["$arrival", "$departure"] },
          1000 * 60
        ]
      }
    }
  },
  {
    $group: {
      _id: "$route_id",
      average_duration: { $avg: "$duration_minutes" }
    }
  },
  {
    $lookup: {
      from: "routes",
      localField: "_id",
      foreignField: "route_id",
      as: "route_info"
    }
  },
  { $unwind: "$route_info" },
  {
    $project: {
      route_id: "$_id",
      origin: "$route_info.origin",
      destination: "$route_info.destination",
      distance_km: "$route_info.distance_km",
      estimated_time_hr: "$route_info.estimated_time",
      average_duration: 1,
      _id: 0
    }
  },
  { $sort: { average_duration: -1 } }
])
```

```
{
  average_duration: 443,
  route_id: 25,
  origin: 'Pingshan',
  destination: 'Krapina',
  distance_km: 176,
  estimated_time_hr: 8
}
{
  average_duration: 360,
  route_id: 23,
  origin: 'Zhongshanmen',
  destination: 'Wupu',
  distance_km: 412,
  estimated_time_hr: 6
}
```

```
{
  average_duration: 330,
  route_id: 2,
  origin: 'Sasar',
  destination: 'Yangzi',
  distance_km: 428,
  estimated_time_hr: 6
}
{
  average_duration: 300,
  route_id: 17,
  origin: 'Bhairāhawā',
  destination: 'Magaria',
  distance_km: 59,
  estimated_time_hr: 6
}
```

Coding and Sample Pipeline Output for Query 11: Average Trip Duration by Route.

Query 12: Busiest Routes by Departure Hour

This query shows which routes have the most departures at different times of day. It can be used to identify peak travel periods and help schedule additional buses or staff.

```
project> db.trip_logs.aggregate([
  {
    $project: {
      departure_hour: {
        $hour: {
          $cond: [
            { $eq: [{ $type: "$departure_time" }, "string"] },
            { $dateFromString: { dateString: "$departure_time" } },
            "$departure_time"
          ]
        }
      },
      route_id: 1
    }
  },
  {
    $group: {
      _id: {
        route_id: "$route_id",
        hour: "$departure_hour"
      },
      total_trips: { $sum: 1 }
    }
  },
  {
    $lookup: {
      from: "routes",
      localField: "_id.route_id",
      foreignField: "route_id",
      as: "route_info"
    }
  },
  { $unwind: "$route_info" },
  {
    $project: {
      route_id: "$_id.route_id",
      departure_hour: "$_id.hour",
      origin: "$route_info.origin",
      destination: "$route_info.destination",
      total_trips: 1,
      _id: 0
    }
  },
  { $sort: { total_trips: -1 } }
])
```

Coding and Sample Pipeline Output for Query 12: Busiest Routes by Departure Hour.

Query 13: Retrieve Total Confirmed Bookings by Routes

This query aims to understand which routes have the highest number of confirmed bookings to optimize vehicle and driver allocation where demand is greatest.

It helps the company teams to plan resources better where the marketing team focus on popular routes, and executives make informed decisions about route expansion.

```
[
  { $match: { status: "Confirmed" } },
  {
    $group: {
      _id: "$route_id",
      confirmed_count: { $sum: 1 }
    }
  },
  {
    $lookup: {
      from: "Routes",
      localField: "_id",
      foreignField: "route_id",
      as: "route_info"
    }
  },
  { $unwind: "$route_info" },
  {
    $project: {
      _id: 0,
      route: {
        $concat: [
          "$route_info.origin",
          " → ",
          "$route_info.destination"
        ]
      },
      confirmed_count: 1
    }
  },
  { $sort: { confirmed_count: -1 } }
]
```

```
confirmed_count : 3
route : "Lima → Cabugao"

confirmed_count : 2
route : "Banggel → Mojokerto"

confirmed_count : 2
route : "Tomilino → Tyrawa Wołoska"

confirmed_count : 1
route : "Oyan → Zhangting"

confirmed_count : 1
route : "Grimshaw → Thị Trấn Việt Lâm"

confirmed_count : 1
route : "Naifalo → Isanlu Itedoijowa"

confirmed_count : 1
route : "Rowokangkung → Dobra"
```

Coding and Sample Pipeline Output for Query 13: Retrieve Total Confirmed Bookings by Routes.

Query 14: Monthly Booking Volume Trend

This query helps to visualize booking frequency by month to identify patterns and anticipate high-demand periods for staffing and vehicle readiness by tracking how booking volumes change month by month.

With this information, the business and finance teams can prepare for busy or slow periods and supports marketing department in timing promotions effectively.

```
[
  {
    $addFields: {
      month: {
        $dateToString: {
          format: "%Y-%m",
          date: { $toDate: "$trip_date" }
        }
      }
    }
  },
  {
    $group: {
      _id: "$month",
      booking_total: { $sum: 1 }
    }
  },
  { $sort: { _id: 1 } }
]
```

```
_id: "2025-11"
booking_total : 6

_id: "2025-12"
booking_total : 7

_id: "2026-01"
booking_total : 4

_id: "2026-02"
booking_total : 5

_id: "2026-03"
booking_total : 4
```

Coding and Sample Pipeline Output for Query 14: Monthly Booking Volume Trend.

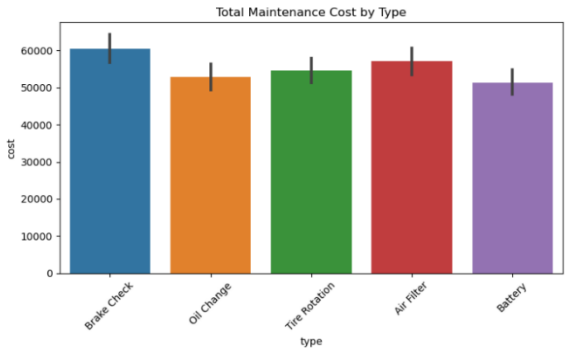
Query 15: The Routes with The Most Cancelled Bookings

This query helps identify potential problem routes (such as long distances and unstable services) by counting the cancellation of each route and combining them with route distances, to formulate more targeted improvement measures and improve user satisfaction and operational efficiency.

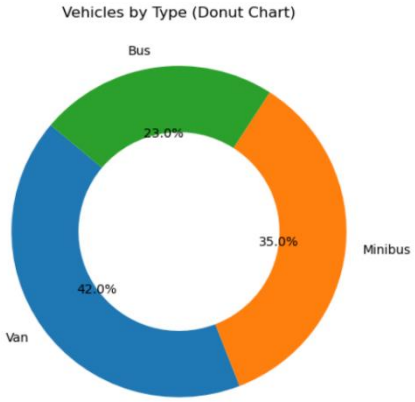
```
[
  {
    $match: {
      status: "Cancelled"
    }
  },
  {
    $group: {
      _id: "$route_id",
      cancelledBookings: {
        $sum: 1
      }
    }
  },
  {
    $lookup: {
      from: "Routes",
      localField: "_id",
      foreignField: "route_id",
      as: "routeInfo"
    }
  },
  {
    $unwind: "$routeInfo"
  },
  {
    $project: {
      _id: 0,
      route_id: "$_id",
      origin: "$routeInfo.origin",
      destination: "$routeInfo.destination",
      distance_km: "$routeInfo.distance_km",
      cancelledBookings: 1
    }
  },
  {
    $sort: {
      cancelledBookings: -1
    }
  }
]
```

Coding for Query 15: The Routes with the Most Cancelled Bookings.

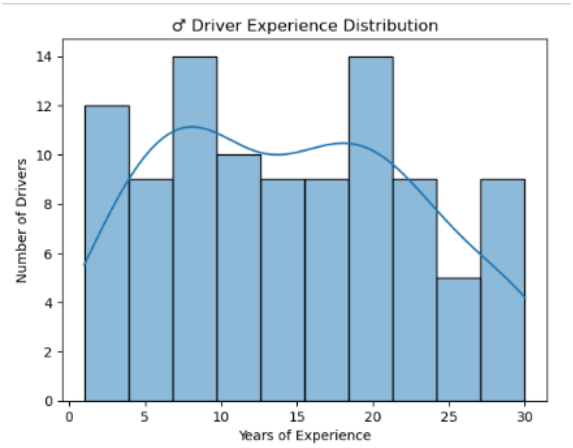
4.2 Visual Representation of Key Findings



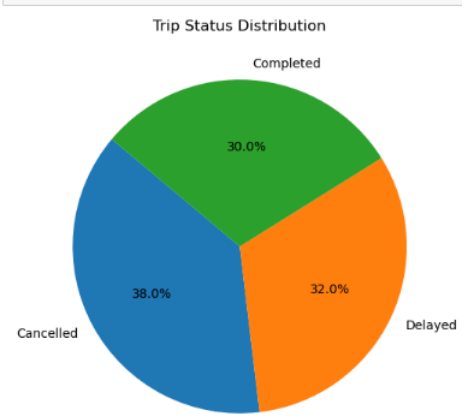
Graph 1



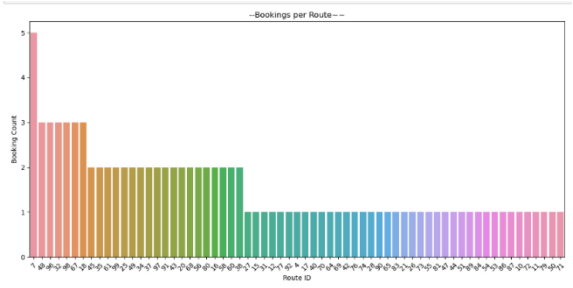
Graph 2



Graph 3



Graph 4



Graph 5

Graph 1 is a bar chart that displays total maintenance activity categorized by maintenance type. It includes services such as brake checks, oil changes, tire rotations, air filter replacements, and battery-related work. This chart is valuable for identifying which types of maintenance occur most frequently or incur the highest costs. In this case, the most significant maintenance type is breaking check, which is notably higher compared to battery change. These trends can indicate recurring vehicle issues or inform targeted preventative maintenance policies.

Graph 2 presents a donut chart that breaks down the fleet by vehicle type, showing the percentage each type contributes to the total. This visual helps to understand the fleet structure and how it aligns with operational needs. The largest share of the fleet consists of van, accounting for 42%, whereas bas represents only 23%. This distribution may reflect the nature of services provided whether local, regional, or long-distance transport.

Graph 3 is a histogram-style line graph illustrating driver experience across the organization. Two peaks emerge, located approximately at 10 and 20 years of experience, each representing about 14 drivers, suggesting a workforce that leans toward mid-career professionals. This data can guide human resource planning, including mentorship opportunities and training initiatives for less experienced drivers.

Graph 4 uses a pie chart to depict the distribution of trip outcomes categorized as completed, delayed, or cancelled. The data indicates that the highest percentage of trips falls under the cancelled category, which comprises 38% of all trips. In contrast, completed accounts for only 30%. The high rate of cancellation trips may point to operational bottlenecks, such as unreliable vehicles, poor scheduling, or external disruptions.

Graph 5 presents a vertical bar chart showing the number of bookings for each route, categorized by route ID. Route ID: 7 has the highest booking count, suggesting that it is the most frequently used or in highest demand. It significantly surpasses other Route ID which had relatively fewer bookings. Recognizing the most utilized routes helps in optimizing fleet allocation and service planning, ensuring that resources align with operational demands.

These visualizations offer a comprehensive overview of the transport system’s performance. They allow stakeholders to identify inefficiencies, prioritize maintenance, assess workforce capabilities, and evaluate route demand. By filling in the key figures and comparisons, this analysis becomes a valuable foundation for strategic improvements and informed decision making.

4.3 SYSTEM DEVELOPMENT CHALLENGES

In the development of this transportation management system, the main challenges faced were during database modeling and performance optimization. Since the system involves multiple data entities, including drivers, vehicles, routes, trips_logs, bookings and maintenance records, there are complex many-to-many or one-to-many relationships between these entities. Therefore, how to reasonably express these data structures in MongoDB's document model has become the primary problem in the modeling phase. At the same time, the system needs to support a large number of aggregate queries based on time, status and ID, which places high demands on the database index strategy. If the index is not set properly, it will directly affect the query response speed and the overall performance of the system. In addition, since MongoDB does not have the standardized transaction mechanism of traditional relational databases, how to ensure the consistency and integrity of data operations is also an issue that needs to be considered in system design.

4.4 DECISIONS AND JUSTIFICATIONS

To address the above challenges, we made a series of key technical and architectural decisions during the development process. In terms of data modeling, the "reference" method is implemented to establish associations between collections. For example, in the Trip_logs collection, driver_id, vehicle_id and route_id are referenced instead of nesting the entire document to maintain data consistency and reduce redundancy.

During the data cleaning phase, Python scripts were used to standardize the raw data, including unifying the date and time format and removing non-compliant fields and saving the processed data as a JSON file for import into the database. In addition, to improve the query performance of the system, we established multiple single-field and composite indexes based on business needs, covering high-frequency query fields such as trip_date, route_id and origin-destination combination fields, to ensure that the system can maintain a high response speed when facing a large number of query requests.

4.5 DISCUSSION OF NOSQL IN TRANSPORTATION CONTEXT

As a mainstream NoSQL database, MongoDB has shown many advantages in this transportation management system project. First, its flexible document model supports efficient storage of unstructured and semi-structured data, which is very suitable for data types with diverse formats and frequent updates in transportation systems, such as travel logs, reservation information, driver records and maintenance history. Secondly, MongoDB supports powerful aggregation pipeline operations, which can implement complex multi-dimensional analysis tasks such as peak hour statistics and vehicle utilization evaluation through stages such as `$group`, `$match` and `$lookup`, can help in improving analysis efficiency. In addition, MongoDB's horizontal sharding function gives it good horizontal scalability, which can maintain query performance as data grows and meet the needs of continued expansion of data volume in the future.

In addition, MongoDB's feature of not requiring a predefined schema significantly improves development flexibility and iteration speed, which is especially suitable for the prototype stage and agile development mode where business requirements change frequently and data structures are not stable. In this project, we can quickly add new fields or structures based on new analysis requirements without complex data migration operations, which are usually costly in relational databases.

However, the NoSQL model also has certain limitations. MongoDB lacks a strong consistency transaction mechanism (especially in early versions), which makes it difficult to ensure consistency for transactional updates involving cross-collection, multi-step operations. Although the multi-document transaction function was introduced after version 4.0, its performance is still inferior to relational databases in high-concurrency scenarios. At the same time, although MongoDB provides `$lookup` to achieve inter-collection connections, this operation is essentially a nested iteration and the performance overhead is obvious when the data volume is large or the nesting level is complex, making it difficult to support frequent association queries.

Furthermore, MongoDB's support for data constraints (such as foreign keys, uniqueness and multi-table dependencies) is relatively weak and developers need to control data consistency and integrity at the application layer. When faced with application scenarios with strict business rules and close logical dependencies, NoSQL model may introduce potential risks.

In summary, MongoDB is very suitable for scenarios in this project that require structural flexibility, high query concurrency and scalability, such as dynamic itinerary data and real-time booking analysis;

but in modules that require strong transaction consistency and multi-table complex logic, the data model should be designed carefully, or a hybrid architecture should be implemented in combination with a relational database to take advantage of both.

5.0 CONCLUSION

This project successfully built a transportation management data system based on MongoDB to simulate real-life business processes such as vehicle dispatch, passenger reservation, trip record and maintenance tracking. The system uses document-based data modeling, combined with index optimization and aggregate query, to achieve efficient management and multi-dimensional analysis of transportation operation data.

Using MongoDB brings significant advantages, including flexible data structure, excellent scalability and good query efficiency for this project. Its modeless document design is particularly suitable for the diverse and semi-structured data types in the transportation system, which can quickly respond to business changes and support complex analysis tasks such as route optimization, maintenance prediction and peak period identification.

In the future, the system can be further expanded in the following directions: introducing real-time data stream analysis to monitor vehicle dynamics and passenger flow; integrating geographic location data to achieve map visualization and route navigation; building machine learning models to improve maintenance prediction accuracy; and connecting with mobile applications or front-end systems to achieve a complete data-driven service platform. These improvements will further enhance the intelligence level of the system and help transportation companies continue to develop in the direction of digitalization and automation.

REFERENCES

[1] Alam, M., & Rahman, A. (2020). Comparative performance evaluation of NoSQL databases for big data applications. *International Journal of Advanced Computer Science and Applications, 11*(4), 96–102.

[2] Alflahi, A. A. E., Mohammed, M. A. Y., & Alsammani, A. (2023). Enhancement of database access performance by improving data consistency in a non-relational database system (NoSQL). *arXiv preprint arXiv:2302.11974*.

[3] Ali, S., Shah, A., & Khan, M. (2020). A comparative study of MongoDB and MySQL databases. *International Journal of Computer Applications, 176*(35), 10–13.

[4] Bisht, M., & Joshi, R. (2023). Performance Analysis of MongoDB in Semi-structured Data Environments. *International Journal of Computer Science and Information Security, 21*(2), 34–41.

[5] Chidambaram, R., & Prakash, S. (2021). Evaluation of relational and non-relational databases. *Turkish Journal of Computer and Mathematics Education, 12*(10), 1844–1850.

[6] Gupta, S., & Mahajan, R. (2024). Real-time data analytics in NoSQL databases: A transportation case study. *Journal of Information Systems and Technology Management, 36*(1), 22–31.

[7] Jain, A., & Saxena, A. (2022). Aggregation and sharding in MongoDB: A practical approach for scalable applications. *International Journal of Database Theory and Application, 15*(4), 112–118.

[8] Kapoor, R., & Gupta, A. (2021). NoSQL data models for high-frequency retail applications. *Journal of Retail Data Science, 3*(2), 40–49.

[9] Kumar, A., & Ghosh, R. (2023). MongoDB adoption in modern online commerce systems. *International Journal of e-Business Research, 19*(1), 55–63.

[10] Mehta, D., & Shah, V. (2021). Leveraging NoSQL for modern streaming platforms. *Journal of Digital Media Systems, 8*(3), 76–85.

[11] Natarajan, P., Singhal, R., & Bose, D. (2024). User-centric data modeling with NoSQL: Applications in streaming and IoT. *IEEE Access, 12*, 11482–11494.

[12] Patel, S., Kumar, R., & Sinha, V. (2021). Comparative performance evaluation of NoSQL and relational databases in IoT environments. *International Journal of Computer Engineering and Applications, 15*(7), 122–130.

[13] Ramesh, T., & Kumar, V. (2021). Data modeling challenges in transport logistics using NoSQL databases. *International Journal of Logistics Research, 9*(1), 16–26.

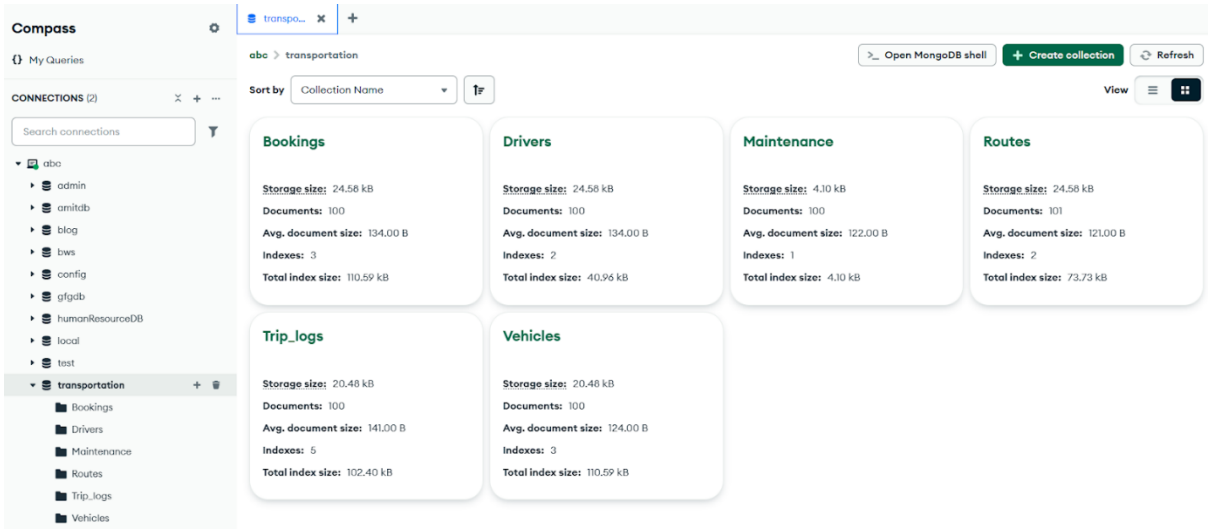
[14] Rana, S., Bansal, A., & Malhotra, P. (2020). Leveraging document-oriented databases in e-commerce: A case study on MongoDB. *International Journal of Software Engineering and Applications, 11*(6), 88–97.

[15] Singh, P., & Sharma, K. (2022). MongoDB-based system design for scalable retail infrastructure. *International Journal of Information Systems and Retail Analytics, 10*(2), 13–20.

APPENDIX A: SAMPLE JSON DOCUMENTS FOR CLEANED_DRIVERS DATASET


```
[
  {
    "driver_id": 1,
    "name": "Angel Farenden",
    "license_number": "DR0",
    "assigned_vehicle_id": 71,
    "experience_years": 30
  },
  {
    "driver_id": 2,
    "name": "Happy Prettyjohns",
    "license_number": "DR3",
    "assigned_vehicle_id": 87,
    "experience_years": 8
  },
  {
    "driver_id": 3,
    "name": "Cherey Carmichael",
    "license_number": "DR87",
    "assigned_vehicle_id": 100,
    "experience_years": 2
  },
  {
    "driver_id": 4,
    "name": "Ivette Van den Velde",
    "license_number": "DR71",
    "assigned_vehicle_id": 10,
    "experience_years": 11
  },
  {
    "driver_id": 11,
    "name": "Oralie Vanacci",
    "license_number": "DR157",
    "assigned_vehicle_id": 84,
    "experience_years": 21
  },
  {
    "driver_id": 12,
    "name": "Halette Ickovits",
    "license_number": "DR416",
    "assigned_vehicle_id": 7,
    "experience_years": 20
  },
  {
    "driver_id": 13,
    "name": "Jackie Iacovielli",
    "license_number": "DR421",
    "assigned_vehicle_id": 84,
    "experience_years": 20
  },
  {
    "driver_id": 14,
    "name": "Lavina Seeviour",
    "license_number": "DR4",
    "assigned_vehicle_id": 7,
    "experience_years": 16
  },
  {
    "driver_id": 20,
    "name": "Ted Craisford",
    "license_number": "DR81",
    "assigned_vehicle_id": 29,
    "experience_years": 30
  },
  {
    "driver_id": 21,
    "name": "Clarence Busse",
    "license_number": "DR761",
    "assigned_vehicle_id": 19,
    "experience_years": 29
  },
  {
    "driver_id": 22,
    "name": "Wendie Jarmaine",
    "license_number": "DR80",
    "assigned_vehicle_id": 37,
    "experience_years": 17
  },
  {
    "driver_id": 23,
    "name": "Angelle Higgan",
    "license_number": "DR59",
    "assigned_vehicle_id": 20,
    "experience_years": 15
  },
```

APPENDIX B: LIST OF COLLECTIONS IN MONGODB COMPASS



APPENDIX C: SAMPLE DATA IN VEHICLE AND ROUTES COLLECTION