Shuttle Management System

Milestone: Project Report

Group 23
Venkata Satya Surya Sai Adhithya Akella
Saketh Kottissa

857-313-5414 (Adhithya Akella)
617-238-8114 (Saketh Kottissa)

akella.ven@northeastern.edu
kottissa.s@northeastern.edu

Percentage of Effort Contributed by Adhithya Akella: 50%
Percentage of Effort Contributed by Saketh Kottissa: 50%

Signature of Student 1: Adhithya Akella
Signature of Student 2: Saketh Kottissa

Submission Date: 04/23/23

Background:

Shuttle management systems frequently rely on manual processes and disorganized data sources, which can result in scheduling conflicts, inefficient resource utilization, and a negative customer experience. Our work aims to address these issues by developing and deploying a comprehensive database system that allows for real-time tracking and management of buses, drivers, passengers, routes, schedules, and garages. Shuttle service providers can use this system to streamline their operations, optimize resource allocation, and provide a more reliable and customer-friendly service.

Problem definition

There are several problems that can arise with a shuttle service, including:

Scheduling conflicts: Shuttle routes and schedules may not align with the needs of passengers, leading to long wait times or missed connections.

Limited capacity: Shuttles may not have enough room to accommodate all passengers, particularly during peak travel times.

Traffic congestion: Shuttles may get stuck in traffic, causing delays and making it difficult for passengers to arrive at their destination on time.

Cost: Operating a shuttle service can be expensive, particularly if the service covers a large geographic area or serves a low number of passengers.

Maintenance and repair: Keeping the shuttles in good working condition can be costly and time-consuming.

Competition: Shuttle services may face competition from other forms of transportation, such as cars, trains, and buses, which may make it difficult for the service to attract and retain customers.

A shuttle management system is an application that helps to manage and optimize the operations of a shuttle transportation provider.

The problem that a shuttle management system aims to solve is to improve the efficiency and profitability of the bus transportation provider by automating and streamlining various tasks and processes. This can include reducing operating costs, increasing revenue, improving customer service, and providing real-time visibility into the providers operations.

The system can be used by transportation companies, schools, universities, airports and other organizations that operate a fleet of shuttles.

The system can include the following functionalities:

- Scheduling and dispatching of buses
- Route planning and optimization
- Ticketing and fare management
- Fleet management and maintenance
- Passenger tracking and safety

- Real-time tracking of buses
- Reporting and analytics
- Customer service and support

The system can be used by multiple stakeholders such as:

- Bus operators
- Drivers
- Passengers
- Administrators
- Maintenance staff
- Dispatch staff
- Accountants

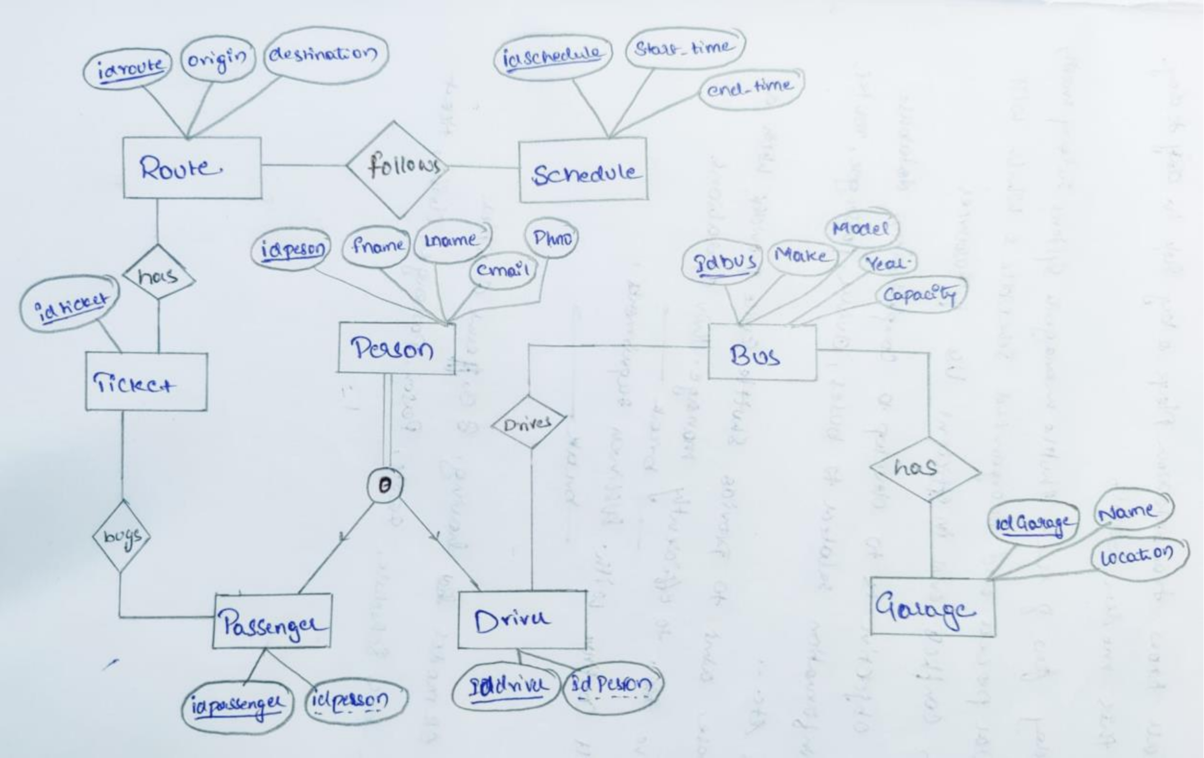
Tables Description:

A shuttle schedule management system using SQL would typically involve several tables to store and organize the data. Here are a few examples of tables that will be used:

1. **Bus:** This table would store information about each bus, such as its make, model, capacity, and maintenance schedule.
2. **Route:** This table would store information about each bus route, such as the route number, stops, and schedule.
3. **Schedule:** This table would store information about the schedule for each bus route, including the departure and arrival times for each stop, as well as the frequency of the bus.
4. **Person:** This table would contain about all the people associated to the Shuttle management system, further divided into passenger and driver.
5. **Driver:** This table would store information about each bus driver, including their name, contact information, and schedule.
6. **Ticket:** This table would store information about the tickets issued to passengers, including the route, the stop, the time, and the fare.
7. **Passenger:** This table would store information about the passengers, including their name, contact information, and their trip history.
8. **Garage:** This table would store information about the stationing of each bus.

Conceptual Data Modelling

EER diagram



Mapping Conceptual Model to Relational Model

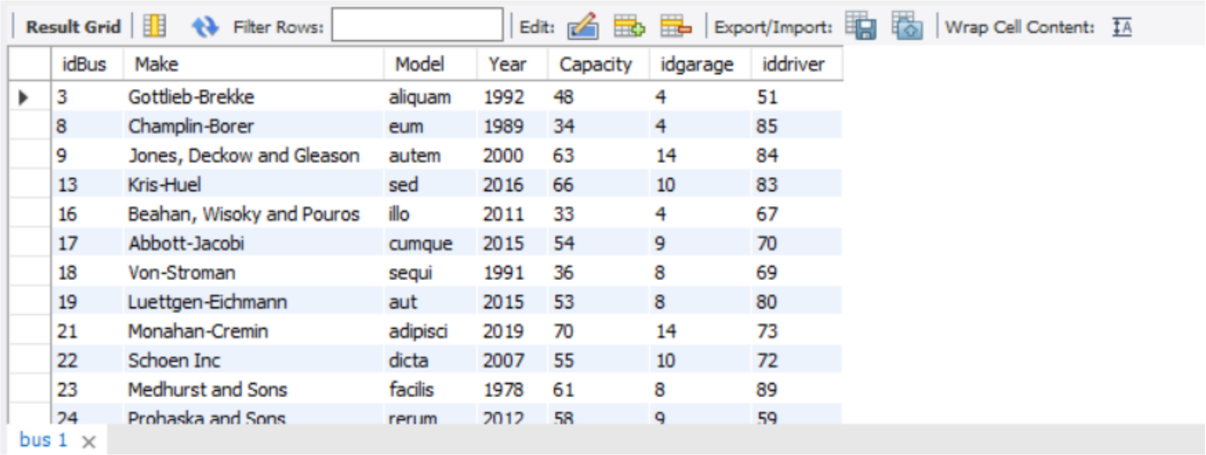
1. Bus: (idbus, Make, Model, Year, Capacity, idgarage, iddriver)
2. Driver: (iddriver, license_number, idperson)
3. Garage: (idgarage, Name, Location)
4. Passenger: (idpassenger, idperson)
5. Person: (idperson, first_name, last_name, email, phone_number)
6. Route: (idroute, origin, destination)
7. Schedule:(idschedule, start_time, end_time, idroute)
8. Ticket: (idticket, idpassenger, idroute)

Implementation of Relation Model via MySQL and NoSQL:

1. MySQL Implementation:

Query1: To use the database and display all the buses in the database.

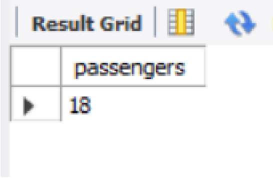
```
use Shuttle_Management_System;
select *from bus
```



idBus	Make	Model	Year	Capacity	idgarage	iddriver
3	Gottlieb-Brekke	aliquam	1992	48	4	51
8	Champlin-Borer	eum	1989	34	4	85
9	Jones, Deckow and Gleason	autem	2000	63	14	84
13	Kris-Huel	sed	2016	66	10	83
16	Beahan, Wisoky and Pouros	illo	2011	33	4	67
17	Abbott-Jacobi	cumque	2015	54	9	70
18	Von-Stroman	sequi	1991	36	8	69
19	Luettgen-Eichmann	aut	2015	53	8	80
21	Monahan-Cremin	adipisci	2019	70	14	73
22	Schoen Inc	dicta	2007	55	10	72
23	Medhurst and Sons	facilis	1978	61	8	89
24	Prohaska and Sons	rerum	2012	58	9	59

Query2: To count the passengers travelling through a particular route.

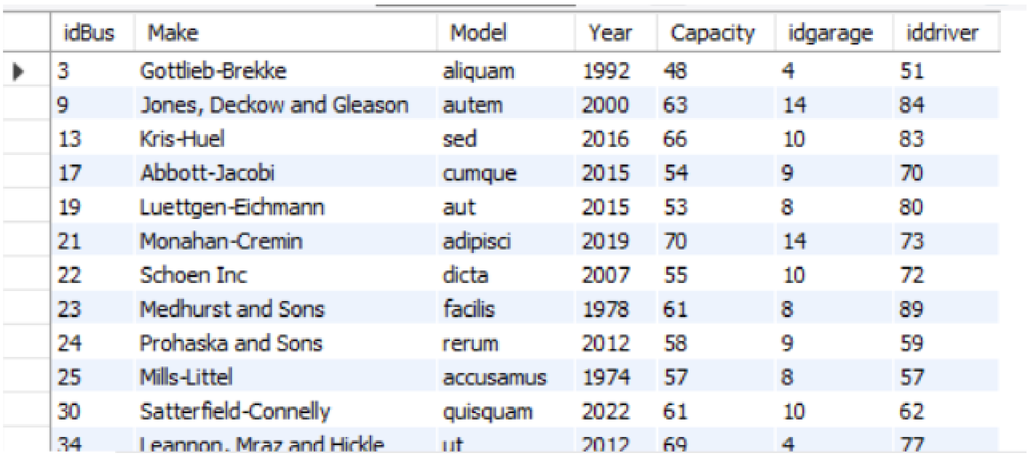
```
SELECT COUNT(*) as passengers
FROM Ticket
WHERE idroute = 14;
```



passengers
18

Query3: To find the buses with capacity greater than 45.

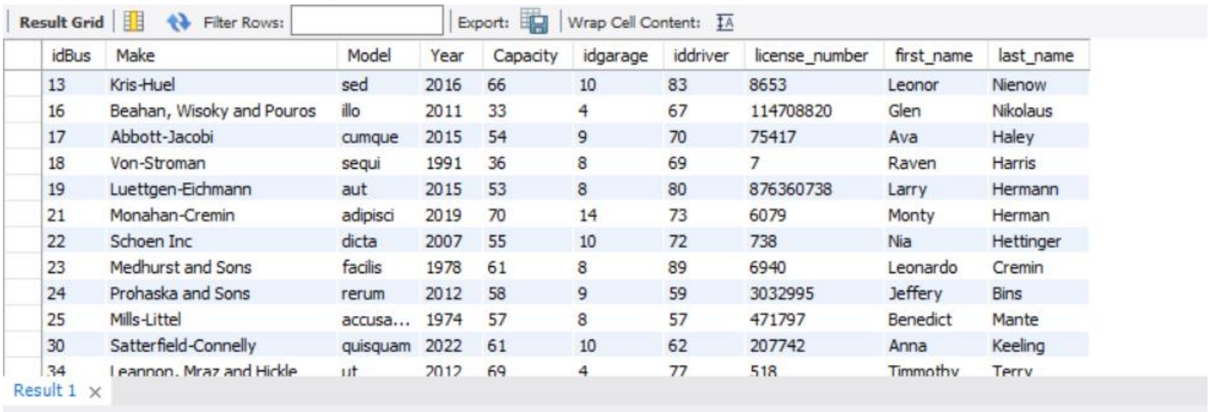
```
SELECT * FROM shuttle_management_system.bus where Capacity > 45;
```



idBus	Make	Model	Year	Capacity	idgarage	iddriver
3	Gottlieb-Brekke	aliquam	1992	48	4	51
9	Jones, Deckow and Gleason	autem	2000	63	14	84
13	Kris-Huel	sed	2016	66	10	83
17	Abbott-Jacobi	cumque	2015	54	9	70
19	Luettgen-Eichmann	aut	2015	53	8	80
21	Monahan-Cremin	adipisci	2019	70	14	73
22	Schoen Inc	dicta	2007	55	10	72
23	Medhurst and Sons	facilis	1978	61	8	89
24	Prohaska and Sons	rerum	2012	58	9	59
25	Mills-Littel	accusamus	1974	57	8	57
30	Satterfield-Connelly	quisquam	2022	61	10	62
34	Leannon, Mraz and Hickle	ut	2012	69	4	77

Query4: To find all the bus details along with diver details.

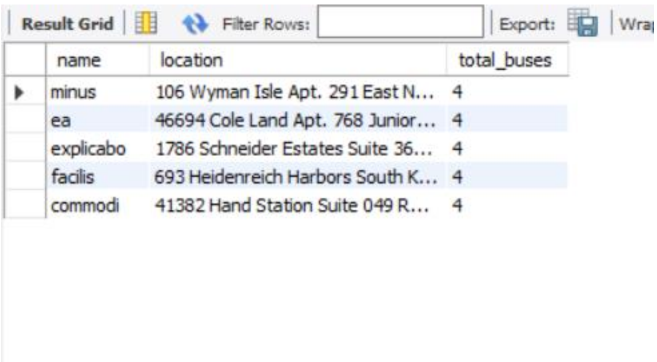
SELECT bus.*, driver.license_number, person.first_name, person.last_name
FROM bus
JOIN driver ON bus.iddriver = driver.iddriver
JOIN person ON driver.idperson = person.idperson;



idBus	Make	Model	Year	Capacity	idgarage	iddriver	license_number	first_name	last_name
13	Kris-Huel	sed	2016	66	10	83	8653	Leonor	Nienow
16	Beahan, Wisoky and Pouros	illo	2011	33	4	67	114708820	Glen	Nikolaus
17	Abbott-Jacobi	cumque	2015	54	9	70	75417	Ava	Haley
18	Von-Stroman	sequi	1991	36	8	69	7	Raven	Harris
19	Luettgen-Eichmann	aut	2015	53	8	80	876360738	Larry	Hermann
21	Monahan-Cremin	adipisci	2019	70	14	73	6079	Monty	Herman
22	Schoen Inc	dicta	2007	55	10	72	738	Nia	Hettinger
23	Medhurst and Sons	facilis	1978	61	8	89	6940	Leonardo	Cremin
24	Prohaska and Sons	rerum	2012	58	9	59	3032995	Jeffery	Bins
25	Mills-Littel	accusa...	1974	57	8	57	471797	Benedict	Mante
30	Satterfield-Connelly	quisquam	2022	61	10	62	207742	Anna	Keeling
34	Leannon, Mraz and Hickle	ut	2012	69	4	77	518	Timmothy	Terry

Query 5: To retrieve the details of the garages with the most buses assigned to them (Nested)

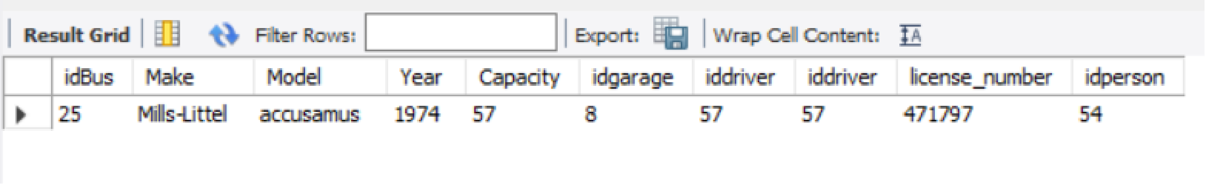
SELECT garage.name, garage.location, COUNT(*) as total_buses
FROM garage
JOIN bus ON garage.idgarage = bus.idgarage
GROUP BY garage.idgarage
HAVING COUNT(*) >= ALL(SELECT COUNT(*) FROM bus GROUP BY idgarage);



name	location	total_buses
minus	106 Wyman Isle Apt. 291 East N...	4
ea	46694 Cole Land Apt. 768 Junior...	4
explicabo	1786 Schneider Estates Suite 36...	4
facilis	693 Heidenreich Harbors South K...	4
commodi	41382 Hand Station Suite 049 R...	4

Query 6: Retrieve the details of the oldest bus and its driver (Correlated)

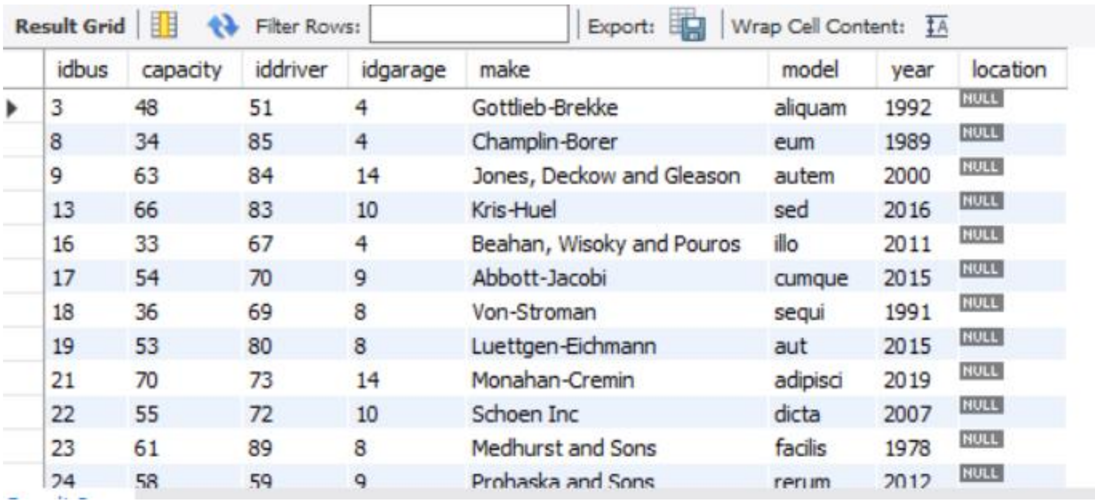
```
SELECT *
FROM bus b
JOIN driver d ON b.iddriver = d.iddriver
WHERE b.year = (
   SELECT MIN(year)
   FROM bus
) ;
```



idBus	Make	Model	Year	Capacity	idgarage	iddriver	iddriver	license_number	idperson
25	Mills-Littel	accusamus	1974	57	8	57	57	471797	54

Query 7: Retrieve the details of all buses and garages (Union)

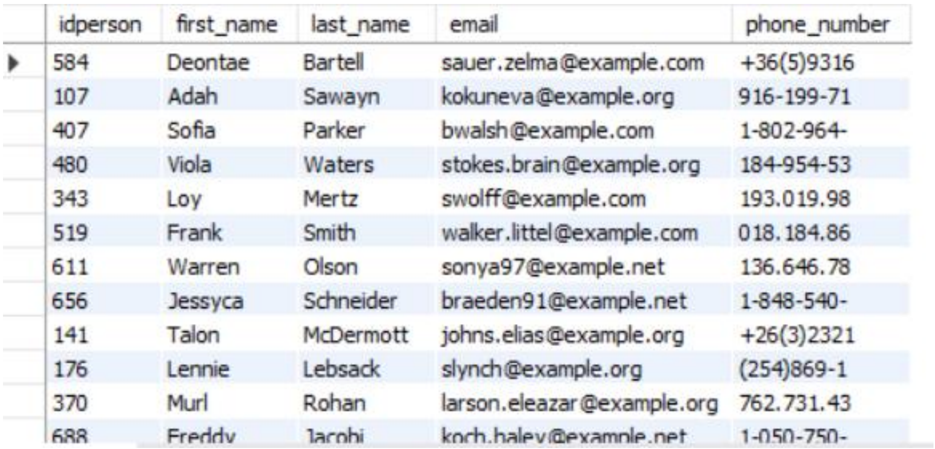
```
SELECT idbus, capacity, iddriver, idgarage, make, model, year, NULL as location
FROM Bus
UNION
SELECT NULL as idbus, NULL as capacity, NULL as iddriver, idgarage, NULL as make, NULL
     as model, NULL as year, location
FROM Garage;
```



idbus	capacity	iddriver	idgarage	make	model	year	location
3	48	51	4	Gottlieb-Brekke	aliquam	1992	NULL
8	34	85	4	Champlin-Borer	eum	1989	NULL
9	63	84	14	Jones, Deckow and Gleason	autem	2000	NULL
13	66	83	10	Kris-Huel	sed	2016	NULL
16	33	67	4	Beahan, Wisoky and Pouros	illo	2011	NULL
17	54	70	9	Abbott-Jacobi	cumque	2015	NULL
18	36	69	8	Von-Stroman	sequi	1991	NULL
19	53	80	8	Luettgen-Eichmann	aut	2015	NULL
21	70	73	14	Monahan-Cremin	adipisci	2019	NULL
22	55	72	10	Schoen Inc	dicta	2007	NULL
23	61	89	8	Medhurst and Sons	facilis	1978	NULL
24	58	59	9	Prohaska and Sons	rerum	2012	NULL

Query 8: Subquery in FROM clause to retrieve the details of all passengers who have bought a ticket for a certain route (Subquery in select and from)

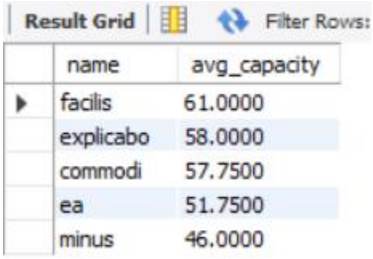
```
SELECT p.idperson, p.first_name, p.last_name, p.email, p.phone_number
FROM person p
INNER JOIN (SELECT DISTINCT idpassenger FROM ticket WHERE idroute = '14') t ON p.idperson = t.idpassenger;
```



idperson	first_name	last_name	email	phone_number
584	Deontae	Bartell	sauer.zelma@example.com	+36(5)9316
107	Adah	Sawayn	kokuneva@example.org	916-199-71
407	Sofia	Parker	bwalsh@example.com	1-802-964-
480	Viola	Waters	stokes.brain@example.org	184-954-53
343	Loy	Mertz	swolff@example.com	193.019.98
519	Frank	Smith	walker.littel@example.com	018.184.86
611	Warren	Olson	sonya97@example.net	136.646.78
656	Jessyca	Schneider	braeden91@example.net	1-848-540-
141	Talon	McDermott	johns.elias@example.org	+26(3)2321
176	Lennie	Lebsack	slynch@example.org	(254)869-1
370	Murl	Rohan	larson.eleazar@example.org	762.731.43
688	Freddy	Jacobi	koch.haley@example.net	1-050-750-

Query 9: Average capacity of buses in each garage: (aggregate)

```
SELECT g.name, AVG(b.capacity) AS avg_capacity
FROM garage g
INNER JOIN bus b ON g.idgarage = b.idgarage
GROUP BY g.name
ORDER BY avg_capacity DESC;
```

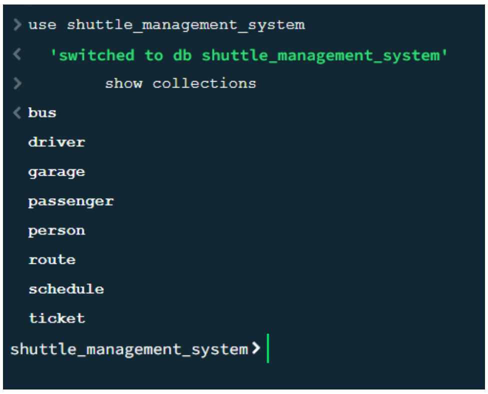


name	avg_capacity
facilis	61.0000
explicabo	58.0000
commodi	57.7500
ea	51.7500
minus	46.0000

2. NoSQL Implementation:

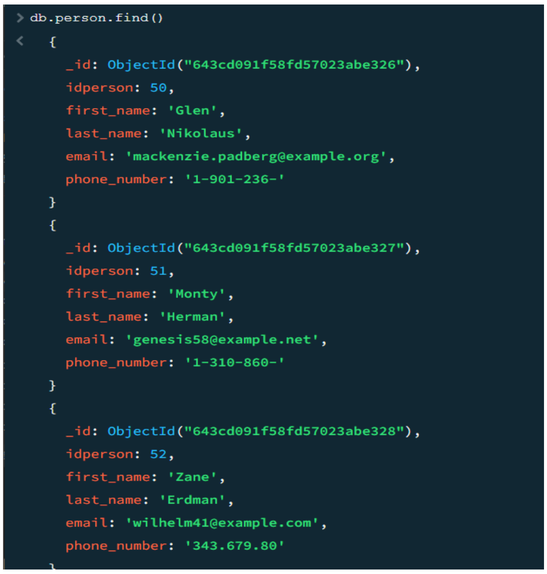
We have created a NoSQL database in mongodb and have successfully migrated our data from SQL tables to NoSQL collections in mongodb.

1) Collections present in our mongodb shuttle_management_system database.



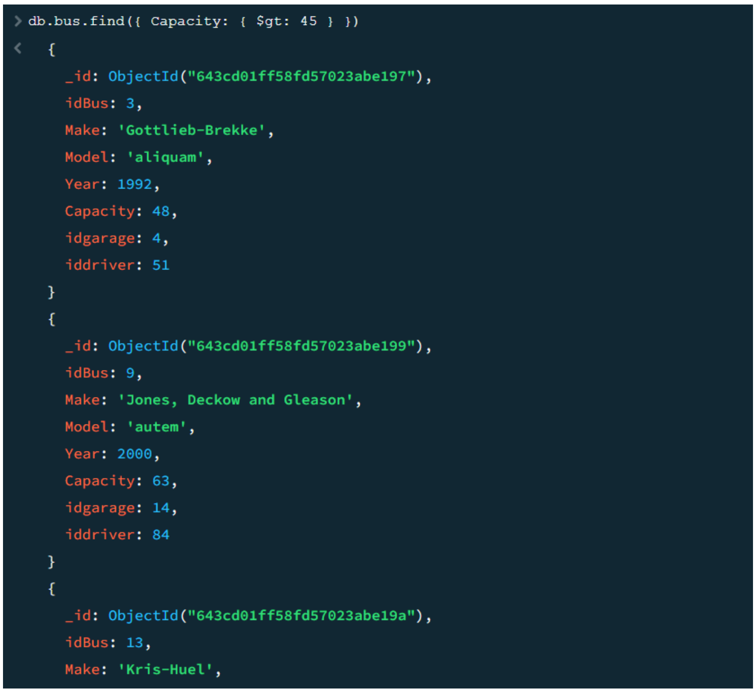
```
> use shuttle_management_system
<   'switched to db shuttle_management_system'
>         show collections
< bus
  driver
  garage
  passenger
  person
  route
  schedule
  ticket
shuttle_management_system>
```

2. Select query on person



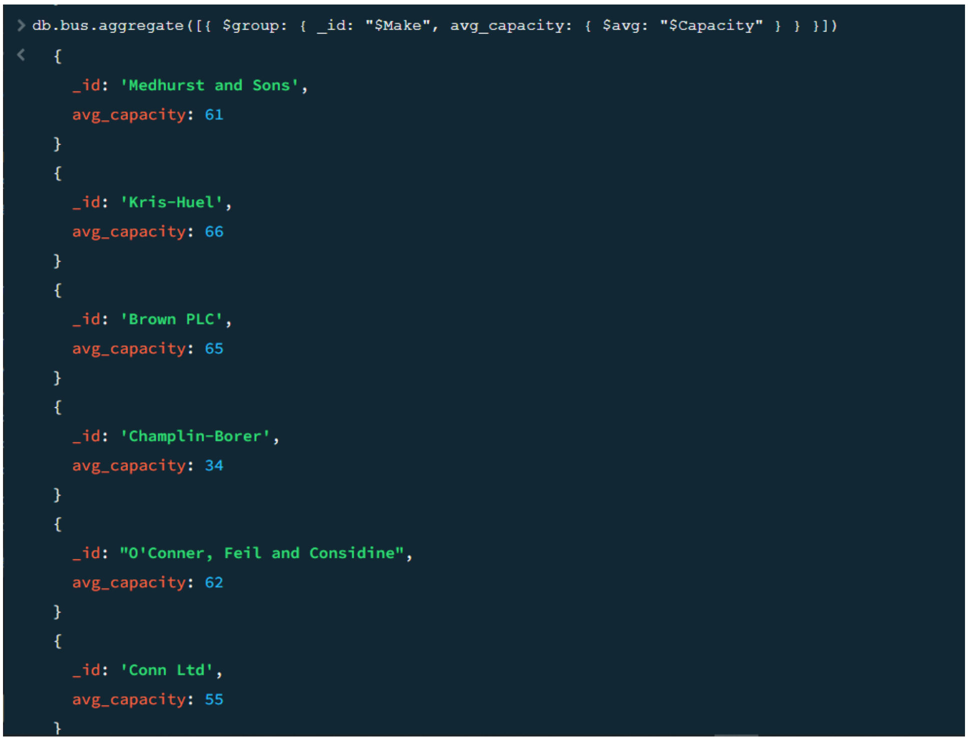
```
> db.person.find()
<   {
      _id: ObjectId("643cd091f58fd57023abe326"),
      idperson: 50,
      first_name: 'Glen',
      last_name: 'Nikolaus',
      email: 'mackenzie.padberg@example.org',
      phone_number: '1-901-236-'
    }
    {
      _id: ObjectId("643cd091f58fd57023abe327"),
      idperson: 51,
      first_name: 'Monty',
      last_name: 'Herman',
      email: 'genesis58@example.net',
      phone_number: '1-310-860-'
    }
    {
      _id: ObjectId("643cd091f58fd57023abe328"),
      idperson: 52,
      first_name: 'Zane',
      last_name: 'Erdman',
      email: 'wilhelm41@example.com',
      phone_number: '343.679.80'
    }
```

3. Query to find buses with capacity greater than 45



```
> db.bus.find({ Capacity: { $gt: 45 } })
< {
    _id: ObjectId("643cd01ff58fd57023abe197"),
    idBus: 3,
    Make: 'Gottlieb-Brekke',
    Model: 'aliquam',
    Year: 1992,
    Capacity: 48,
    idgarage: 4,
    iddriver: 51
  }
  {
    _id: ObjectId("643cd01ff58fd57023abe199"),
    idBus: 9,
    Make: 'Jones, Deckow and Gleason',
    Model: 'autem',
    Year: 2000,
    Capacity: 63,
    idgarage: 14,
    iddriver: 84
  }
  {
    _id: ObjectId("643cd01ff58fd57023abe19a"),
    idBus: 13,
    Make: 'Kris-Huel',
```

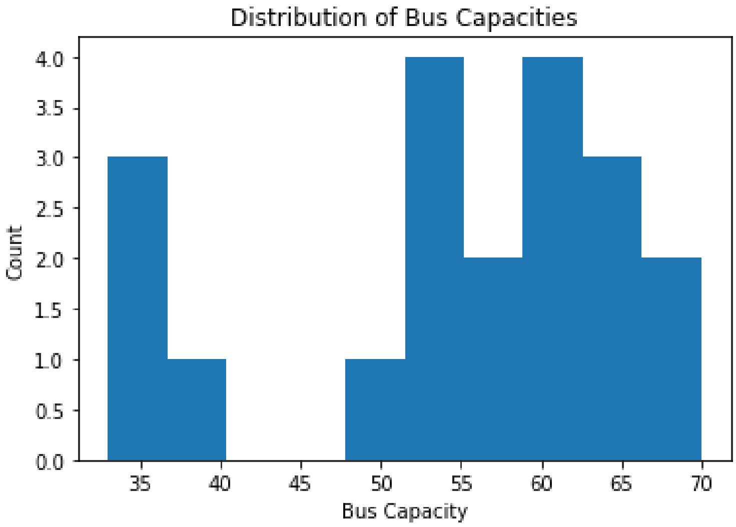
4. Query to average capacity of every bus make

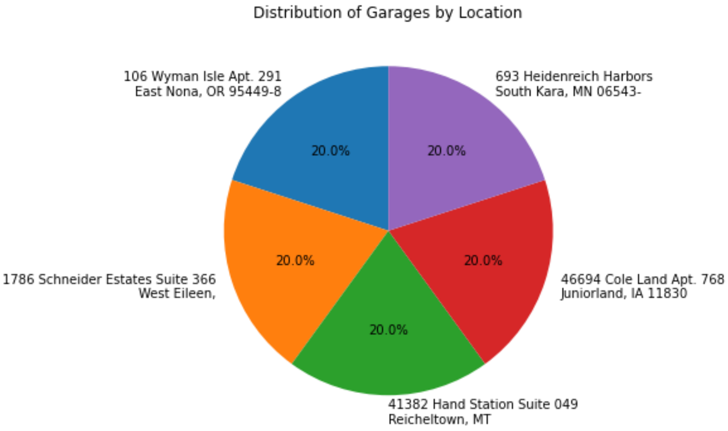


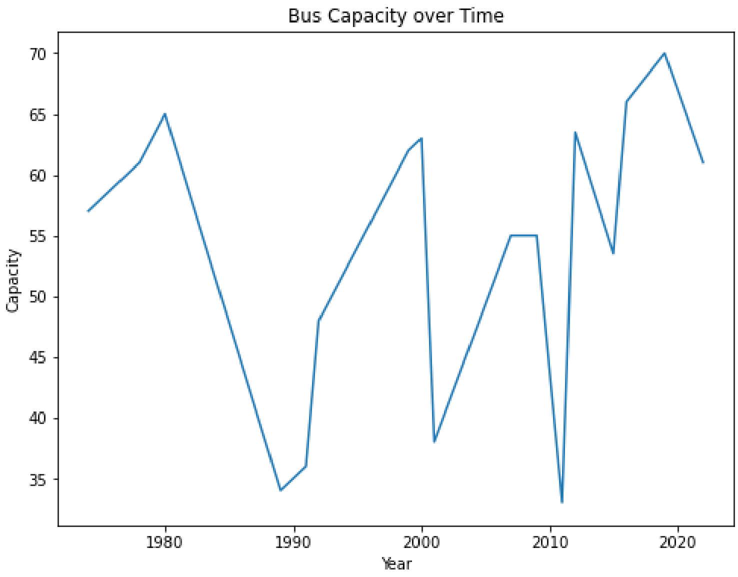
```
> db.bus.aggregate([{ $group: { _id: "$Make", avg_capacity: { $avg: "$Capacity" } } }])
< {
    _id: 'Medhurst and Sons',
    avg_capacity: 61
  }
  {
    _id: 'Kris-Huel',
    avg_capacity: 66
  }
  {
    _id: 'Brown PLC',
    avg_capacity: 65
  }
  {
    _id: 'Champlin-Borer',
    avg_capacity: 34
  }
  {
    _id: "O'Conner, Feil and Considine",
    avg_capacity: 62
  }
  {
    _id: 'Conn Ltd',
    avg_capacity: 55
  }
```

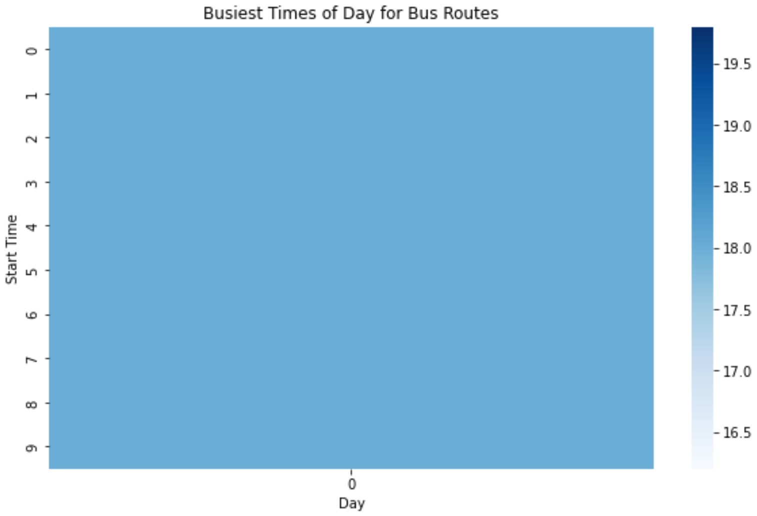
3. Database Access via Python:

Python is used to access the database. It uses the "mysql.connector" package, which has all functions for connections and SQL queries. To connect to the database “mysql.connector.connect” and to send a query to it a cursor is created, use “con.cursor()”, ”mycur.execute(query)”. The “mycur.fetchall()” function is used to retrieve the result set. The result set list is turned into a dataframe using the pandas package “pd.DataFrame()” function and using the seaborn package “sns.barplot”, “sns.lineplot”, etc.. for various graphs and additional analysis.









Summary:

There are eight tables in the database: Bus, Driver, Garage, Passenger, Person, Route, Schedule, and Ticket. To store related data, each table has its own set of columns. With tables for buses, drivers, garages, routes, schedules, and tickets, the database looks to be developed for a transportation management system.

Recommendations

Consider including similar information in embedded documents. Instead of maintaining the driver and garage IDs in the Bus collection, you could embed the Driver and Garage documents directly in the Bus document. This can improve query efficiency and simplify the data model.

Indexing can help you enhance query performance. To speed up queries that filter or sort by these fields, you may create indexes on the capacity field in the Bus collection and the origin and destination fields in the Route collection, for example.

To enhance performance, consider denormalizing data. For example, to make it easier to find routes that use buses from a specific manufacturer, you could replicate the make field from the Bus collection in the Route collection.

Consider utilizing a consistent and easy-to-understand naming convention for collection and field names. This can make writing queries and understanding the data model easier.

Run diagnostic tools like db.stats() on a regular basis to improve and manage the database, as well as monitor performance and apply best practices like data backups and disaster recovery plans.