INF2003 Database Systems

Group Project 1

Group ID: 31

Student ID	Student Name	Contribution (%)	Contribution (Descriptions)
2200564	Jordan Gho Jun Sheng	20	Proposal, Complex function, Report, Video
2200548	Oh Jiang Yi	20	Proposal, CR function, GUI, Report, Powerpoint
2201062	Tan Jeng Siang Joel	20	Proposal, Complex function, Report, Powerpoint
2201066	Yeo Chin Seng Samuel	20	Proposal, UD Functions, GUI, Report, Powerpoint
2203365	Yuan Yucheng	20	Proposal, Complex function, Report, Video

<2024-10-12, 22:00>

Submitted as part of the requirement for INF2003 Database Systems

Table of Content

1. Introduction

1.1. Project Overview

Our Air Passenger Arrival Data Management System addresses critical challenges in tourist management and economic planning by consolidating fragmented air passenger data into a unified, comprehensive database. This innovative system enables real-time analysis of arrival patterns, integrating information on passengers, countries of origin, length of stay, and airlines. By overcoming issues of data fragmentation and ineffective reporting, it provides tourism boards, aviation authorities, and destination managers with powerful tools for data-driven decision-making. The system's capabilities extend to enhanced resource allocation, targeted marketing campaigns, improved tourist experiences, and the promotion of sustainable tourism practices. Ultimately, this solution transforms scattered data into actionable insights, fostering more efficient and effective management of tourism resources and potentially increasing tourism revenue.

1.2. Objectives

- Store and manage the arrival data from different sources (air, tourism markets, etc.).
- To analyse visitor trends by region, demographic factors, and length of stay.
- To support reporting and visualisation of arrival data, aiding in tourism planning and resource management.

2. System Architecture Design and Requirement

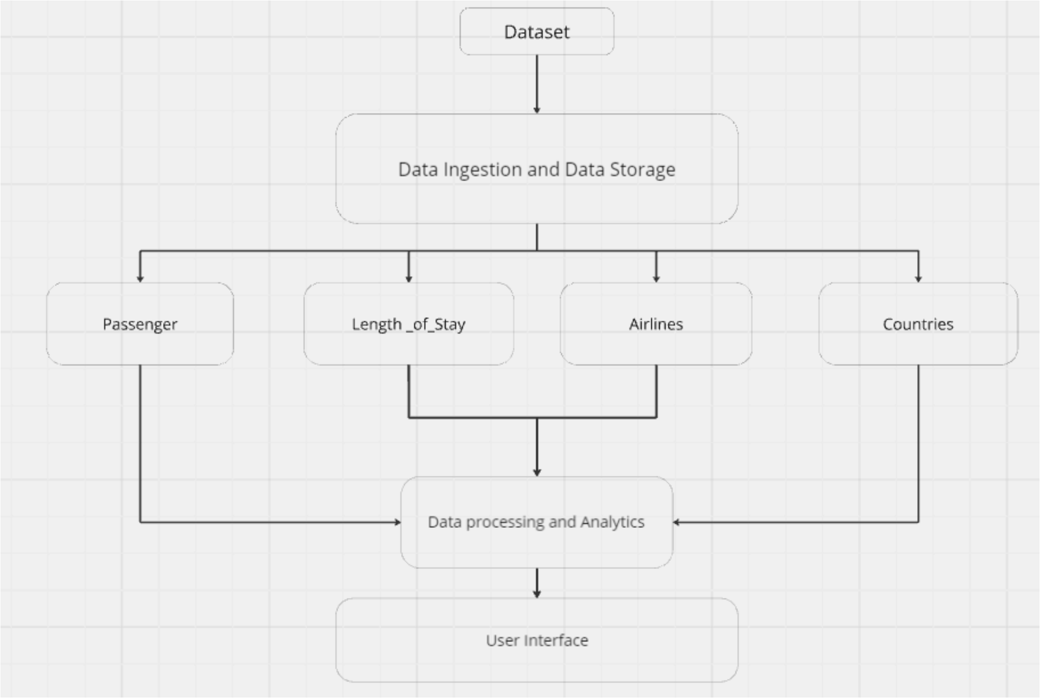


Fig 2.1 Overall System Architecture

2.1. External Data Source

- Our application requires some user data which is not able to be found anywhere, hence we generated those data.
- Data exported as CSV files for easy handling and compatibility

2.2. Data Ingestion

- Cleaned data by removing inconsistencies and handling missing values
- Preprocessed data to standardise formats and prepare for analysis
- Developed automated process to import cleaned data into MariaDB

2.3. Data Storage

- Created 4 main tables in MariaDB:
  - Passenger: Stores visitor details (e.g. name, age, gender, airlines)
  - Airlines: Contains airline information and airline counts
  - Countries: Holds data on visitor numbers by country
  - Length_of_Stay: Tracks duration of visits
- Implemented primary keys for data integrity and foreign keys for relational structure

2.4. Data Processing & Analysis

- Implemented CRUD (Create, Read, Update, Delete) functions for data management
- Utilised Python with pandas for data manipulation and analysis
- Employed matplotlib for creating visualisations and charts
- Performed analyses on:
  - Analyse Airline Popularity
  - Analyse Tourism Duration
  - Analyse Airline Trend

2.5. User Interface

- Developed GUI using Python's tkinter library
- Features include:
  - Interactive data visualisations
  - Report generation capabilities
  - User-friendly data query interface
  - Navigation controls for different analyses

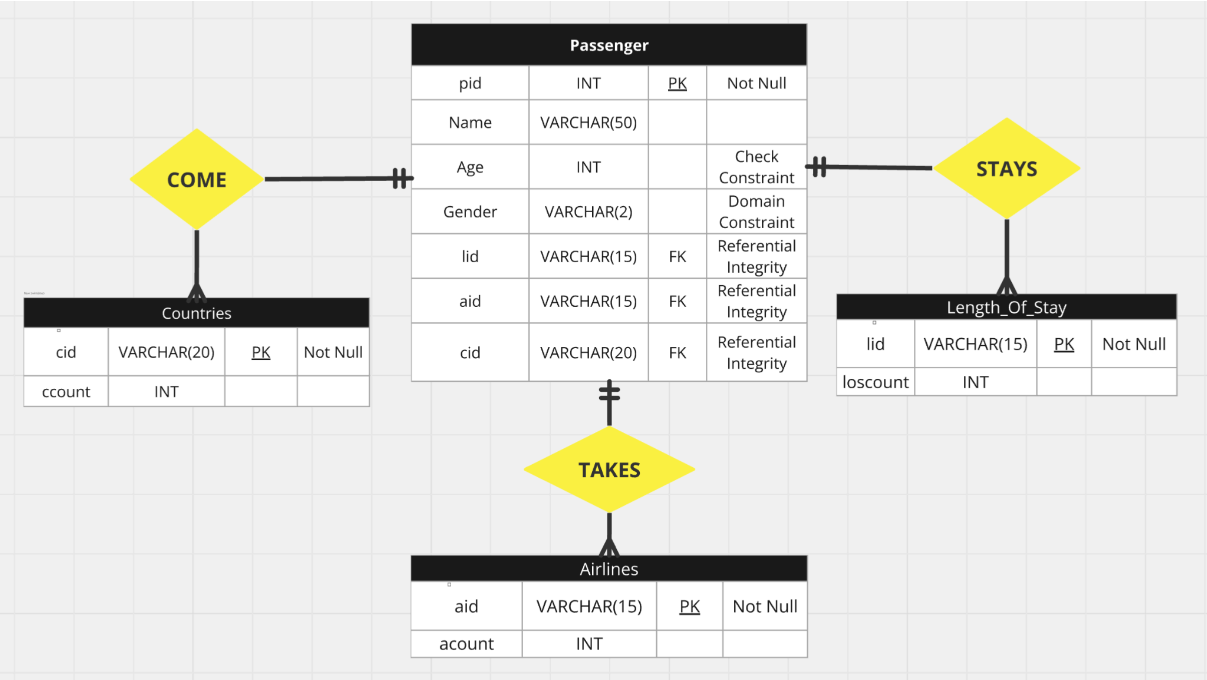
Entity	Description	Attributes
Passenger	Stores essential details about	pid (INT PRIMARY KEY),

	visitors travelling to Singapore	Name (VARCHAR(50)), Age (INT), Gender (VARCHAR(2)) lid (VARCHAR(15), FOREIGN KEY), aid (VARCHAR(15), FOREIGN KEY), cid(VARCHAR(20), FOREIGN KEY)
Countries	Store information about visitor numbers by country.	cid (VARCHAR(20) PRIMARY KEY), ccount (INT)
Airlines	Store information about visitors numbers by airline	aid (VARCHAR(15), PRIMARY KEY) acount (INT)
Length_of_Stay	Store information about how long visitors stay in Singapore	lid (VARCHAR(15), PRIMARY KEY), loscount (INT)

Table 1: Entity-Attribute Description for Arrival Visitor Database

3. Implementation

3.1. ER Diagram



Passenger

pid	INT	PK	Not Null
Name	VARCHAR(50)		
Age	INT		Check Constraint
Gender	VARCHAR(2)		Domain Constraint
lid	VARCHAR(15)	FK	Referential Integrity
aid	VARCHAR(15)	FK	Referential Integrity
cid	VARCHAR(20)	FK	Referential Integrity

Countries

cid	VARCHAR(20)	PK	Not Null
ccount	INT		

Length_Of_Stay

lid	VARCHAR(15)	PK	Not Null
loscount	INT		

Airlines

aid	VARCHAR(15)	PK	Not Null
acount	INT		

Fig 3.1: ER Diagram for Arrival Visitor Database

3.2. Functionalities

Functionalities	Description
CRUD functions (See Appendix B)	
Create Passenger by Name, Age, Gender, Length of Stay, Airline, Country	This functionality will allow the system to add passenger arrival data.
Read Passenger Arrival by Name, Age, Gender, Length of Stay, Airline, Country	This functionality will allow the users to view the amount of passengers arriving by all, pid and name.
Update Passenger by Name, Age, Gender, Length of Stay, Airline, Country	This functionality will allow the system to update passenger arrival data via pid.
Delete Passenger by Name, Age, Gender, Length of Stay, Airline, Country	This functionality will allow the system to remove passenger arrival data via pid.
Complex Functions (See Appendix C)	
Analyse Airline Popularity	Compares passenger counts across airlines entering Singapore to determine the most popular carriers. Generates a ranking and visualisation of airline popularity based on incoming visitor data.
Analyse Tourism Duration	Examines visitor stay durations in Singapore, categorising lengths into ranges (e.g., 1-3 days, 4-6 days) and identifying the most common duration. Provides insights for tailoring tourism strategies based on typical visit lengths.
Analyse Airline Trend	Analyses the relationship between visitors' countries of origin and airlines for Singapore tourism. Identifies trends in travel routes and airline preferences through comparative data analysis and charting.

GUI Function (See Appendix D)	
Tkinter	Provides a user interface to interact with the various functions implemented such that the user is able to perform analysis and CRUD functions. These functions include: Analysis of Airline popularity, Analysis of Tourism Duration, and Analysis of Airline Trend.

3.3. Constraints

Constraints (See Appendix E)	Justification
Primary Key Constraint	Ensure each rows in a table is uniquely identifiable e.g. pid
Not Null Constraint	Ensure that column cannot have NULL values especially primary keys e.g. pid, enforces that certain fields must always have a value
Check Constraint	Implement CHECK constraints for fields like Age to enforce domain integrity by limiting the values that can be placed in a column or set of columns
Foreign Key Constraint	Establish a relationship between two tables, restricting actions that would invalidate the relationship
Referential Integrity	Foreign key constraints ensure data consistency across tables.
Domain Constraint	Ensure that an attribute’s value must come from a predefined set or domain e.g. Gender attribute restricting to (‘M’, ‘F’)

4. Discussions

Scalability:

We can add, remove and update passenger details while ensuring that all tables in the database are updated accordingly. As the volume of data increases, especially with continuous inflows of passenger arrival information, the system will need to be able to handle higher loads efficiently. Currently, the system works well with the available dataset. However, further optimization would be required to manage significantly larger datasets, potentially involving sharding or indexing strategies in the database to ensure query performance remains high.

Reliability:

The system ensures data consistency and integrity by implementing CRUD operations with appropriate primary and foreign key relationships. Data accuracy is maintained through thorough data cleaning and preprocessing before ingestion into the system. Additionally, validation checks are performed during data entry to ensure the correctness of the input values. Given the dependency on external datasets, periodic data quality checks need to be performed to ensure that the incoming data is accurate and up-to-date, minimising issues related to outdated or incomplete data.

Security:

The current system implements basic security measures, including user input validation to prevent SQL injection during data entry and updates. However, more advanced security measures should be considered for real-world deployment. These could include user authentication, role-based access control (RBAC) to ensure only authorised users can modify or view certain data, and encryption of sensitive information like passenger details. Additionally, regular backup processes and data recovery mechanisms should be put in place to prevent data loss due to unexpected failures.

Limitations and Future Improvements:

The current system is primarily focused on Southeast Asian countries, which limits its applicability to a global context. Additionally, the airlines included in the database are restricted to those with high popularity and frequency of flights from South East Asian countries to Singapore. While this approach allows for a more targeted analysis of regional travel patterns, it may not provide a comprehensive view of international travel trends.

This limitation in geographical scope and airline selection presents a significant opportunity for future enhancement. Expanding the system to include a broader range of countries and airlines would provide a more holistic view of global travel patterns and allow for more extensive comparative analyses. Future iterations of the system could incorporate data from all continents, major international airlines, and a wider variety of flight routes. This expansion would not only increase the system's utility for global travel analysis but also enhance its potential for identifying broader trends in tourism and air travel. Furthermore, incorporating data from diverse geographical regions and airlines could lead to more robust predictive models and insights, potentially benefiting stakeholders in the global tourism and aviation industries.

5. Reflection & takeaways

This relational database project has been both challenging and incredibly rewarding. Together, we've navigated the complexities of managing large datasets and developed a system that adapts to real-world constraints. This collaborative effort has not only sharpened our technical skills but also given us valuable insights into the practical applications of database management in industries like tourism and aviation.

One of the most significant challenges we faced was ensuring the accuracy, consistency, and proper integration of data within a relational database. Learning to implement efficient data handling strategies—from ingestion to analysis—has deepened our understanding of database management systems. Moreover, we now appreciate the intricacies of designing a system that meets current needs while remaining scalable for future growth. This foresight in system design is something we now recognize as essential in any tech-related project.

Additionally, working on this project emphasized the importance of scalability. As systems grow, optimizing performance, ensuring security, and maintaining data integrity become crucial. Through this experience, we have gained not only technical skills in database implementation and manipulation but also a practical perspective on the impact of these systems in industries like tourism and aviation.

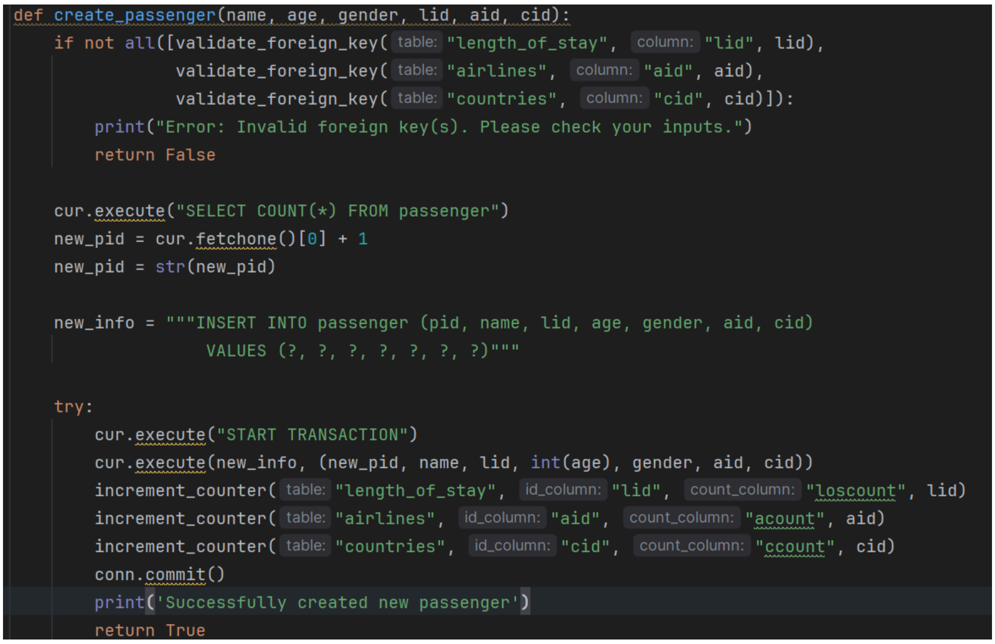
Reflecting on our journey, we realize that the skills we've developed go beyond technical proficiency. We've enhanced our problem-solving abilities, strengthened our communication, and learned how to leverage each team member's strengths to overcome challenges. These soft skills, combined with our technical knowledge, have prepared us well for future careers and academic pursuits.

As we conclude this project, we take pride in what we've accomplished as a team. We've not only built a functional database system but also gained invaluable experience in project management, teamwork, and problem-solving. Together, these skills form a solid foundation for our future endeavors.

Appendix A

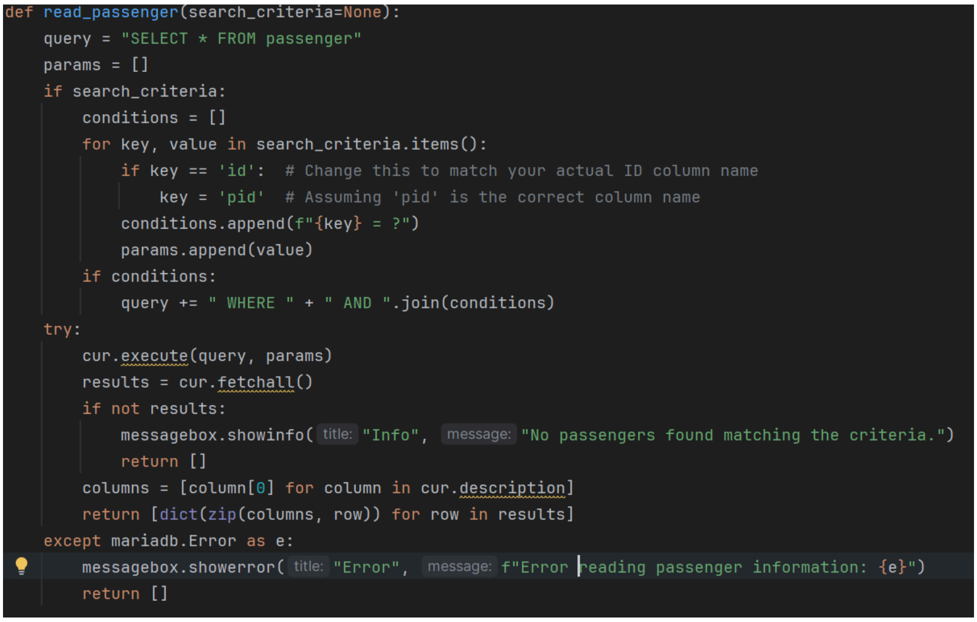
Download link for source code: https://github.com/wheks/INF2003-Project

Appendix B



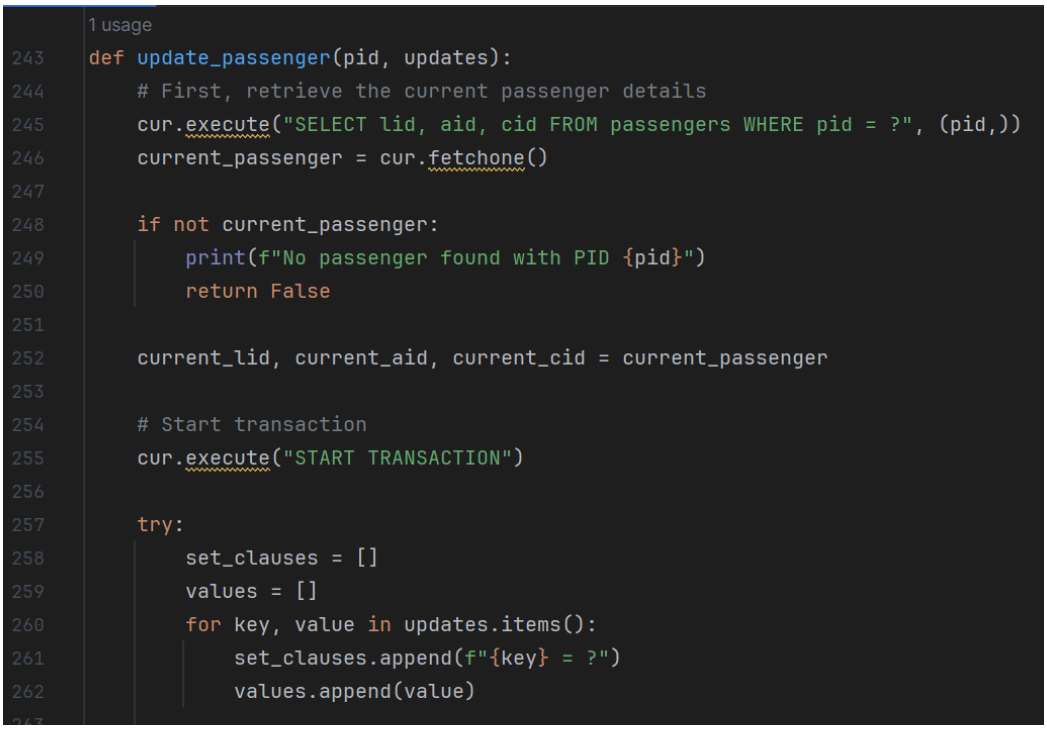
```python
def create_passenger(name, age, gender, lid, aid, cid):
    if not all([validate_foreign_key( table: "length_of_stay",  column: "lid", lid),
                validate_foreign_key( table: "airlines",  column: "aid", aid),
                validate_foreign_key( table: "countries",  column: "cid", cid)]):
        print("Error: Invalid foreign key(s). Please check your inputs.")
        return False

    cur.execute("SELECT COUNT(*) FROM passenger")
    new_pid = cur.fetchone()[0] + 1
    new_pid = str(new_pid)

    new_info = """INSERT INTO passenger (pid, name, lid, age, gender, aid, cid)
                VALUES (?, ?, ?, ?, ?, ?, ?)"""

    try:
        cur.execute("START TRANSACTION")
        cur.execute(new_info, (new_pid, name, lid, int(age), gender, aid, cid))
        increment_counter( table: "length_of_stay",  id_column: "lid",  count_column: "loscount", lid)
        increment_counter( table: "airlines",  id_column: "aid",  count_column: "acount", aid)
        increment_counter( table: "countries",  id_column: "cid",  count_column: "ccount", cid)
        conn.commit()
        print('Successfully created new passenger')
        return True
```

Fig B1: Create Function



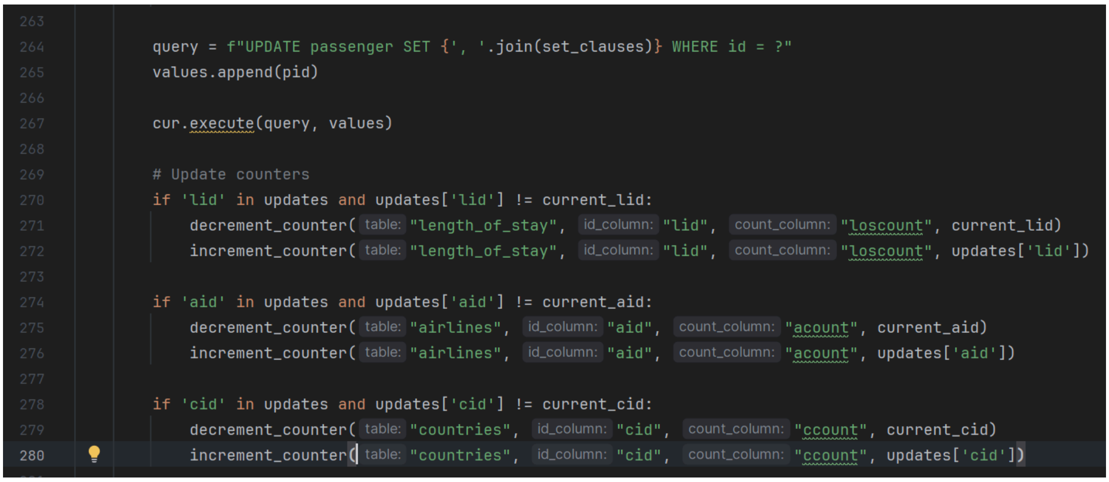
```python
def read_passenger(search_criteria=None):
    query = "SELECT * FROM passenger"
    params = []
    if search_criteria:
        conditions = []
        for key, value in search_criteria.items():
            if key == 'id':  # Change this to match your actual ID column name
                key = 'pid'  # Assuming 'pid' is the correct column name
            conditions.append(f"{key} = ?")
            params.append(value)
        if conditions:
            query += " WHERE " + " AND ".join(conditions)
    try:
        cur.execute(query, params)
        results = cur.fetchall()
        if not results:
            messagebox.showinfo( title: "Info",  message: "No passengers found matching the criteria.")
            return []
        columns = [column[0] for column in cur.description]
        return [dict(zip(columns, row)) for row in results]
    except mariadb.Error as e:
        messagebox.showerror( title: "Error",  message: f"Error reading passenger information: {e}")
        return []
```

Fig B2: Read Function



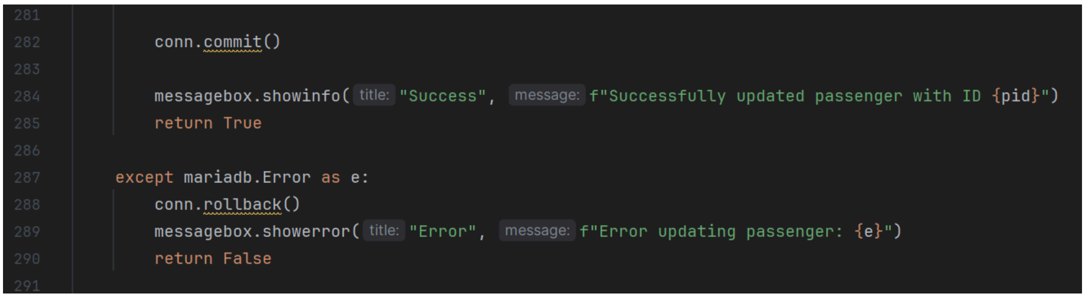
```
1 usage
def update_passenger(pid, updates):
    # First, retrieve the current passenger details
    cur.execute("SELECT lid, aid, cid FROM passengers WHERE pid = ?", (pid,))
    current_passenger = cur.fetchone()

    if not current_passenger:
        print(f"No passenger found with PID {pid}")
        return False

    current_lid, current_aid, current_cid = current_passenger

    # Start transaction
    cur.execute("START TRANSACTION")

    try:
        set_clauses = []
        values = []
        for key, value in updates.items():
            set_clauses.append(f"{key} = ?")
            values.append(value)
```

Fig B3: Update Function (1)



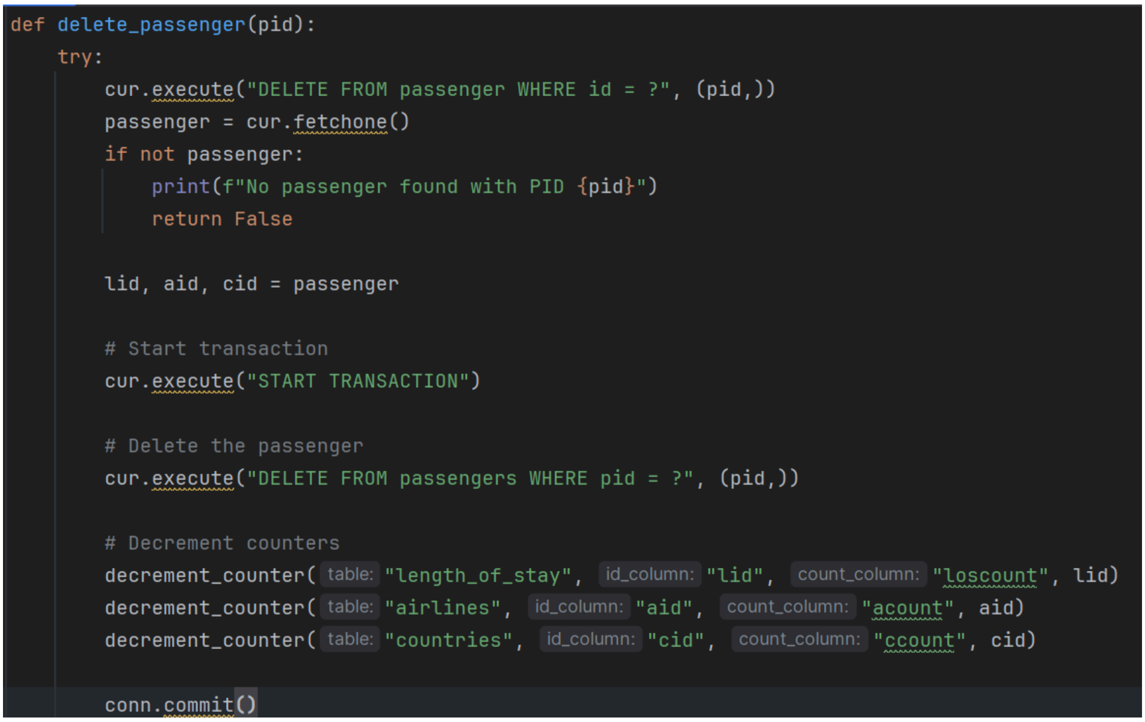
```

        query = f"UPDATE passenger SET {', '.join(set_clauses)} WHERE id = ?"
        values.append(pid)

        cur.execute(query, values)

        # Update counters
        if 'lid' in updates and updates['lid'] != current_lid:
            decrement_counter( table: "length_of_stay",  id_column: "lid",  count_column: "loscount", current_lid)
            increment_counter( table: "length_of_stay",  id_column: "lid",  count_column: "loscount", updates['lid'])

        if 'aid' in updates and updates['aid'] != current_aid:
            decrement_counter( table: "airlines",  id_column: "aid",  count_column: "acount", current_aid)
            increment_counter( table: "airlines",  id_column: "aid",  count_column: "acount", updates['aid'])

        if 'cid' in updates and updates['cid'] != current_cid:
            decrement_counter( table: "countries",  id_column: "cid",  count_column: "ccount", current_cid)
            increment_counter( table: "countries",  id_column: "cid",  count_column: "ccount", updates['cid'])
```

Fig B4: Update Function (2)



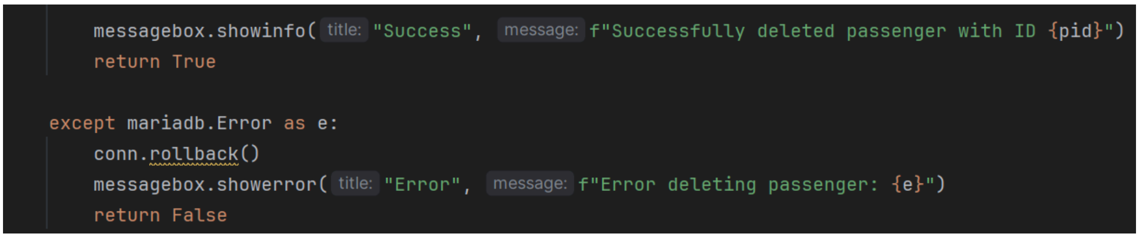
```

        conn.commit()

        messagebox.showinfo( title: "Success",  message: f"Successfully updated passenger with ID {pid}")
        return True

    except mariadb.Error as e:
        conn.rollback()
        messagebox.showerror( title: "Error",  message: f"Error updating passenger: {e}")
        return False

```

Fig B5: Update Function (3)



```
def delete_passenger(pid):
    try:
        cur.execute("DELETE FROM passenger WHERE id = ?", (pid,))
        passenger = cur.fetchone()
        if not passenger:
            print(f"No passenger found with PID {pid}")
            return False

        lid, aid, cid = passenger

        # Start transaction
        cur.execute("START TRANSACTION")

        # Delete the passenger
        cur.execute("DELETE FROM passengers WHERE pid = ?", (pid,))

        # Decrement counters
        decrement_counter( table: "length_of_stay",  id_column: "lid",  count_column: "loscount", lid)
        decrement_counter( table: "airlines",  id_column: "aid",  count_column: "acount", aid)
        decrement_counter( table: "countries",  id_column: "cid",  count_column: "ccount", cid)

        conn.commit()
```

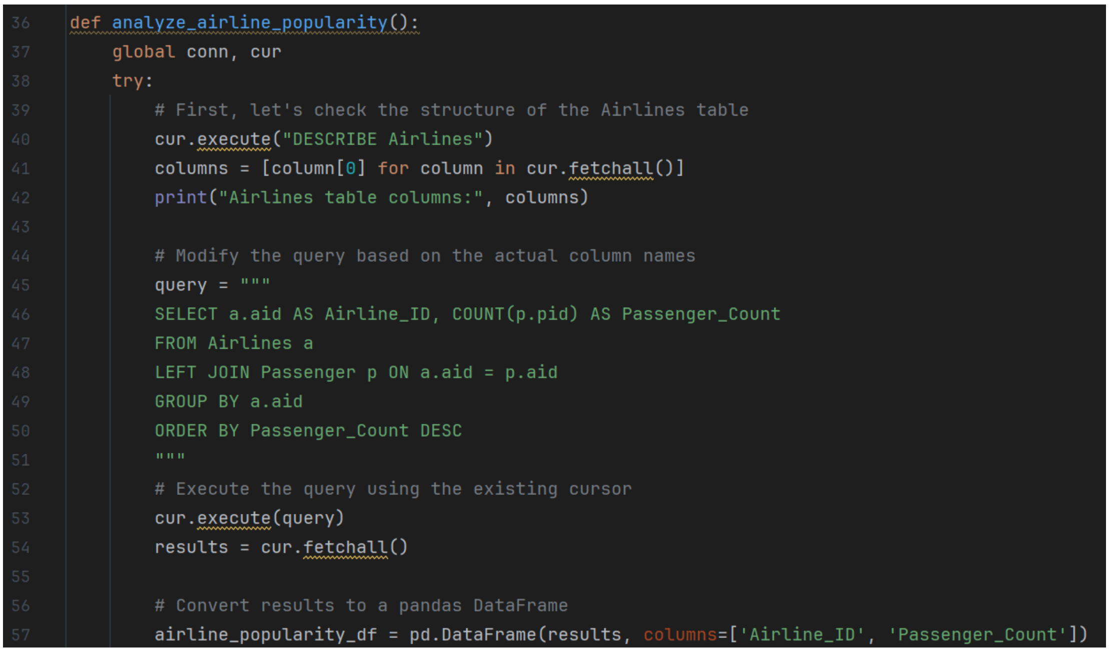
Fig B6: Delete Function (1)



```
        messagebox.showinfo( title: "Success",  message: f"Successfully deleted passenger with ID {pid}")
        return True

    except mariadb.Error as e:
        conn.rollback()
        messagebox.showerror( title: "Error",  message: f"Error deleting passenger: {e}")
        return False
```

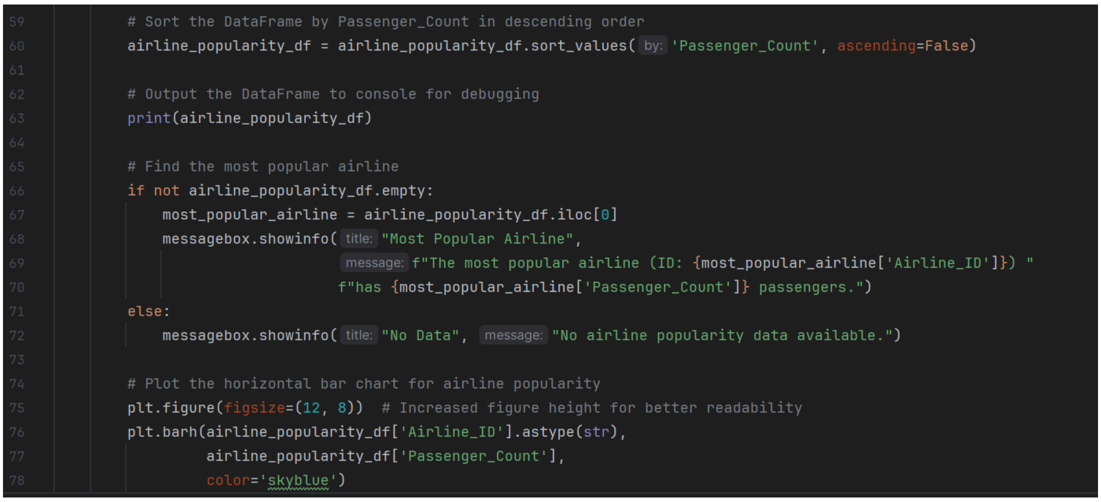
Fig B7: Delete Function (2)

Appendix C



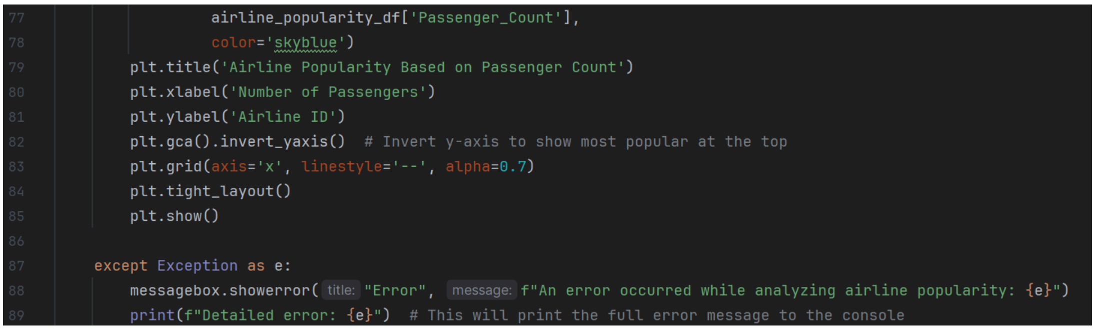
```python
def analyze_airline_popularity():
    global conn, cur
    try:
        # First, let's check the structure of the Airlines table
        cur.execute("DESCRIBE Airlines")
        columns = [column[0] for column in cur.fetchall()]
        print("Airlines table columns:", columns)

        # Modify the query based on the actual column names
        query = """
        SELECT a.aid AS Airline_ID, COUNT(p.pid) AS Passenger_Count
        FROM Airlines a
        LEFT JOIN Passenger p ON a.aid = p.aid
        GROUP BY a.aid
        ORDER BY Passenger_Count DESC
        """
        # Execute the query using the existing cursor
        cur.execute(query)
        results = cur.fetchall()

        # Convert results to a pandas DataFrame
        airline_popularity_df = pd.DataFrame(results, columns=['Airline_ID', 'Passenger_Count'])
```

Fig C1: Analyse Airline Popularity (1)



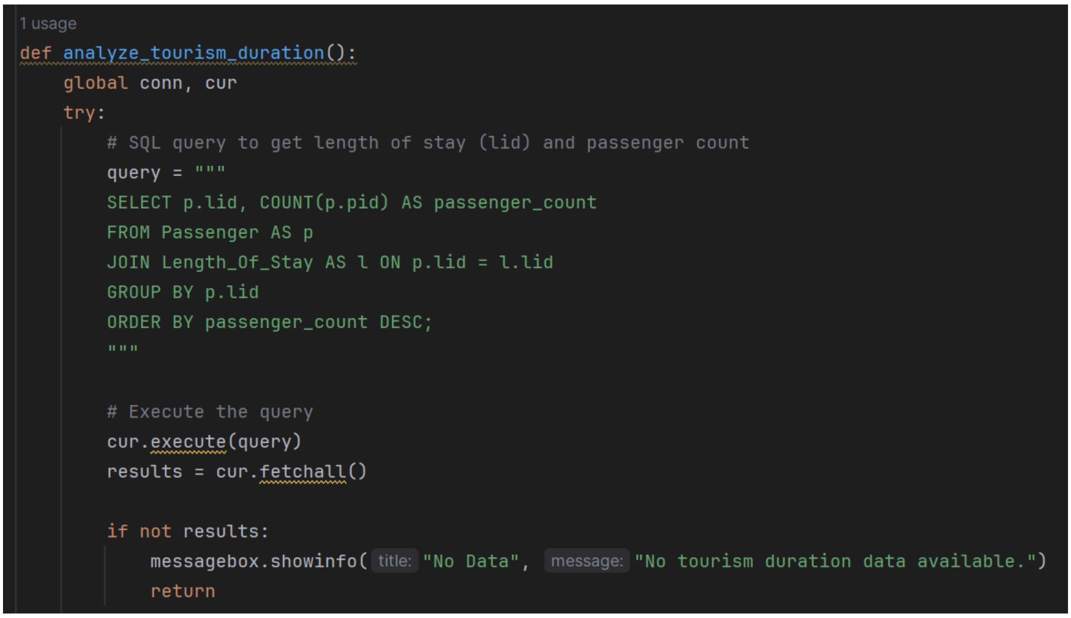
```python
        # Sort the DataFrame by Passenger_Count in descending order
        airline_popularity_df = airline_popularity_df.sort_values('Passenger_Count', ascending=False)

        # Output the DataFrame to console for debugging
        print(airline_popularity_df)

        # Find the most popular airline
        if not airline_popularity_df.empty:
            most_popular_airline = airline_popularity_df.iloc[0]
            messagebox.showinfo("Most Popular Airline",
                                f"The most popular airline (ID: {most_popular_airline['Airline_ID']}) "
                                f"has {most_popular_airline['Passenger_Count']} passengers.")
        else:
            messagebox.showinfo("No Data", "No airline popularity data available.")

        # Plot the horizontal bar chart for airline popularity
        plt.figure(figsize=(12, 8))  # Increased figure height for better readability
        plt.barh(airline_popularity_df['Airline_ID'].astype(str),
                 airline_popularity_df['Passenger_Count'],
                 color='skyblue')
```

Fig C2: Analyse Airline Popularity (2)



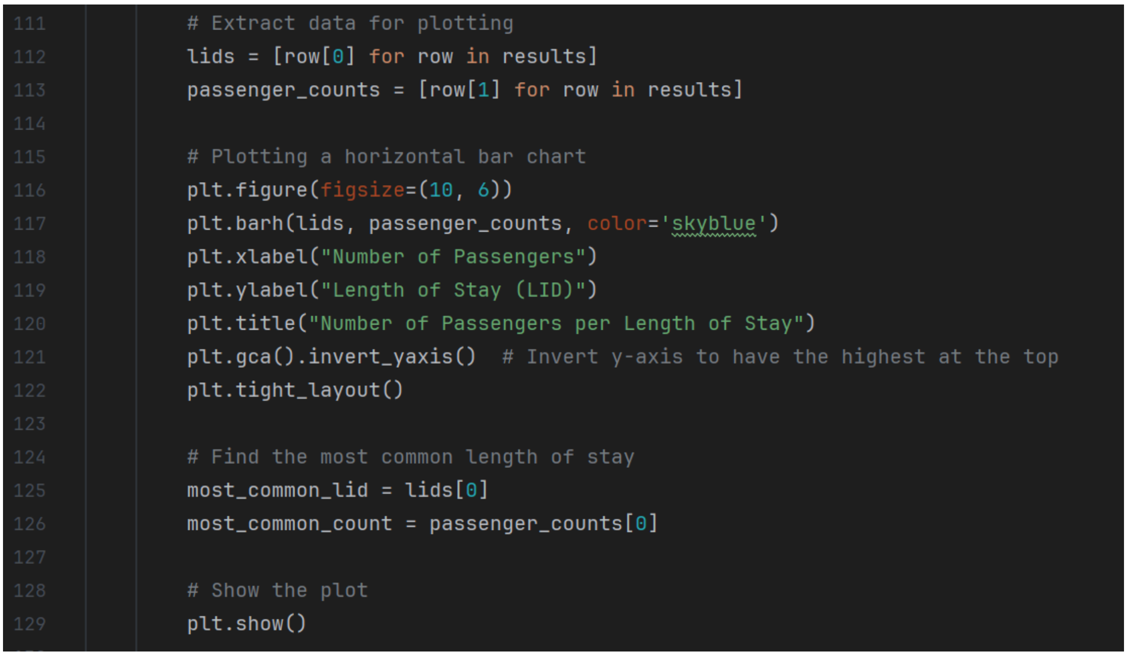
```python
                 airline_popularity_df['Passenger_Count'],
                 color='skyblue')
        plt.title('Airline Popularity Based on Passenger Count')
        plt.xlabel('Number of Passengers')
        plt.ylabel('Airline ID')
        plt.gca().invert_yaxis()  # Invert y-axis to show most popular at the top
        plt.grid(axis='x', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred while analyzing airline popularity: {e}")
        print(f"Detailed error: {e}")  # This will print the full error message to the console
```

Fig C3: Analyse Airline Popularity (3)



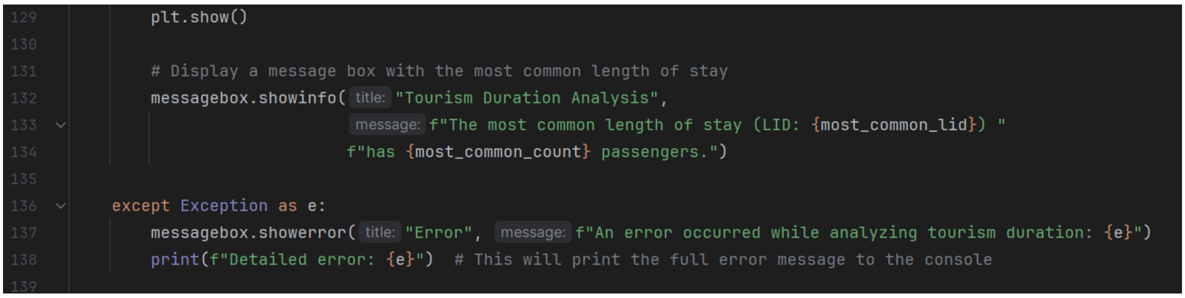
```
1 usage
def analyze_tourism_duration():
    global conn, cur
    try:
        # SQL query to get length of stay (lid) and passenger count
        query = """
        SELECT p.lid, COUNT(p.pid) AS passenger_count
        FROM Passenger AS p
        JOIN Length_Of_Stay AS l ON p.lid = l.lid
        GROUP BY p.lid
        ORDER BY passenger_count DESC;
        """

        # Execute the query
        cur.execute(query)
        results = cur.fetchall()

        if not results:
            messagebox.showinfo( title: "No Data",  message: "No tourism duration data available.")
            return
```

Fig C4: Analyse Tourism Duration (1)



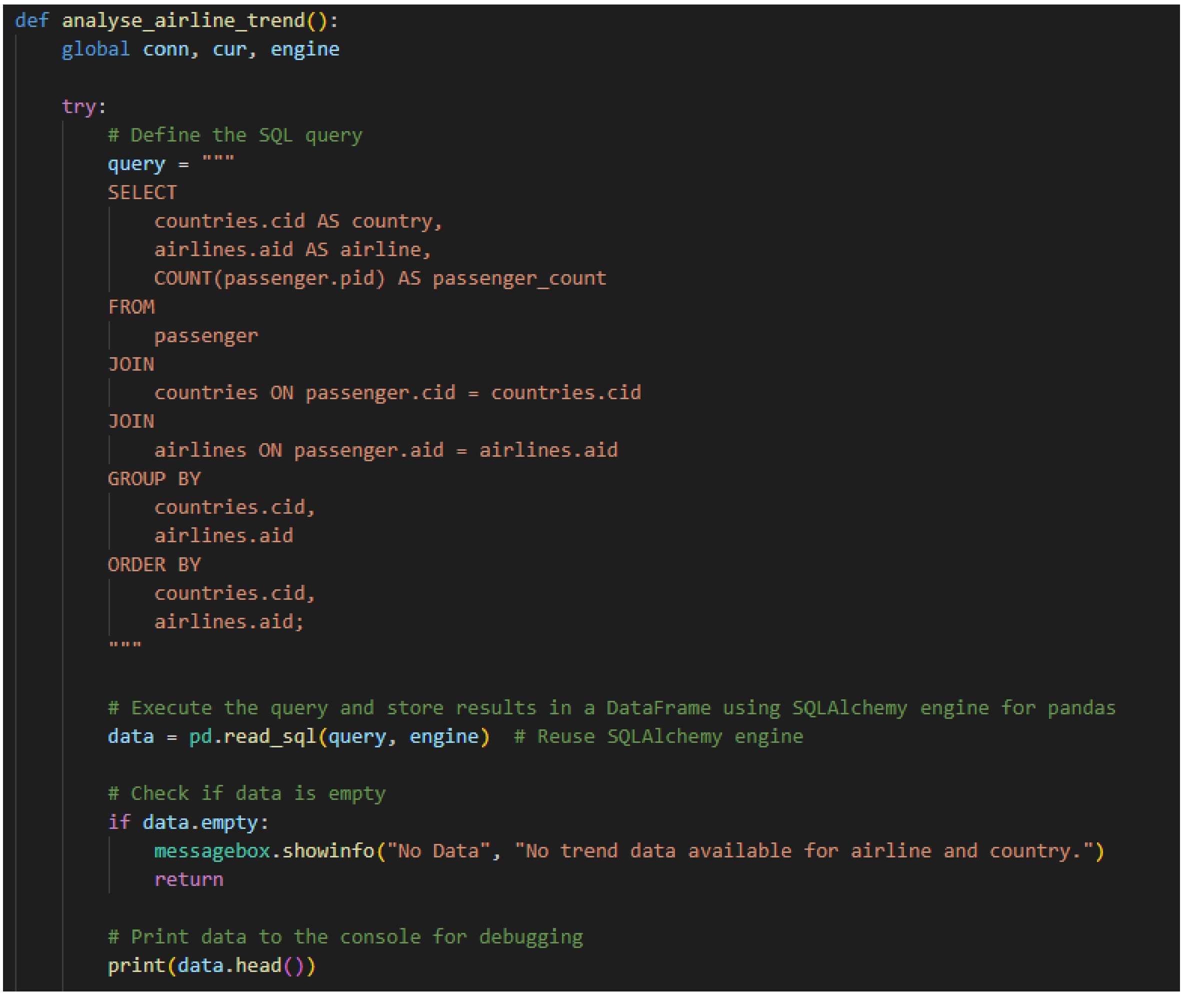
```
        # Extract data for plotting
        lids = [row[0] for row in results]
        passenger_counts = [row[1] for row in results]

        # Plotting a horizontal bar chart
        plt.figure(figsize=(10, 6))
        plt.barh(lids, passenger_counts, color='skyblue')
        plt.xlabel("Number of Passengers")
        plt.ylabel("Length of Stay (LID)")
        plt.title("Number of Passengers per Length of Stay")
        plt.gca().invert_yaxis()  # Invert y-axis to have the highest at the top
        plt.tight_layout()

        # Find the most common length of stay
        most_common_lid = lids[0]
        most_common_count = passenger_counts[0]

        # Show the plot
        plt.show()
```

Fig C5: Analyse Tourism Duration (2)



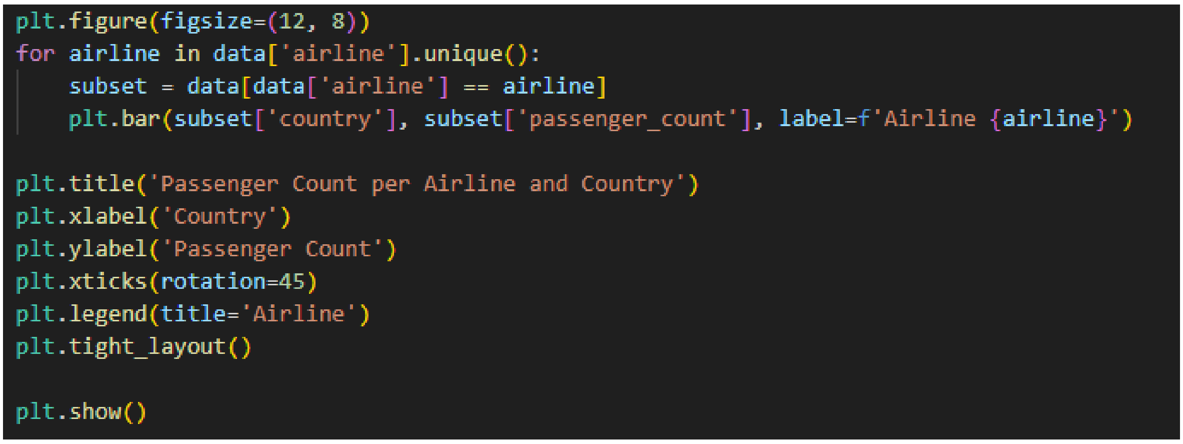
```
        plt.show()

        # Display a message box with the most common length of stay
        messagebox.showinfo( title: "Tourism Duration Analysis",
                             message: f"The most common length of stay (LID: {most_common_lid}) "
                            f"has {most_common_count} passengers.")

    except Exception as e:
        messagebox.showerror( title: "Error",  message: f"An error occurred while analyzing tourism duration: {e}")
        print(f"Detailed error: {e}")  # This will print the full error message to the console
```

Fig C6: Analyse Tourism Duration (3)



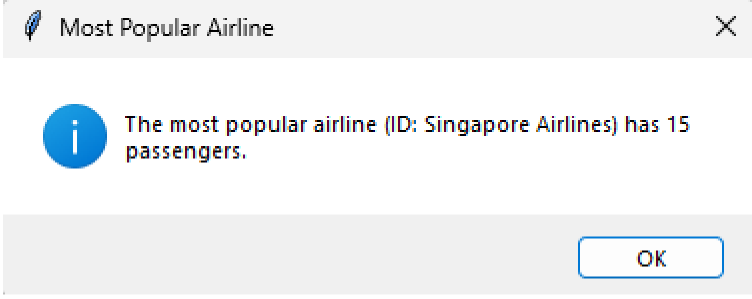
```python
def analyse_airline_trend():
    global conn, cur, engine

    try:
        # Define the SQL query
        query = """
        SELECT
            countries.cid AS country,
            airlines.aid AS airline,
            COUNT(passenger.pid) AS passenger_count
        FROM
            passenger
        JOIN
            countries ON passenger.cid = countries.cid
        JOIN
            airlines ON passenger.aid = airlines.aid
        GROUP BY
            countries.cid,
            airlines.aid
        ORDER BY
            countries.cid,
            airlines.aid;
        """

        # Execute the query and store results in a DataFrame using SQLAlchemy engine for pandas
        data = pd.read_sql(query, engine)  # Reuse SQLAlchemy engine

        # Check if data is empty
        if data.empty:
            messagebox.showinfo("No Data", "No trend data available for airline and country.")
            return

        # Print data to the console for debugging
        print(data.head())
```

Fig C7: Analyse Airline Trend (1)



```python
plt.figure(figsize=(12, 8))
for airline in data['airline'].unique():
    subset = data[data['airline'] == airline]
    plt.bar(subset['country'], subset['passenger_count'], label=f'Airline {airline}')

plt.title('Passenger Count per Airline and Country')
plt.xlabel('Country')
plt.ylabel('Passenger Count')
plt.xticks(rotation=45)
plt.legend(title='Airline')
plt.tight_layout()

plt.show()
```

Fig C8: Analyse Airline Trend (2)



The most popular airline (ID: Singapore Airlines) has 15 passengers.

Fig C9: Window popup of the most popular airline

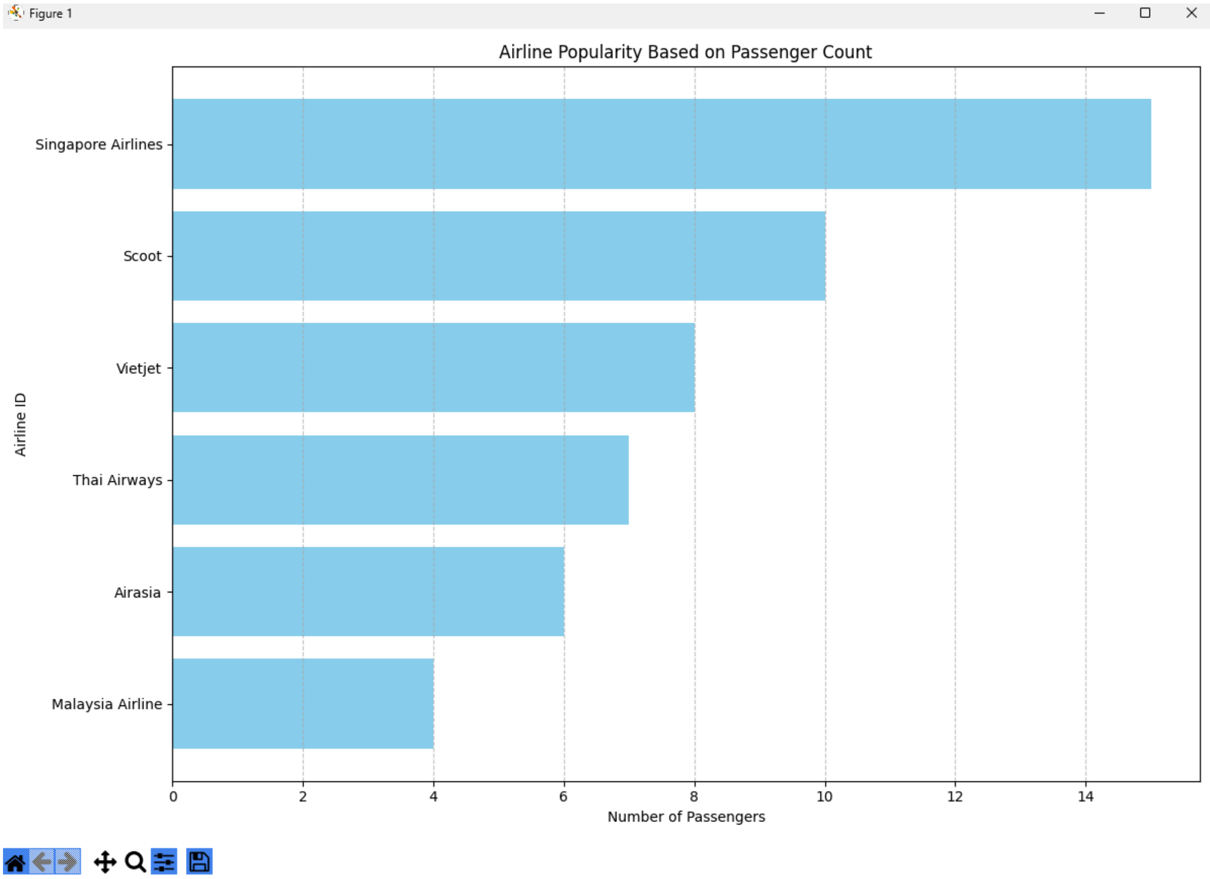
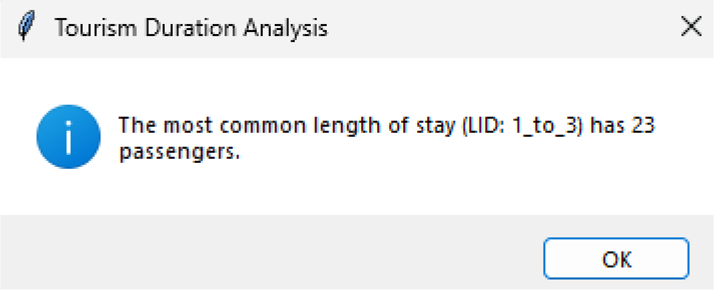


Fig C10: Horizontal Barchart to visualise airline popularity



The most common length of stay (LID: 1_to_3) has 23 passengers.

Fig C11: Window popup of the highest length of stay

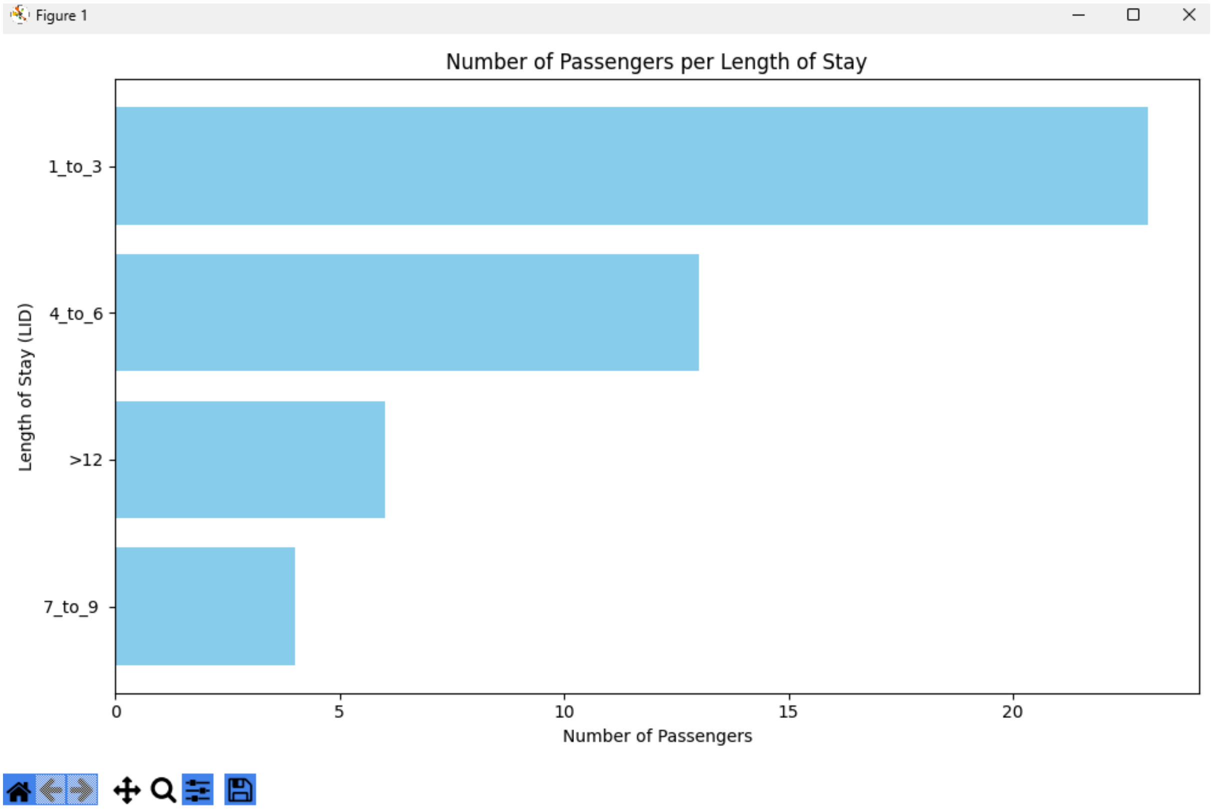


Fig C12: Horizontal Barchart to visualise highest length of stay

Appendix D

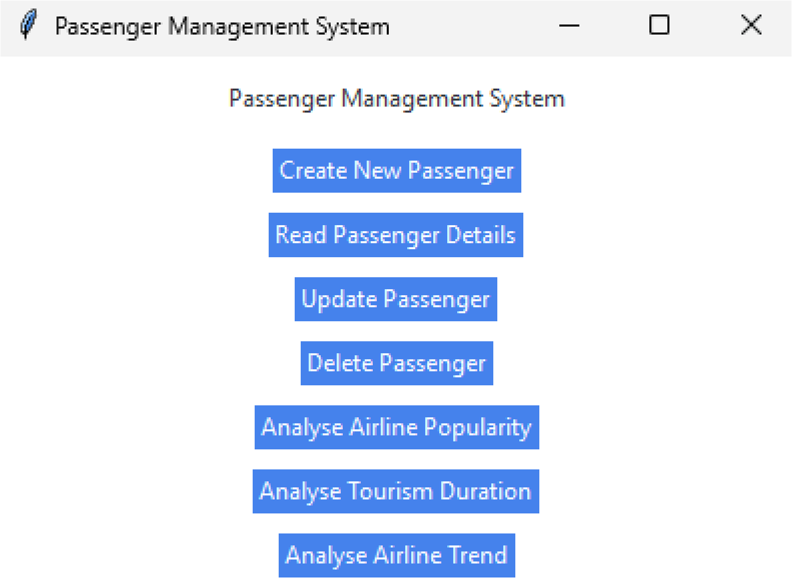
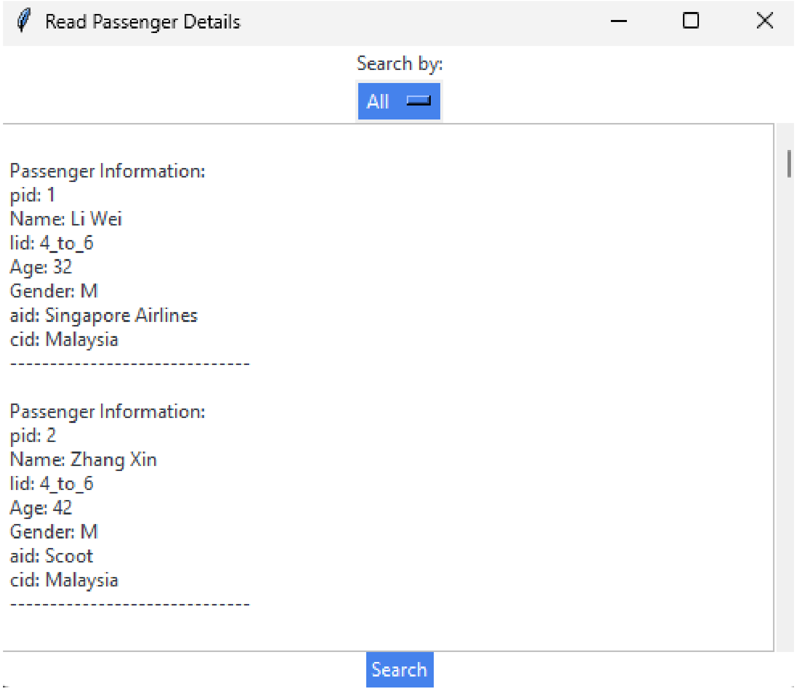


Fig D1: Initialise main window



Search by:
All

```
Passenger Information:
pid: 1
Name: Li Wei
lid: 4_to_6
Age: 32
Gender: M
aid: Singapore Airlines
cid: Malaysia
------------------------------

Passenger Information:
pid: 2
Name: Zhang Xin
lid: 4_to_6
Age: 42
Gender: M
aid: Scoot
cid: Malaysia
------------------------------
```

Fig D2: Window popup for viewing passenger details with filter

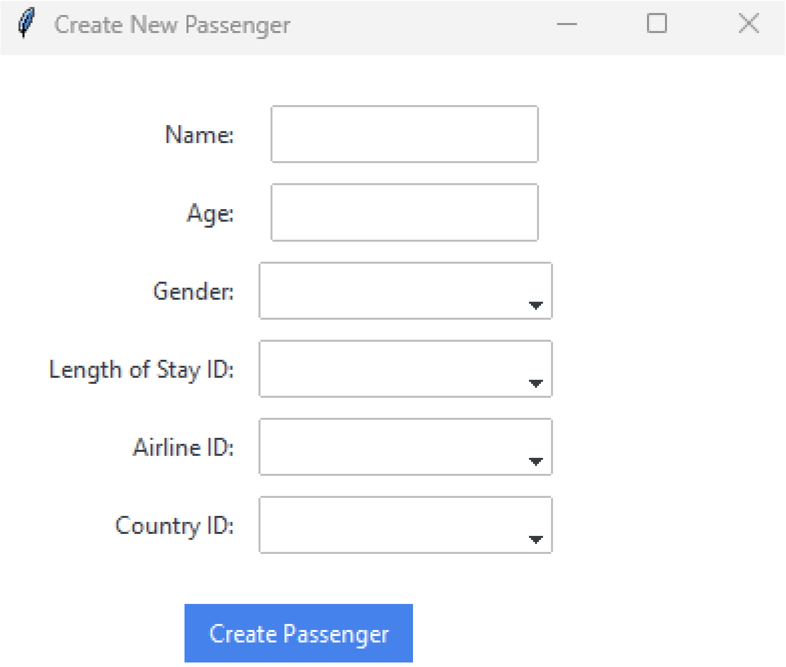


Fig D3: Window popup for creating passengers

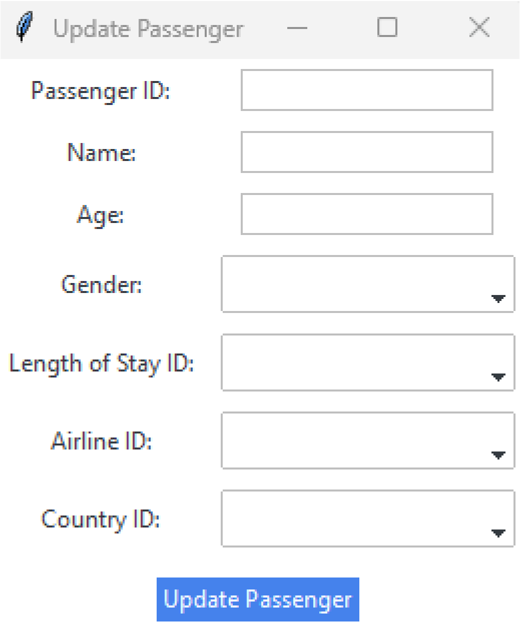


Fig D4: Window popup for updating passengers

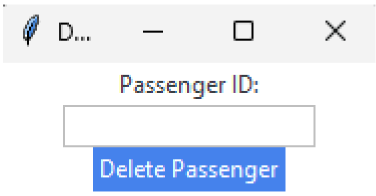
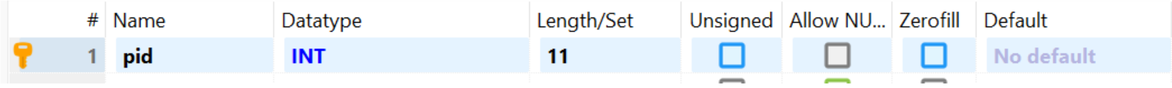


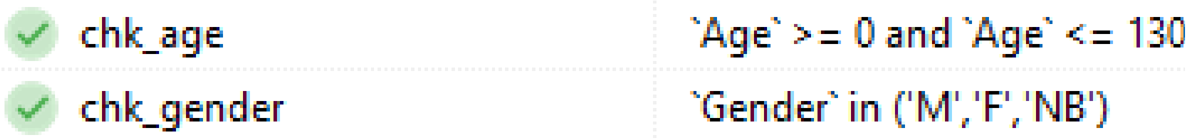
Fig D5: Window popup for deleting passengers

Appendix E



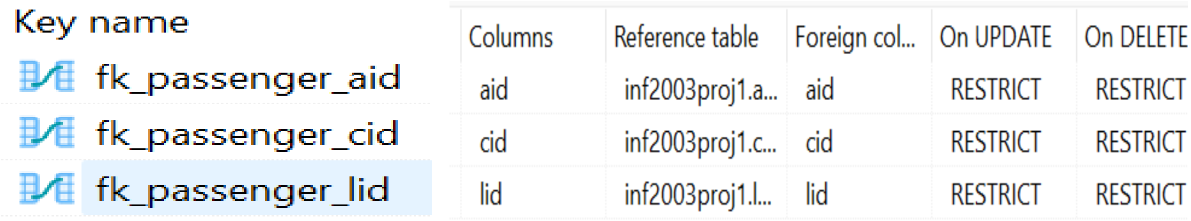
	#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default
🔑	1	pid	INT	11	□	□	□	No default

Fig E1: Primary Key and Not Null Constraint



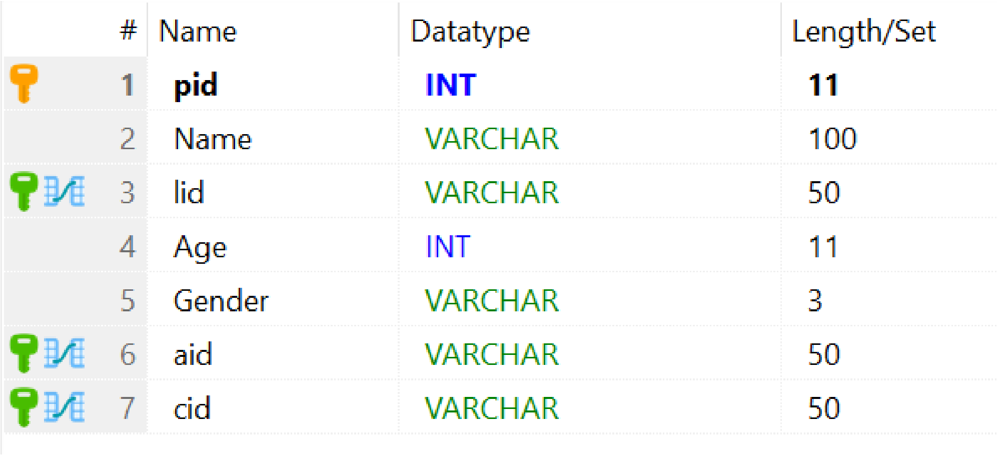
✓	chk_age	`Age` >= 0 and `Age` <= 130
✓	chk_gender	`Gender` in ('M','F','NB')

Fig E2: Check Constraint on Age and Gender for "Passenger" Table



Key name	Columns	Reference table	Foreign col...	On UPDATE	On DELETE
fk_passenger_aid	aid	inf2003proj1.a...	aid	RESTRICT	RESTRICT
fk_passenger_cid	cid	inf2003proj1.c...	cid	RESTRICT	RESTRICT
fk_passenger_lid	lid	inf2003proj1.l...	lid	RESTRICT	RESTRICT

Fig E3: Foreign Key Constraint on aid, cid and lid for "Passenger" Table



	#	Name	Datatype	Length/Set
🔑	1	pid	INT	11
	2	Name	VARCHAR	100
🔑	3	lid	VARCHAR	50
	4	Age	INT	11
	5	Gender	VARCHAR	3
🔑	6	aid	VARCHAR	50
🔑	7	cid	VARCHAR	50

Fig E4: Referential Integrity and Domain Constraints