

## Travel Advisor Database Management System

### Group 12

Student1 : Anjali Ingle  
Student2: Ameya Deshmukh

(617)-372-0345 (Tel of Student 1)  
857-396-7408 (Tel of Student 2)

ingle.a@northeastern.edu  
deshmukh.amey@notheastern.edu

Percentage of Effort Contributed by Student1: \_\_\_\_ 50 \_\_\_\_  
Percentage of Effort Contributed by Student2: \_\_\_\_ 50 \_\_\_\_

Signature of Student 1: \_\_\_\_ Anjali \_\_\_\_  
Signature of Student 2: \_\_\_\_ Ameya \_\_\_\_

Submission Date: \_\_\_\_ 12/10/23 \_\_\_\_

# **Travel Booking Management System Report**

## **Executive Summary:**

The Travel Advisory Management System is a one-stop platform for travel bookings. It's designed to make trip planning easier by storing details about cities, hotels, and transportation options, including prices. This system helps users find destinations, compare travel choices like flights or trains, and pick hotels. For administrators, it offers insights like popular destinations, favored travel methods, and average customer spending.

We meticulously designed our travel advisory system by initiating the creation of a comprehensive data model. This involved crafting Entity-Relationship (EER) and Unified Modeling Language (UML) diagrams, meticulously outlining the structure and relationships within our data.

Subsequently, we translated our conceptual designs into a practical implementation by creating a robust relational data model. This model served as the backbone for our MySQL database, facilitating the organization and management of our travel-related information.

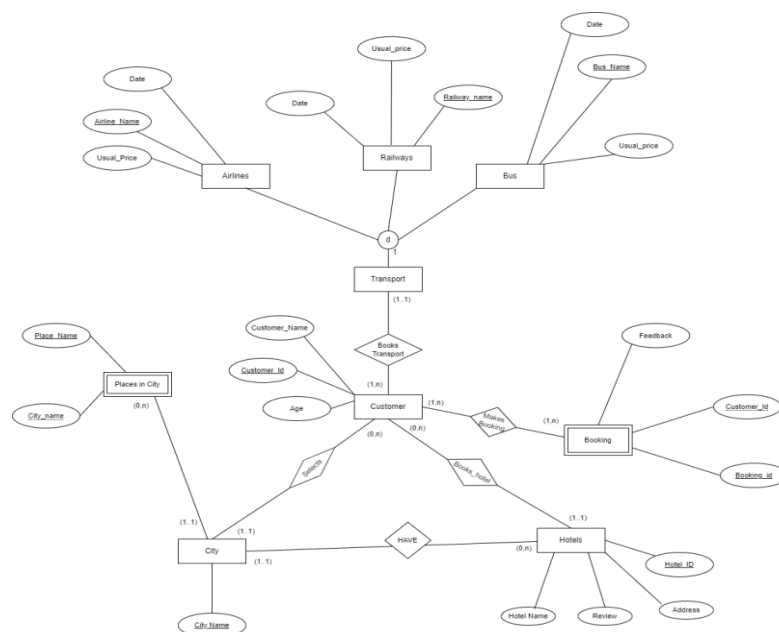
Utilizing this structured foundation, we seamlessly established connectivity between our MySQL data and MongoDB using Python scripts. Leveraging MongoDB Compass, we effortlessly translated our relational model into MongoDB collections.

Furthermore, our system was elevated through the integration of a user-friendly front-end developed in Streamlit. Incorporating insightful

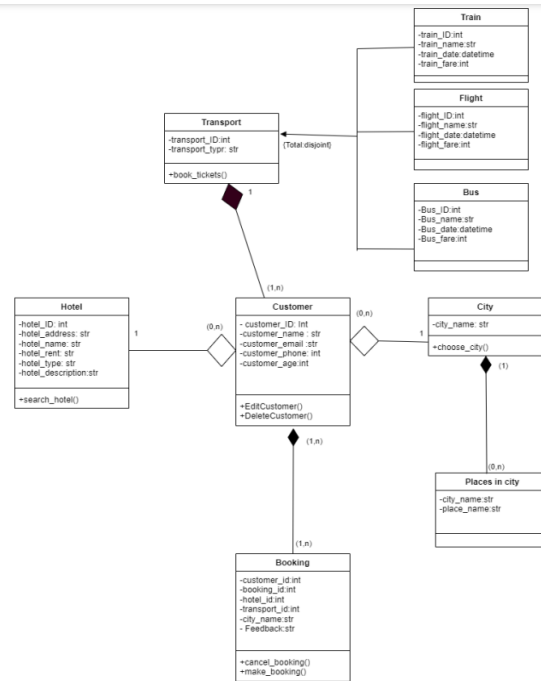
visualizations such as graphs, the interface showcased business analytics derived from our meticulously linked SQL and NoSQL databases.

In essence, our travel advisory platform embodies meticulous planning, conceptualization through EER and UML diagrams, structured data management via relational models, seamless integration with MongoDB, and a user-centric interface through Streamlit, culminating in an insightful and user-friendly travel advisory system.

### Conceptual Data Modelling :



### EER DIAGRAM



## UML DIAGRAM

### Relational Model:

Customer(Customer\_ID(PK),Customer\_Name,Age,City\_NAME(FK),Hotel\_ID,Transport\_ID)

Booking(Customer\_ID(PK),Booking\_id(PK),Feedback)

Hotel(Hotel\_ID,Address,Review,Hotel\_Name,City\_Name(FK))

City(City\_Name)

Places in city(Place\_Name(PK),City\_name(PK))

Transport (Transport\_ID(PK),Transport\_Type(PK) )

Railways (Transport\_ID(FK),Railway\_Name (PK),Date,Usual\_price)

Airlines (Transport\_ID(FK),Airline\_Name (PK),Date,Usual\_price)

Bus (Transport\_ID(FK),Bus\_Name (PK),Date,Usual\_price)

Hotels (Hotel\_ID,Address,Review,Hotel\_Name)

Makes\_Booking(Customer\_ID(FK),Booking\_id(FK))

### MYSQL Implementation :

1) Total bookings made by each customer

```
select c.customer_name,
```

```
t.count as 'Booked_using platform'
```

from customers c

join (select customer\_id,count(\*) as count

from booking\_data group by Customer\_ID)on c.customer\_id=t.customer\_id

	customer_name	Booked_using platform
▶	John Doe	1
	Jane Smith	3
	Alice Johnson	3
	Bob Williams	3
	Eva Brown	1
	Michael Clark	1
	Sophia Lee	5
	David Taylor	3
	Olivia White	2
	William Harris	1
	Ava Martin	3
	James Anderson	5
	Daniel Martinez	2
	Mia Lopez	5
	Alexander King	1
	Grace Hall	2
	Liam Young	3
	Isabella Lewis	3
	Noah Adams	3

2) Find Hotels With a High Number of Bookings for every City

select SELECT h.City\_Name, h.Hotel\_ID, h.Hotel\_Name,

COUNT(b.Hotel\_ID) AS num\_bookingsFROM hotels h JOIN

booking\_data b ON h.Hotel\_ID = b.Hotel\_ID

GROUP BY h.City\_Name, h.Hotel\_ID, h.Hotel\_Name

ORDER BY h.City\_Name, num\_bookings DESC;

	City_Name	Hotel_ID	Hotel_Name	num_bookings
▶	Ahmedabad	31	Ahmedabad Hotel 1	6
	Bengaluru	11	Taj West End	7
	Chennai	16	Taj Coromandel	5
	Hyderabad	26	Hyderabad Hotel 1	7
	Kolkata	21	The Oberoi Grand	11
	Mumbai	1	Taj Mahal Palace	6
	New Delhi	6	The Imperial	8

3) Feedback given by customers for hotels

SELECT C.Customer\_ID, B.Feedback, B.Booking\_ID,

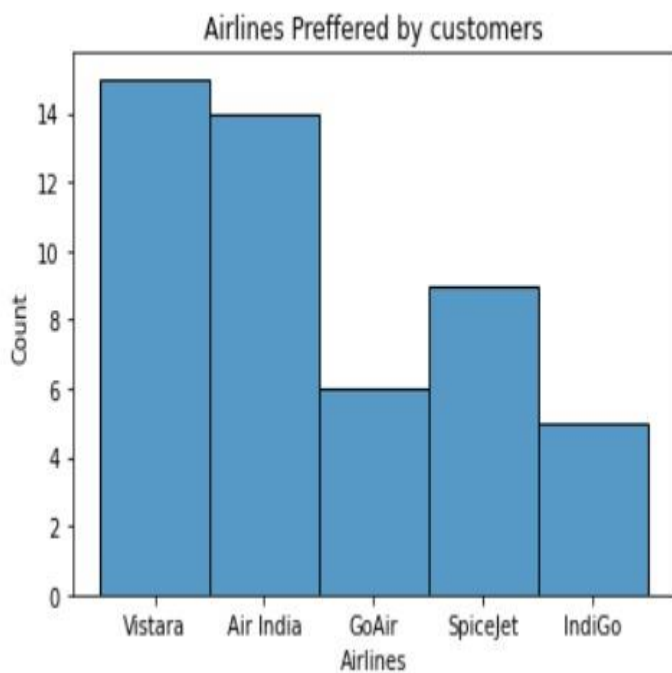
B.Hotel\_ID, H.Hotel\_NameFROM Customers C

JOIN booking\_data B ON C.Customer\_ID = B.Customer\_ID

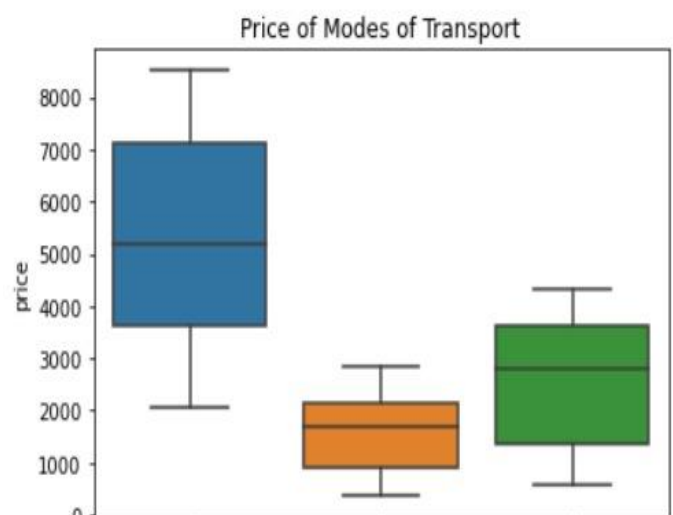
JOIN hotels H ON B.Hotel\_ID = H.Hotel\_ID WHERE B.Feedback IS NOT NULL;

Customer_ID	Feedback	Booking_ID	Hotel_ID	Hotel_Name
3	Good	42	1	Taj Mahal Palace
12	Excellent	40	1	Taj Mahal Palace
2	Excellent	36	1	Taj Mahal Palace
4	Excellent	35	1	Taj Mahal Palace
8	Good	14	1	Taj Mahal Palace
2	Average	10	1	Taj Mahal Palace
17	Average	49	6	The Imperial
18	Good	45	6	The Imperial
19	Good	27	6	The Imperial
10	Good	23	6	The Imperial
11	Excellent	17	6	The Imperial
18	Good	15	6	The Imperial
5	Good	8	6	The Imperial
11	Excellent	4	6	The Imperial
19	Good	34	11	Taj West End
15	Excellent	33	11	Taj West End
3	Good	32	11	Taj West End
16	Good	24	11	Taj West End
3	Excellent	20	11	Taj West End
15	Good	12	11	Taj West End
12	Good	6	11	Taj West End
4	Excellent	50	16	Taj Coromandel
4	Average	44	16	Taj Coromandel
12	Average	39	16	Taj Coromandel

### Streamlite Application Implementation:

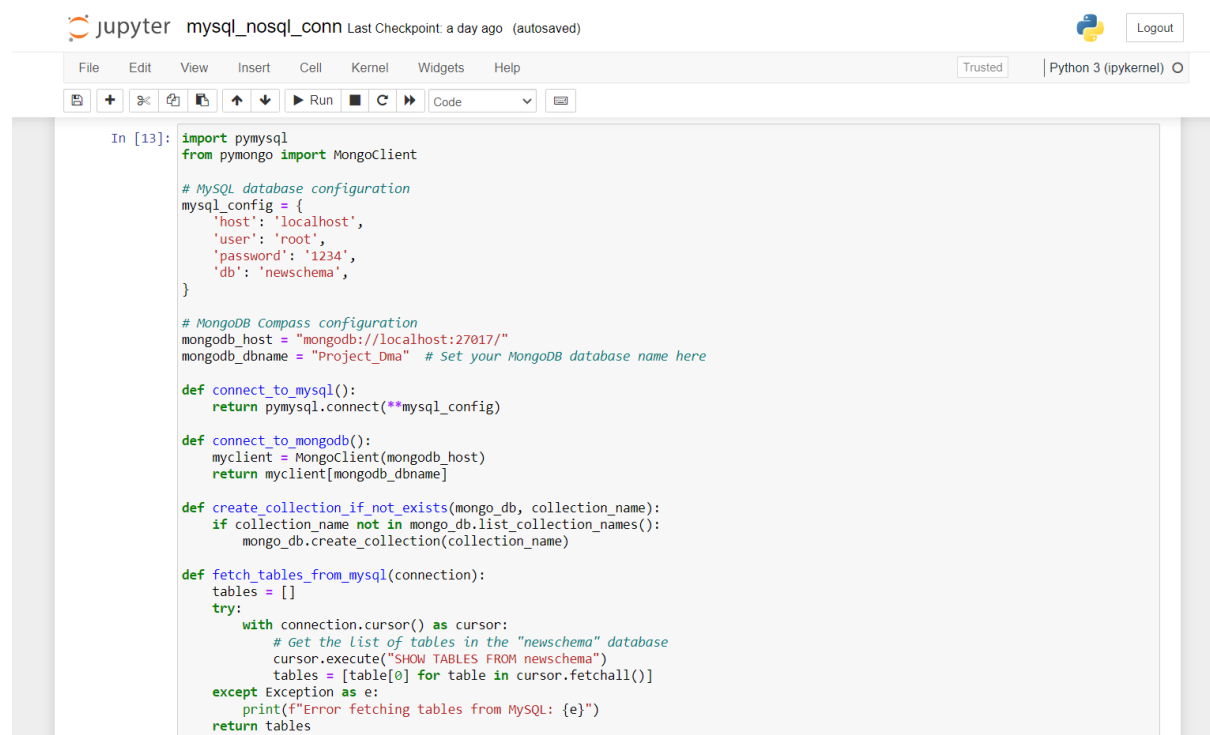


Transport\_Mode Preferred by customers



## NOSQL Implementation :

We connected our MYSQL database with MONGODB using Python.



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook title is 'mysql\_nosql\_conn' and it shows 'Last Checkpoint: a day ago (autosaved)'. The code is written in Python 3 (ipykernel). The code defines functions to connect to MySQL and MongoDB, create collections if they don't exist, and fetch tables from MySQL.

```
In [13]: import pymysql
from pymongo import MongoClient

# MySQL database configuration
mysql_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '1234',
    'db': 'newschema',
}

# MongoDB Compass configuration
mongodb_host = "mongodb://localhost:27017/"
mongodb_dbname = "Project_Dma" # Set your MongoDB database name here

def connect_to_mysql():
    return pymysql.connect(**mysql_config)

def connect_to_mongodb():
    myclient = MongoClient(mongodb_host)
    return myclient[mongodb_dbname]

def create_collection_if_not_exists(mongo_db, collection_name):
    if collection_name not in mongo_db.list_collection_names():
        mongo_db.create_collection(collection_name)

def fetch_tables_from_mysql(connection):
    tables = []
    try:
        with connection.cursor() as cursor:
            # Get the list of tables in the "newschema" database
            cursor.execute("SHOW TABLES FROM newschema")
            tables = [table[0] for table in cursor.fetchall()]
    except Exception as e:
        print(f"Error fetching tables from MySQL: {e}")
    return tables
```

We created collections in MongoDB which stored all our data.

The screenshot shows the MongoDB Compass interface for a local instance at localhost:27017. The 'Collections' tab is active, displaying a list of collections for the 'Project\_Dma' database. The collections listed are 'airways', 'booking\_data', 'bus', 'city', and 'customers'. Each collection entry shows its storage size (all 4.10 kB), the number of documents, the average document size, the number of indexes, and the total index size.

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
airways	4.10 kB	42	136.00 B	1	4.10 kB
booking_data	4.10 kB	50	141.00 B	1	4.10 kB
bus	4.10 kB	42	136.00 B	1	4.10 kB
city	4.10 kB	3	44.00 B	1	4.10 kB
customers	4.10 kB	20	100.00 B	1	4.10 kB

Customer travelling by airways.

The screenshot shows the MongoDB Compass interface with the 'Aggregations' tab selected. A pipeline is defined to filter customers who traveled by airways and group the results by destination. The pipeline consists of five stages: \$match, \$lookup, \$unwind, \$group, and \$project. The output shows a sample of 10 documents, grouped by destination (Hyderabad, Bengaluru, Mumbai).

```
1 [
2   {
3     $match: {
4       Transport_ID: 1, // Filter customers who traveled by airways
5     },
6   },
7   {
8     $lookup: {
9       from: "airways",
10      localField: "Transport_ID",
11      foreignField: "Transport_ID",
12      as: "airwaysInfo",
13    },
14  },
15  {
16    $unwind: "$airwaysInfo",
17  },
18  {
19    $group: {
20      _id: {
21        Customer_ID: "$Customer_ID",
22        Source: "$airwaysInfo.Source",
23        Destination: "$airwaysInfo.Destination",
24      },
25    }
26  }
27 ]
```

**PIPELINE OUTPUT**  
Sample of 10 documents

averagePrice	Customer_ID	Source	Destination
8504	17	New Delhi	Hyderabad
3878	7	Ahmedabad	Bengaluru
7762	11	Bengaluru	Mumbai

Count of modes of transportation preferred by customers.



Pipeline

\$group

\$project

Generate aggregation

Explain

Export

Run

More Options

COUNT OF M...

SAVE

CREATE NEW

EXPORT TO LANGUAGE

PREVIEW

STAGES

TEXT

```
1 [
2   {
3     $group: {
4       _id: "$Transport_ID",
5       count: { $sum: 1 },
6     },
7   },
8   {
9     $project: {
10      _id: 0,
11      Transport_ID: {
12        $cond: {
13          if: { $eq: ["$_id", 1] },
14          then: "Airways",
15          else: {
16            $cond: {
17              if: { $eq: ["$_id", 2] },
18              then: "Railways",
19              else: "Bus",
20            },
21          },
22        },
23      },
24      count: "$count",
25    },
26  ],
```

PIPELINE OUTPUT

Sample of 3 documents

OUTPUT OPTIONS

Transport\_ID: "Bus"

count: 15

Transport\_ID: "Airways"

count: 17

Transport\_ID: "Railways"

count: 18

### Summary & Recommendation :

"Our Travel Advisory Management System underwent extensive development, actively enhancing the transition to MongoDB with clear visual representations. Notably, we prioritized database cleanliness, refining MongoDB collections for streamlined, efficient NoSQL implementation.

In tandem, we proactively refined EER and UML diagrams, offering comprehensive insights into data structures. Our active efforts concentrated on improving data consistency between MySQL and MongoDB, bolstering reliability.

To further elevate the system, a focus on continuous improvements in clean database practices and optimal NoSQL implementations remains a primary area for ongoing enhancements. These active refinements ensure a more efficient and user-centric platform for travelers and administrators alike."