IEOR215 Project Report (Group 3): Space Travel Database

Boya Shao (3039713094), Jialin Wang (3039679866),
Zilan Luo(3039670051), Xuanyang Wu (3034617548),
Yuejia Li (3039671572), Junyao Lu (3039671325),
Zhiyi Wang (3039713250), Binglan Lin (3039671520)

December 14, 2023

Contents

1 Introduction

In our IEOR 215 project, titled "Space Travel Database," we embark on a unique journey to design a sophisticated database system aimed at visualizing and facilitating the management of space travel operations. This endeavor addresses the growing need for robust data management solutions in the context of the expanding commercial spaceflight industry. As space travel transitions from science fiction to reality, the demand for robust and innovative data management systems becomes paramount. Our project simulates a scenario where space travel is as commonplace as air travel today, necessitating sophisticated database systems to manage complex operations and customer interactions. By developing a forward-looking database solution, we aim to contribute to the advancement and efficiency of the rapidly evolving space tourism industry.

The primary objective of this project is to design a comprehensive database system that can efficiently handle the intricacies of space travel operations. This includes managing bookings, tracking flights, coordinating crew assignments, and ensuring customer satisfaction. Through this project, we aim to demonstrate not only our technical prowess in database design but also our capacity to envision and support the future of space travel.

Moreover, we make the following assumptions about the fictional database about the space travel system:

1. Randomly Generated Data: To simulate a realistic space travel environment, all data in our database, including customer profiles, flight details, and booking transactions, are randomly generated. This approach allows us to test the robustness of our database under diverse and unpredictable scenarios.

2. Booking Agent-Customer Interplay: Our model posits that all ticket purchases for space travel must be conducted through booking agents, who may also be customers. All individuals who have ordered tickets through the space travel system are regarded as members.

3. Distinct Staff Operational Roles: We assume that cabin crew and maintenance personnel have exclusive and non-interchangeable roles. Additionally, there are various other positions within the spaceline company staff, contributing to the complexity and efficiency of space travel operations.

In conclusion, our project "Space Travel Database" is a step towards imagining the future of space tourism and its associated data management challenges. As we delve into this project, we hope to uncover insights and strategies that could one day be applied in the real world of commercial space travel.

2 EER Diagram

The simplified EER diagram is shown in Figure 1.

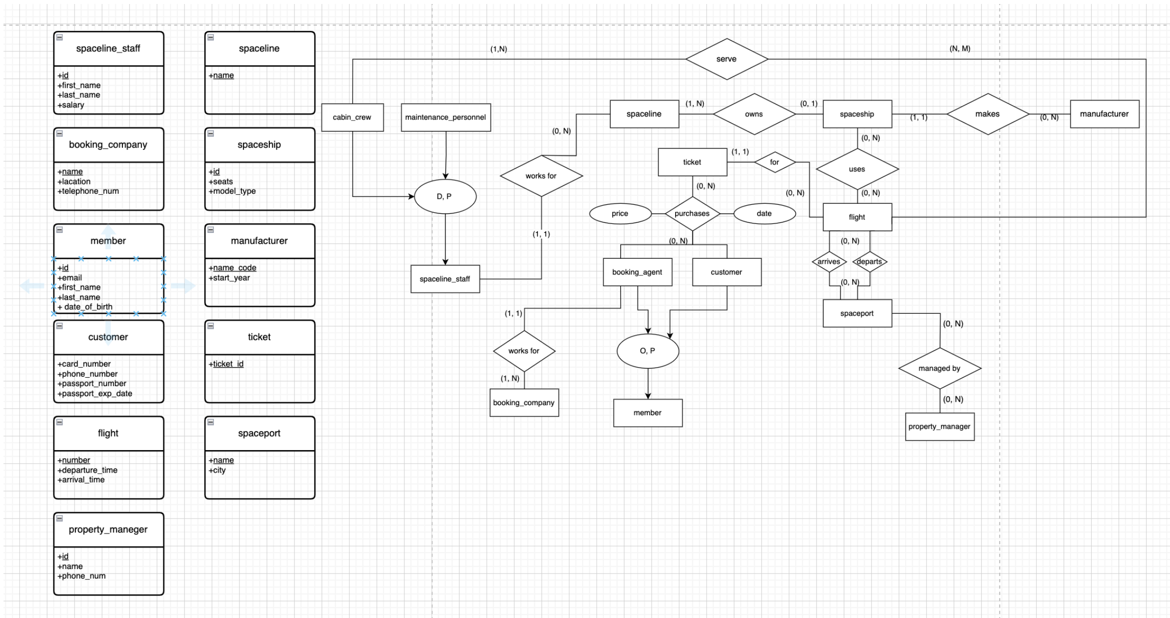

Figure 1: EER Diagram for the Project

This schema models a space travel system, including spacelines, spaceships, manufacturers, flights, staff, customers, bookings, and spaceports, among other entities. The relationships between entities are defined through keys, linking the tables and enabling the representation of complex interactions within the space travel domain. The detailed descriptions of the tables are expressed in the following Table 2.

3 Relational Design (Schema)

Based on the EER diagram, we design the following schema:

1. Spaceline(name)
2. Spaceship(id, seats, model_type, spaceline_name[1], manufacturer_name_code[3])
3. Manufacturer(name_code, start_year)
4. Ticket(ticket_id, flight_number[6])
5. Spaceline_staff(id, first_name, last_name, salary, spaceline_name[1])

5a. Cabin_crew(id[5])

5b. Maintenance_personnel(id[5])

6. Flight(number, departure_time, arrival_time)
7. Member(id, email, first_name, last_name, date_of_birth)

7a. Booking_agent(id[7], booking_company_name[8])

7b. Customer(id[7], card_number, phone_number, passport_number, passport_exp_date)

8. Booking_company(name, location, telephone_num)
9. Spaceport(name, city)
10. Property_manager(id, name, phone_num)
11. Purchases(ticket_id[4], booking_agent_id[7a], customer_id[7b], price, date)
12. Uses(spaceship_id[2], flight_number[6])
13. Arrives(flight_number[6], spaceport_name[9])
14. Departs(flight_number[6], spaceport_name[9])

Table Name	Description
Spaceline	Stores information about spaceline company.
Spaceship	Information about spaceships, which are used to carry passengers between planets and spaceports.
Manufacturer	Contains details about manufacturers constructing the spaceships.
Ticket	Records flight ticket information.
Spaceline_staff	Stores information about spaceline staff working for the spaceline companies.
Cabin_crew	Represents cabin crew members, who serve on the flights.
Maintenance_personnel	Includes information about maintenance personnel who work behind the scenes.
Flight	Contains flight details, which is a single travel.
Member	Stores information about members, who order tickets through the space travel system.
Booking_agent	Represents booking agents who book tickets for customers.
Customer	Contains customer details who buy tickets directly or through a booking agent.
Booking_company	Records information about booking companies booking agents work for.
Spaceport	Contains details about spaceports where spaceships arrive at and depart from.
Property_manager	Represents property managers who manage and operate the specific spaceport.
Purchases	Stores information about ticket purchases.
Uses	Represents the usage of spaceships in flights.
Arrives	Records arrival information for flights at spaceports.
Departs	Records departure information for flights at spaceports.
Managed	Indicates the management of spaceports by property managers.
Serve	Represents the service of cabin crew members on flights.

Table 1: Table Descriptions

15. Managed(spaceport_name[9], property_manager_id[10])
16. Serve(cabin_crew_id[5a], flight_number[6])

4 Tables and SQL

Creating tables of the space travel system using MySQL, we present the following MySQL Relationship View in Figure 2. Additionally, example SQL codes to create tables and retrieve information are shown for visualization below.

4.1 Example SQL for creating tables

```
CREATE TABLE `spaceline` (
  `name` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `spaceship` (
  `id` varchar(255) NOT NULL,
  `seats` int(11) DEFAULT NULL,
  `model_type` varchar(255) NOT NULL,
  `spaceline_name` varchar(255) NOT NULL,
  `manufacturer_name_code` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `flight` (
  `number` varchar(255) NOT NULL,
  `departure_time` time NOT NULL,
  `arrival_time` time NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `uses` (
  `spaceship_id` varchar(255) NOT NULL,
```

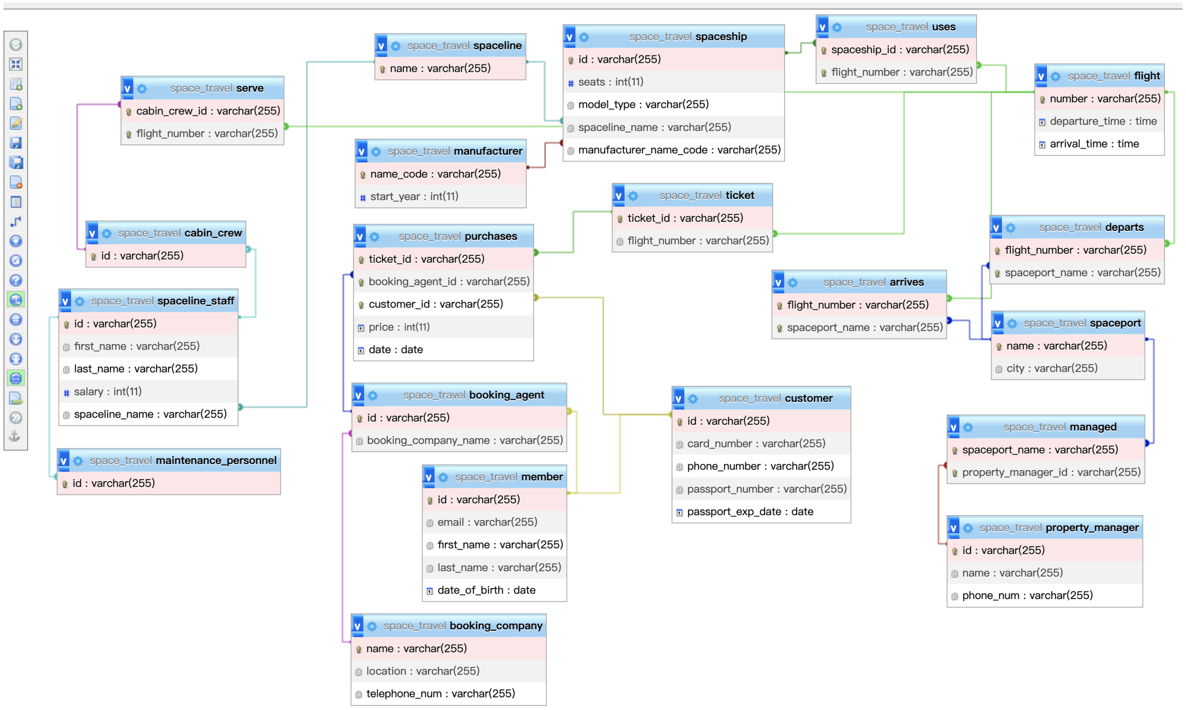

Figure 2: MySQL Relationship View (phpMyAdmin)

```
  `flight_number` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `spaceline`
  ADD PRIMARY KEY (`name`);

ALTER TABLE `spaceship`
  ADD PRIMARY KEY (`id`),
  ADD KEY `spaceline_name` (`spaceline_name`),
  ADD KEY `manufacturer_name_code` (`manufacturer_name_code`);

ALTER TABLE `flight`
  ADD PRIMARY KEY (`number`);

ALTER TABLE `uses`
  ADD PRIMARY KEY (`spaceship_id`,`flight_number`),
  ADD KEY `spaceship_id` (`spaceship_id`),
  ADD KEY `flight_number` (`flight_number`);

ALTER TABLE `spaceship`
  ADD CONSTRAINT `spaceship_ibfk_1` FOREIGN KEY (`spaceline_name`) REFERENCES `
    spaceline` (`name`),
  ADD CONSTRAINT `spaceship_ibfk_2` FOREIGN KEY (`manufacturer_name_code`) REFERENCES
    `manufacturer` (`name_code`);

ALTER TABLE `uses`
  ADD CONSTRAINT `uses_ibfk_1` FOREIGN KEY (`spaceship_id`) REFERENCES `spaceship` (`
    id`),
  ADD CONSTRAINT `uses_ibfk_2` FOREIGN KEY (`flight_number`) REFERENCES `flight` (`
    number`);
```

4.2 Example SQL to retrieve information

4.2.1 Example 1

One of the customers wants to view all his or her booked flights. Eg: Emma Tsai would like to view all the flights she booked before.

```
SELECT DISTINCT T.flight_number
FROM ticket AS T, purchases AS P, member AS M
WHERE T.ticket_id = P.ticket_id AND P.customer_id = M.id AND M.first_name = 'Emma'
 AND M.last_name = 'Tsai';
```

flight_number
CH02
UN02

Figure 3: Result for Example 1

4.2.2 Example 2

The top 10 performance commissions (10 percent of the total ticket price) earned by booking agents.

```
SELECT B.id AS BookingAgentID, B.booking_company_name as BookingCompanyName, SUM(p
.price * 0.1) AS CommissionEarned
FROM Purchases p, Booking_Agent B
WHERE p.booking_agent_id = B.id
GROUP BY B.id
ORDER BY CommissionEarned DESC
LIMIT 10;
```

BookingAgentID	BookingCompanyName	CommissionEarned 1
12834	Hope	1568.4
37252411	Expedia	1424.8
16354455	Expedia	1049.9
72633495	Trip	990.8
712974649	Booking	947.4
3524162	Flyover	777.7
82779772	Trip	766.4
26424174	Expedia	614.2
725611809	Booking	605.0
19283	Hope	599.0

Figure 4: Result for Example 2

4.2.3 Example 3

Show the top 3 customers who paid the highest total money for flight tickets in the past.

```sql
WITH temp_table1 AS
(SELECT P.customer_id AS CID, SUM(P.price) AS total_money
FROM purchases AS P
GROUP BY CID
ORDER BY total_money DESC
LIMIT 3)
SELECT M.first_name, M.last_name, temp_table1.total_money
FROM temp_table1, member AS M
WHERE M.id = temp_table1.CID;
```

first_name	last_name	total_money
Heidi	Lin	10888
Emma	Tsai	8899
Tony	Garcia	8555

Figure 5: Result for Example 3

4.2.4 Example 4

Show the top 5 customers who traveled the most frequently in the past year (but more than once).

```sql
SELECT P.customer_id, M.first_name, M.last_name,  COUNT(P.ticket_id) as
traveling_times
FROM Member M, Purchases P
WHERE M.id = P.customer_id AND P.date >= '2023-01-01'
GROUP BY P.customer_id
HAVING COUNT(P.ticket_id) > 1
ORDER BY traveling_times DESC
LIMIT 5;
```

customer_id	first_name	last_name	traveling_times
566684	Emma	Tsai	4
1528484	Selena	Anderson	3
1927283	Isabella	Martinez	3
364572	Ju–won	Kim	3
5407569	Min–jun	Choi	3

Figure 6: Result for Example 4

5 Potential Analysis

5.1 Normalization Analysis

All the relations in our schema are in 1NF, in which all attributes are single-valued.

5.1.1 2NF: Purchases

11. Purchases(ticket_id[4], booking_agent_id[7a], customer_id[7b], price, date)

The non-prime attributes price and date is not fully dependent on the primary key. {ticket_id → price, date}

We can modify Purchases to 2NF:

Sale(ticket_id[4], price, date)

Purchases(ticket_id[4], booking_agent_id[7a], customer_id[7b])

After normalizing Purchases to 2NF, all relations have satisfied 1NF, 2NF, 3NF, and BCNM.

5.2 Data Analysis

5.2.1 Total Money Paid and Age

We are trying to investigate the relationship between age and the total money paid for booking flight tickets. The SQL code is shown below. Then we draw the scatter plot concerning age and total money paid in Figure 7. We find that, generally, elder people tend to pay more total money for booking tickets than younger people. However, more data points are clustered between ages 10 and 40, which means that young people are the main force for purchasing tickets for space travel.

```
WITH temp_table1 AS
(SELECT P.customer_id AS CID, SUM(P.price) AS total_money FROM purchases AS P
GROUP BY CID
ORDER BY total_money DESC)
SELECT M.first_name, M.last_name, DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(), M.
date_of_birth)), '%Y') + 0 AS age, temp_table1.total_money
FROM temp_table1, member AS M
WHERE M.id = temp_table1.CID;
```

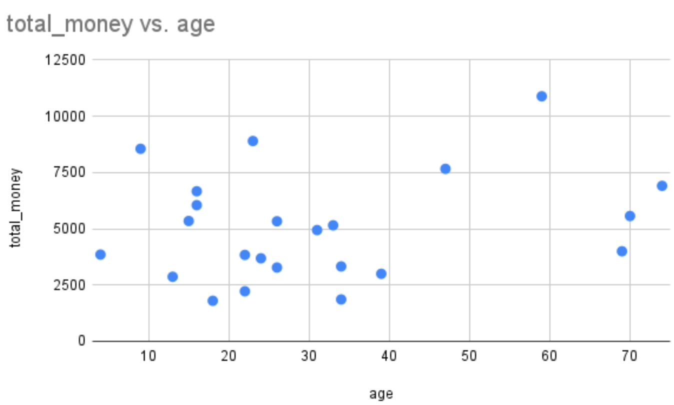

Figure 7: Total Money and Age

5.2.2 Traveling Frequency and Age

We are investigating the relationship between age and the number of flights customers have taken in the past year. The SQL code is shown below. Then we draw the scatter plot concerning age and traveling frequency in Figure 8. We find that younger people tend to travel more frequently and people between 10 and 40 years old are the people who travel the most frequently. People older than 40 seldom take a flight. At this point, we can regard people between 10 and 40 years old as our main customers. For future analysis, we could stratify this range into smaller groups of people and set more accurate targets.

```
WITH temp_table1 AS
(SELECT P.customer_id AS CID, COUNT(P.ticket_id) AS traveling_frequency FROM
purchases AS P
WHERE P.date >= '2023-01-01'
GROUP BY CID
ORDER BY traveling_frequency DESC)
SELECT M.first_name, M.last_name, DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(), M.
date_of_birth)), '%Y') + 0 AS age, temp_table1.traveling_frequency
FROM temp_table1, member AS M
```

```
    WHERE M.id = temp_table1.CID;
```

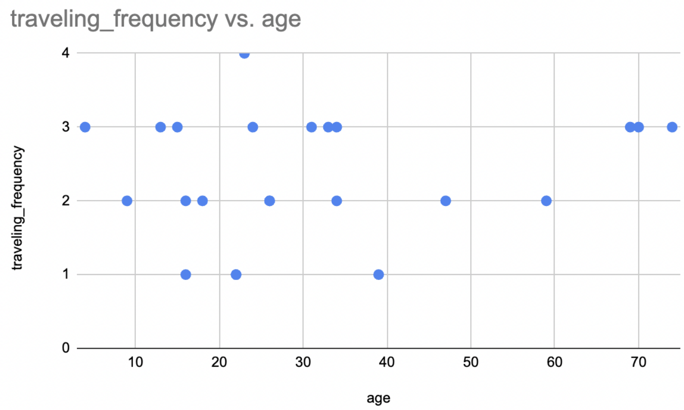

Figure 8: Traveling Frequency and Age

5.2.3 Booking Agent Commissions and Location

We are investigating the relationship between the booking agent commissions and their location. The SQL code is shown below. Then we draw the bar chart concerning the booking agent commissions and their location in Figure 9. From the chart, we can see that booking agents in Shanghai earn the highest commission, and Beijing is the second one while booking agents in New York earn the least. We can infer that booking agents in Shanghai are the most popular and their attributes can be further discussed for boosting sales of agents in other cities.

```
    SELECT BC.location, SUM(p.price * 0.1) AS CommissionEarned
    FROM Purchases p, Booking_Agent B, Booking_company BC
    WHERE p.booking_agent_id = B.id and B.booking_company_name = BC.name
    GROUP BY BC.location
    ORDER BY CommissionEarned DESC;
```

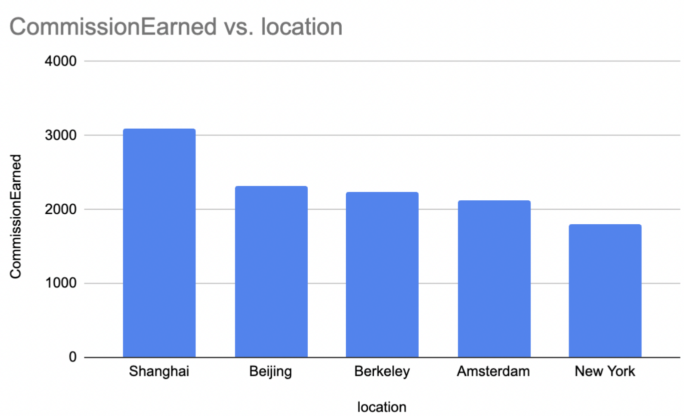

Figure 9: Booking Agent Commissions and Location

6 Conclusion and Discussion

In this project, our primary goal is to construct a comprehensive fictional database about the space travel system, utilize EER modeling and relational schema to visualize space travel operation management, and also try different queries using MYSQL to extract useful information from our database.

The structure of our database design is founded upon an elaborate EER diagram, which shows a clear relationship between various entities, such as spaceline companies, spaceship tickets, booking agents, and customers. Our database records all the information about which customer has purchased which ticket, which spaceship arrives and lands on which spaceport, etc.

After building the complete database, we could use SQL queries to extract multifaceted information, which can provide us with a more convenient way to invoke the specified information we need. For example, we select all the flight numbers of Emma Tsai's booked flights to view all the flights she had taken. We can also rank the booking agents by their total commissions to evaluate their performance and show the top 3 customers who paid the highest total money for their flight tickets to formulate a more reasonable sales strategy for our customers. In this way, we can not only have a comprehensive knowledge of the space travel industry but also have the approach to enhance the management of different entities in this field.

Looking ahead, the practical implications of our database in future real-life scenarios are promising. Implementation of such a system could revolutionize space travel management by enhancing customer experience, optimizing resource utilization, and improving operations. Real-time monitoring of flights, predictive maintenance of spaceships, and data-driven decision-making can all be realized by this database system. A robust database system could serve as a crucial tool for ensuring safety, compliance with regulations, and standardization across the sector. It is also a comprehensive repository for regulatory bodies, researchers, and industry stakeholders, aiding in the advancement and governance of space exploration.

In conclusion, our project provides invaluable experience in designing, implementing, and querying a complex relational database system for space travel management. The potential real-world applications of such a system are vast. As we envision the future, the insights gained from this project are fundamental in shaping and optimizing the landscape of space travel.