SQL AIRLINE MANAGEMENT SYSTEM PROJECT

A Comprehensive Database Solution for Airline Operations

PROJECT CATEGORY : Database Management System (DBMS)

TOOLS & TECHNOLOGIES USED :

- SQL
- MySQL
- MySQL Workbench

DEVELOPED BY : Safiya Fatima

Project Overview :

- This project is a **SQL-based Airline Management System** developed using **MySQL Workbench**.
- It demonstrates how airline data such as **flights, passengers, and bookings** can be stored, managed, and analyzed using structured SQL queries.
- The project focuses on **real-world airline operations**, including passenger bookings, ticket pricing analysis, revenue calculation, and travel routes.

Technologies Used :

- SQL
- MySQL
- MySQL Workbench

Database Tables :

The project consists of three main tables:

- **FLIGHTS**

Stores flight details such as flight number, source city, and destination city.

- **PASSENGERS**

Stores passenger details including name, age, and city.

- **BOOKINGS**

Stores booking information such as passenger ID, flight ID, ticket price, and travel date.

All tables are connected using **primary keys and foreign keys** to maintain data integrity.

Key Features :

- Designed a relational airline database schema
- Implemented primary key and foreign key relationships
- Used JOIN operations to combine data from multiple tables
- Analyzed ticket prices using aggregate functions
- Calculated total revenue per destination
- Categorized passengers based on ticket price using CASE statements
- Applied subqueries for advanced filtering and analysis

How to Run the Project :

1. Open **MySQL Workbench**
2. Create a new SQL file
3. Copy and paste the SQL script from the **Project Implementation (SQL Queries and Outputs)** section
   OR
   download the complete SQL script from the **GitHub repository / OneDrive link** provided in this project.
4. **Execute the script step-by-step** (database creation → tables → inserts → queries)
5. View the results of each task using SELECT queries

Project Repository & Source Code links :

- GitHub Repository : https://github.com/safiyafatima34-cpu/SQL-Airline-Management-System

- SQL Script (OneDrive) : https://1drv.ms/u/c/e8b5f8901446d194/IQDSN1eGbn-EQYvXEmSgdlyBAQax0Izy6i5MbJefj0K3jgM?e=XsdthT

Concepts Covered :

- Database design and normalization
- Primary key and foreign key constraints
- INNER JOIN operations
- Aggregate functions (SUM, AVG)
- GROUP BY clause
- Subqueries
- CASE statements
- Data filtering using WHERE, BETWEEN, and logical operators

Key Performance Indicators (KPIs) :

The following KPIs were derived using SQL aggregate functions to analyze booking and revenue trends in the Airline Management System:

1. Total Ticket Revenue

- Represents the total income generated from all flight bookings.
- Calculated using SUM(ticket_price).

2. Average Ticket Price

- Indicates the average cost of a ticket across all bookings.
- Calculated using AVG(ticket_price).

3. Most Expensive Booking

- Identifies the highest-priced ticket booked in the system.
- Useful for understanding premium travel patterns.

4. Revenue by Destination

- Shows total ticket revenue generated for each destination city.
- Helps identify high-revenue destinations.

5. Average Spend per Passenger

- Represents the average amount spent by each passenger on flight bookings.
- Useful for analyzing passenger value.

Project Status :

✓ Completed

✓ Tested in MySQL Workbench

This project demonstrates **practical SQL skills** through real-world airline data scenarios.

Project Requirements and Problem Statement :

1. Database Design Requirements (Tables) :

Below is the given Database Schema (As Provided by my mentor) :

- **FLIGHTS TABLE :**

PK FLIGHTS

Column1	Column2	Column3	Column4
flight_id	flight_no	source_city	destination
1	A1	mumbai	delhi
2	A2	delhi	chennai
3	A3	banglore	mumbai
4	A4	chennai	pune
5	A5	mumbai	banglore
6	A6	hyderabad	pune

- **PASSENGERS TABLE :**

passengers

PK

Column1	Column2	Column3	Column4
passenger_id	name	age	city
1			mumbai
2			delhi
3			banglore
4			chennai
5			pune

- **BOOKINGS TABLE :**

bookings				
PK	FK	FK		
Column1	Column2	Column3	Column4	Column5
booking_id	passenger_id	flight_id	ticket_price	travel_date
1	1	1		12-10-2025
2	2	2		13-10-2025
3	3	2		14-10-2025
4	4	3		15-10-2025
5	5	4		16-10-2025
6	1	5		12-10-2025
7	2	5		18-10-2025
8	2	6		12-10-2025
9	3	1		20-10-2025
10	4	4		21-10-2025

The above tables represent the database schema provided as part of the project requirements.

The project is designed using three relational tables: Passengers, Flights, and Bookings, with appropriate primary and foreign key relationships.

2. Task / Query Requirements :

Below has the Task Requirements (As Provided by my mentor) :

TASKS

Task 1: Show all passengers with their flight details
Task 2: List all flights that depart from Mumbai
Task 3: Find average ticket price for each flight route
Task 4: Show passengers who booked tickets costing more than ₹6000
Task 5: Find the most expensive flight booked
Task 6: Categorize passengers based on ticket price using case
Task 7: Find passengers who paid above the average ticket price
Task 8 — List all passengers who are from Delhi or Mumbai
Task 9 — Show bookings made between '2025-10-09' and '2025-10-12'
Task 10 — Show total ticket revenue for each destination
Task 11 — Find average ticket price paid by each passenger
Task 12 — Show passenger name, flight number, and travel date
Task 13 — Find which cities passengers are traveling to
Task 14 -- Show all passengers with their booked flight numbers
Task 15 _ Show passenger names with their travel route (source → destination)
Task 16 _ Show total amount spent by each passenger

AIRLINE SYSTEM

The above image shows the set of 16 SQL tasks assigned as part of the Airline Management System project.

These tasks involve data retrieval, joins, aggregation functions, and conditional logic based on the given database schema.

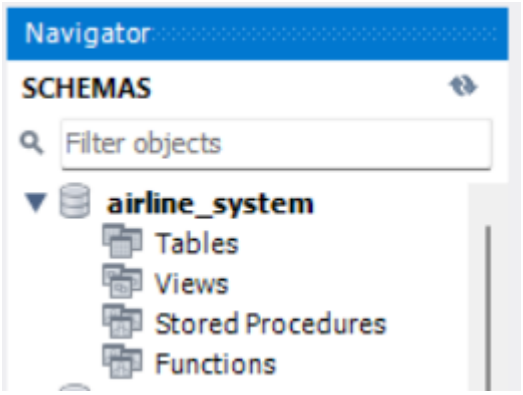
Project Implementation (SQL Queries and Outputs) :

-- CREATING DATABASE NAMED "AIRLINE_SYSTEM" & USING IT

SQL query :

```
create database AIRLINE_SYSTEM;

use AIRLINE_SYSTEM;
```

OUTPUT :



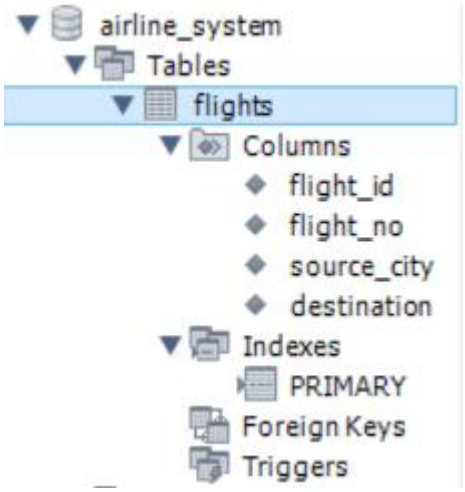
-- TABLE-1 : FLIGHTS

-- 1.) CREATING TABLE-1 NAMED "FLIGHTS"

SQL query :

```
create table FLIGHTS (

flight_id int primary key not null,

flight_no varchar (10),

source_city varchar (30),

destination varchar (30)

);
```

OUTPUT :



-- 2.) INSERTING VALUES IN TABLE-1 (FLIGHTS)

SQL query :

```
insert into FLIGHTS (flight_id, flight_no, source_city, destination)
values
(1,'A1','mumbai','delhi'),
(2,'A2','delhi','chennai'),
(3,'A3','banglore','mumbai'),
(4,'A4','chennai','pune'),
(5,'A5','mumbai','banglore'),
(6,'A6','hyderabad','pune');
```

OUTPUT :

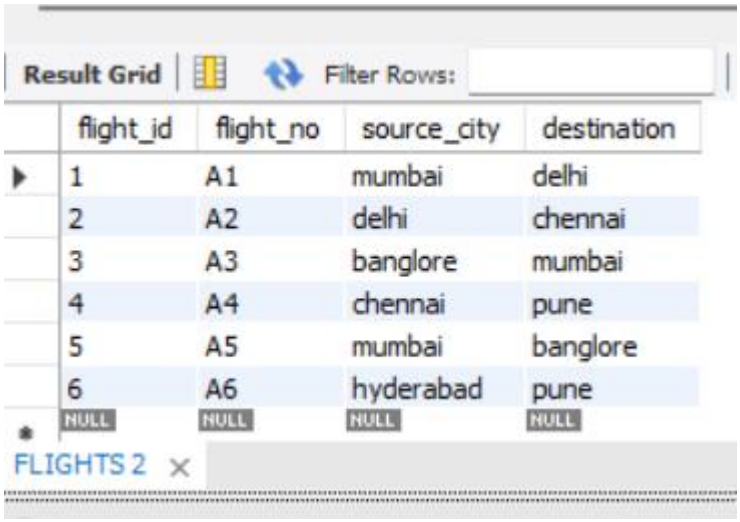
All the above values used in the SQL query are inserted into the **FLIGHTS** table.

-- 3.) VIEWING TABLE-1 (FLIGHTS)

SQL query :

```
select * from FLIGHTS;
```

OUTPUT :



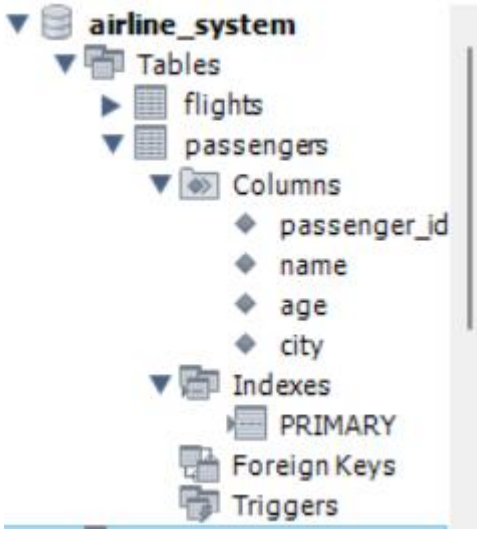
-- TABLE 2 : PASSENGERS

-- 1.) CREATING TABLE-2 NAMED "PASSENGERS"

SQL query :

```
create table PASSENGERS (
passenger_id int primary key not null,
name varchar (30),
age int,
city varchar (30)
);
```

OUTPUT :



-- 2.) INSERTING VALUES IN TABLE-2 (PASSENGERS)

SQL query :

```
insert into PASSENGERS (passenger_id, name, age, city)
values
(1,'safiya','40','mumbai'),
(2,'zoya','35','delhi'),
(3,'hajera','50','banglore'),
(4,'ayesha','25','chennai'),
(5,'azlaan','18','pune');
```

OUTPUT :

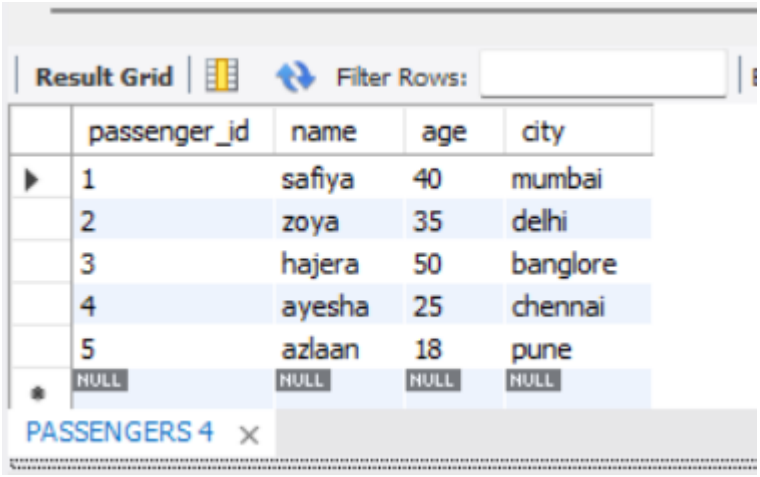
All the above values used in the SQL query are inserted into the **PASSENGERS** table.

-- 3.) VIEWING TABLE-2 (PASSENGERS)

SQL query :

```
select * from PASSENGERS;
```

OUTPUT :



--TABLE 3 : BOOKINGS

-- 1.) CREATING TABLE-3 NAMED "BOOKINGS"

SQL query :

```
create table BOOKINGS (
```

booking_id int primary key not null,

passenger_id int,

foreign key (passenger_id) references passengers (passenger_id),

flight_id int,

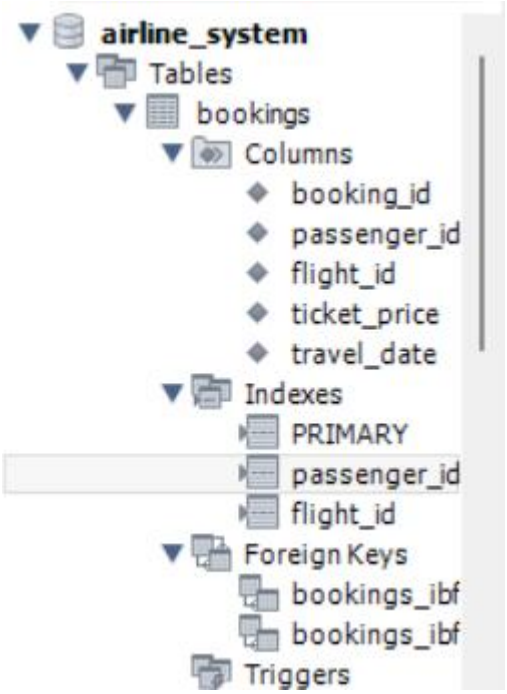
foreign key (flight_id) references flights (flight_id),

ticket_price int,

travel_date date

);

OUTPUT :



-- 2.) INSERTING VALUES IN TABLE-3 (BOOKINGS)

SQL query :

insert into BOOKINGS (booking_id, passenger_id, flight_id, ticket_price, travel_date)

values

(1, 1, 1, 50000, '2025-10-12'),

(2, 2, 2, 48000, '2025-10-13'),

(3, 3, 2, 55000, '2025-10-14'),

(4, 4, 3, 61000, '2025-10-15'),

(5, 5, 4, 45000, '2025-10-16'),

(6, 1, 5, 71000, '2025-10-12'),

(7, 2, 5, 40000, '2025-10-18'),

(8, 2, 6, 60000, '2025-10-12'),

(9, 3, 1, 50000, '2025-10-20'),

(10, 4, 4, 47000, '2025-10-21');

OUTPUT :

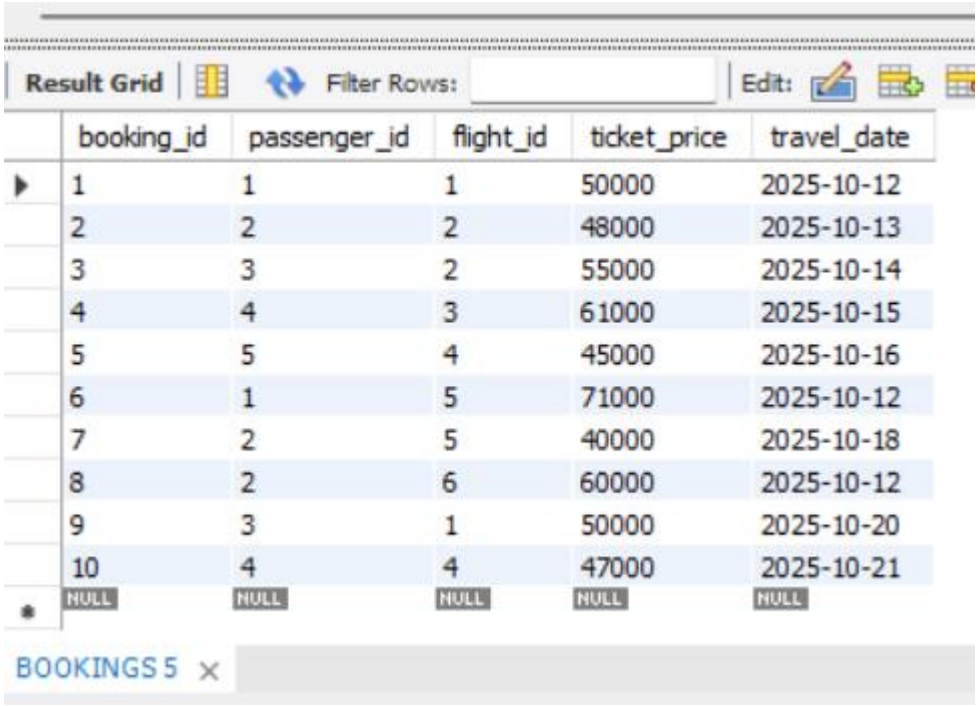
All the above values used in the SQL query are inserted into the **BOOKINGS** table.

-- 3.) VIEWING TABLE-3 (BOOKINGS)

SQL query :

select * from BOOKINGS;

OUTPUT :



booking_id	passenger_id	flight_id	ticket_price	travel_date
1	1	1	50000	2025-10-12
2	2	2	48000	2025-10-13
3	3	2	55000	2025-10-14
4	4	3	61000	2025-10-15
5	5	4	45000	2025-10-16
6	1	5	71000	2025-10-12
7	2	5	40000	2025-10-18
8	2	6	60000	2025-10-12
9	3	1	50000	2025-10-20
10	4	4	47000	2025-10-21
NULL	NULL	NULL	NULL	NULL

-- Task 1: Show all passengers with their flight details

SQL query :

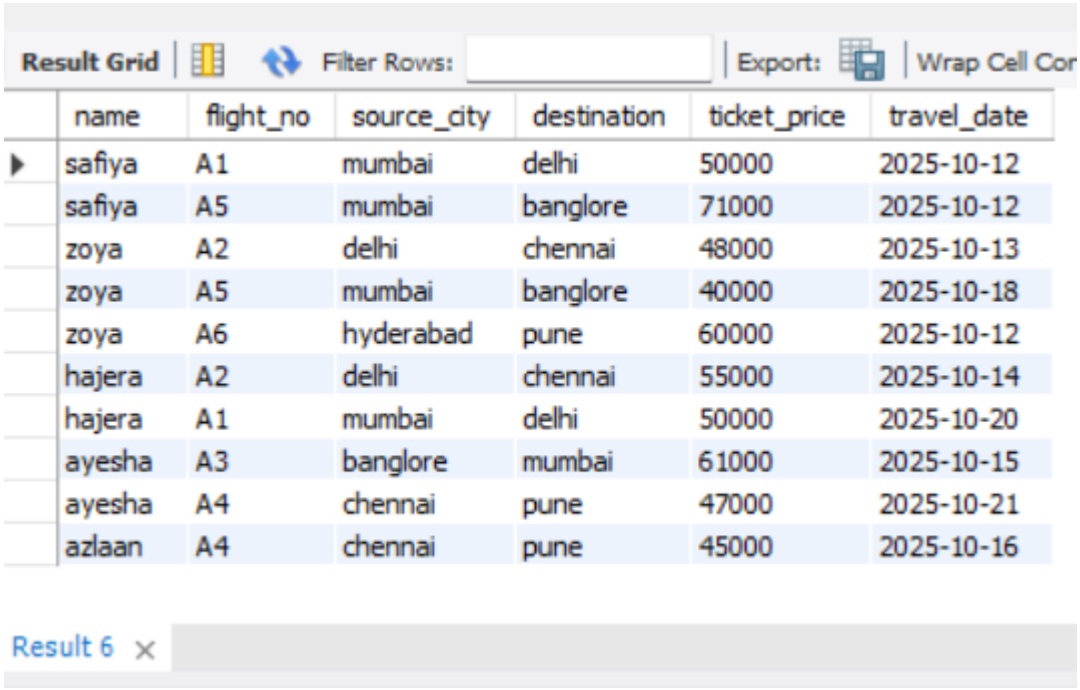
select p.name, f.flight_no, f.source_city, f.destination, b.ticket_price, b.travel_date

from passengers p

join bookings b on p.passenger_id = b.passenger_id

join flights f on b.flight_id = f.flight_id;

OUTPUT :



Result Grid | Filter Rows: | Export: | Wrap Cell Cor

name	flight_no	source_city	destination	ticket_price	travel_date
safiya	A1	mumbai	delhi	50000	2025-10-12
safiya	A5	mumbai	banglore	71000	2025-10-12
zoya	A2	delhi	chennai	48000	2025-10-13
zoya	A5	mumbai	banglore	40000	2025-10-18
zoya	A6	hyderabad	pune	60000	2025-10-12
hajera	A2	delhi	chennai	55000	2025-10-14
hajera	A1	mumbai	delhi	50000	2025-10-20
ayesha	A3	banglore	mumbai	61000	2025-10-15
ayesha	A4	chennai	pune	47000	2025-10-21
azlaan	A4	chennai	pune	45000	2025-10-16

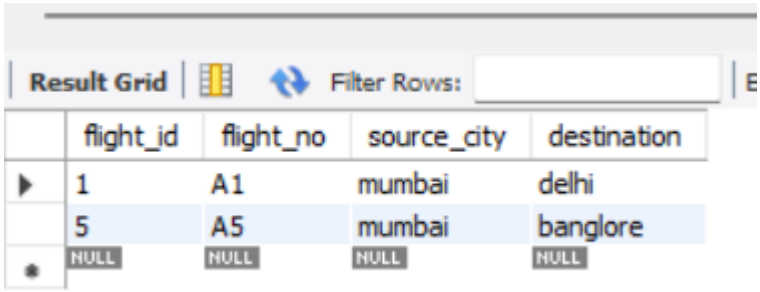
Result 6 ×

-- Task 2: List all flights that depart (leave) from Mumbai

SQL query :

select * from flights where source_city = 'mumbai';

OUTPUT :



Result Grid | Filter Rows:

flight_id	flight_no	source_city	destination
1	A1	mumbai	delhi
5	A5	mumbai	banglore
NULL	NULL	NULL	NULL

-- Task 3: Find average ticket price for each flight route(source → destination)

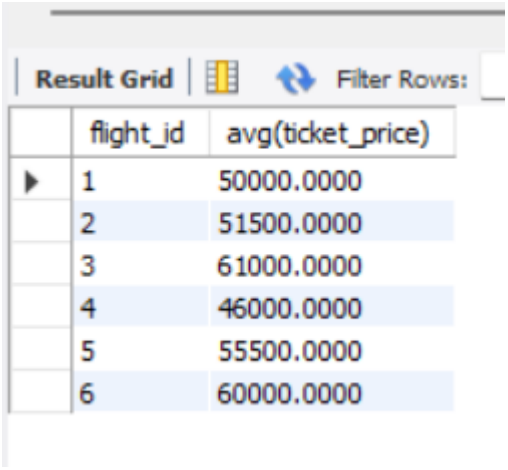
SQL query :

select flight_id, avg(ticket_price)

from bookings

group by flight_id;

OUTPUT :



Result Grid | Filter Rows:

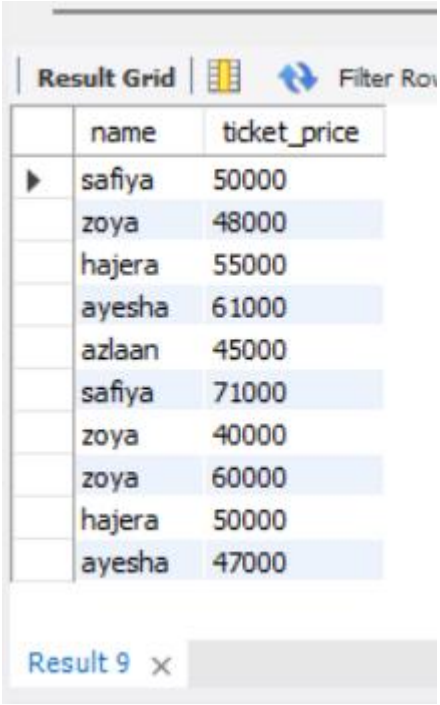
flight_id	avg(ticket_price)
1	50000.0000
2	51500.0000
3	61000.0000
4	46000.0000
5	55500.0000
6	60000.0000

-- Task 4: Show passengers who booked tickets costing more than 6000 rupees

SQL query :

```
SELECT p.name, b.ticket_price
FROM BOOKINGS b
JOIN PASSENGERS p ON b.passenger_id = p.passenger_id
WHERE b.ticket_price > 6000;
```

OUTPUT :



Result Grid | Filter Ro

name	ticket_price
safiya	50000
zoya	48000
hajera	55000
ayesha	61000
azlaan	45000
safiya	71000
zoya	40000
zoya	60000
hajera	50000
ayesha	47000

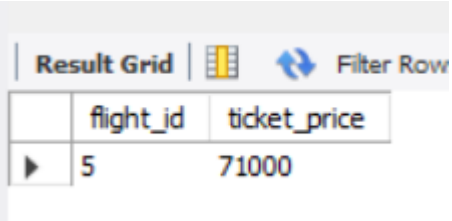
Result 9

-- Task 5: Find the most expensive flight booked

SQL query :

```
select flight_id, ticket_price
from bookings
order by ticket_price desc limit 1;
```

OUTPUT :



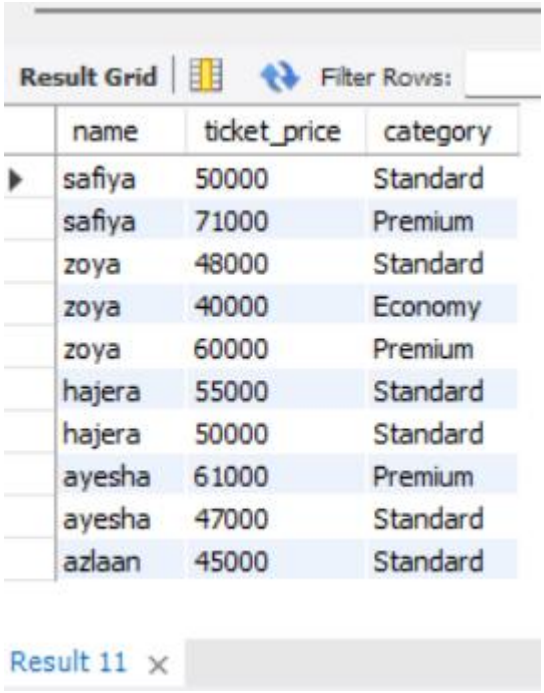
	flight_id	ticket_price
▶	5	71000

-- Task 6: Categorize passengers based on ticket price using case

SQL query :

```
SELECT p.name, b.ticket_price,
CASE
   WHEN b.ticket_price >= 60000 THEN 'Premium'
   WHEN b.ticket_price BETWEEN 45000 AND 59999 THEN 'Standard'
   ELSE 'Economy'
END AS category
FROM PASSENGERS p
JOIN BOOKINGS b ON p.passenger_id = b.passenger_id;
```

OUTPUT :



Result Grid | Filter Rows:

name	ticket_price	category
safiya	50000	Standard
safiya	71000	Premium
zoya	48000	Standard
zoya	40000	Economy
zoya	60000	Premium
hajera	55000	Standard
hajera	50000	Standard
ayesha	61000	Premium
ayesha	47000	Standard
azlaan	45000	Standard

Result 11

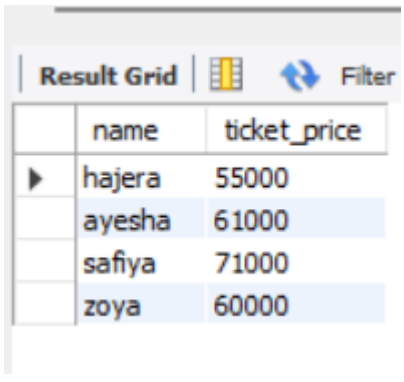
-- Task 7: Find passengers who paid above the average ticket price

-- avg(ticket_price)----> 52700

SQL query :

```
SELECT p.name, b.ticket_price
FROM BOOKINGS b
JOIN PASSENGERS p ON b.passenger_id = p.passenger_id
WHERE b.ticket_price > (SELECT AVG(ticket_price) FROM BOOKINGS);
```

OUTPUT :



Result Grid | Filter

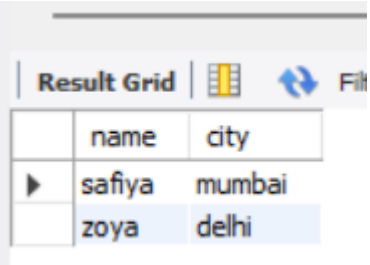
name	ticket_price
hajera	55000
ayesha	61000
safiya	71000
zoya	60000

-- Task 8 : List all passengers who are from Delhi or Mumbai

SQL query :

select name,city from passengers where city = 'delhi' or city = 'mumbai';

OUTPUT :



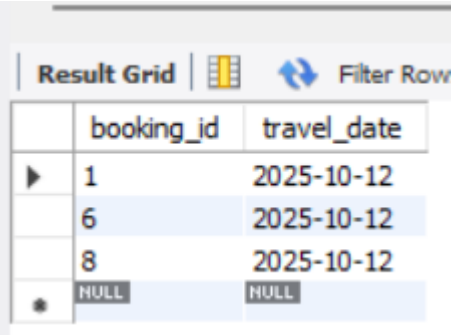
name	city
safiya	mumbai
zoya	delhi

-- Task 9: Show bookings made between '2025-10-09' and '2025-10-12'

SQL query :

select booking_id,travel_date from bookings where travel_date between '2025-10-09' and '2025-10-12';

OUTPUT :



booking_id	travel_date
1	2025-10-12
6	2025-10-12
8	2025-10-12
NULL	NULL

-- Task 10 : Show total ticket revenue (total income) for each destination

SQL query :

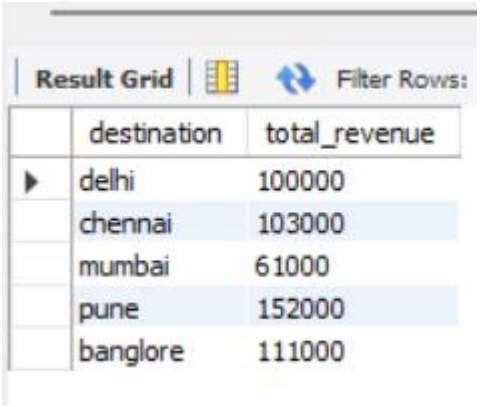
SELECT f.destination, SUM(b.ticket_price) AS total_revenue

FROM BOOKINGS b

JOIN FLIGHTS f ON b.flight_id = f.flight_id

GROUP BY f.destination;

OUTPUT :



Result Grid | Filter Rows:

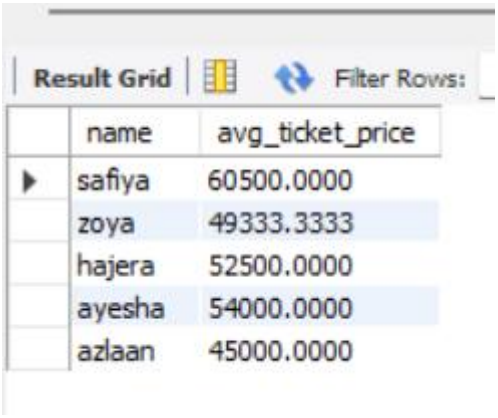
destination	total_revenue
delhi	100000
chennai	103000
mumbai	61000
pune	152000
banglore	111000

-- Task 11 : Find average ticket price paid by each passenger

SQL query :

```
SELECT p.name, AVG(b.ticket_price) AS avg_ticket_price
FROM BOOKINGS b
JOIN PASSENGERS p ON b.passenger_id = p.passenger_id
GROUP BY p.name;
```

OUTPUT :



Result Grid | Filter Rows:

name	avg_ticket_price
safiya	60500.0000
zoya	49333.3333
hajera	52500.0000
ayesha	54000.0000
azlaan	45000.0000

-- Task 12 : Show passenger name, flight number, and travel date

SQL query :

```
SELECT p.name, f.flight_no, b.travel_date
FROM BOOKINGS b
JOIN PASSENGERS p ON b.passenger_id = p.passenger_id
JOIN FLIGHTS f ON b.flight_id = f.flight_id;
```

OUTPUT :

Result Grid | Filter Rows:

name	flight_no	travel_date
safiya	A1	2025-10-12
safiya	A5	2025-10-12
zoya	A2	2025-10-13
zoya	A5	2025-10-18
zoya	A6	2025-10-12
hajera	A2	2025-10-14
hajera	A1	2025-10-20
ayesha	A3	2025-10-15
ayesha	A4	2025-10-21
azlaan	A4	2025-10-16

-- Task 13 : Find which cities passengers are traveling to

SQL query :

```
select destination from flights;
```

OUTPUT :

Result Grid

destination
delhi
chennai
mumbai
pune
banglore
pune

-- Task 14 : Show all passengers with their booked flight numbers

SQL query :

```
SELECT p.name, f.flight_no
FROM PASSENGERS p
JOIN BOOKINGS b ON p.passenger_id = b.passenger_id
JOIN FLIGHTS f ON b.flight_id = f.flight_id;
```

OUTPUT :

Result Grid

name	flight_no
safiya	A1
safiya	A5
zoya	A2
zoya	A5
zoya	A6
hajera	A2
hajera	A1
ayesha	A3
ayesha	A4
azlaan	A4

Result 19

-- Task 15 : Show passenger names with their travel route (source → destination)

SQL query :

```
SELECT
   p.name AS passenger_name,
   f.source_city,
   f.destination
FROM bookings b
JOIN passengers p ON b.passenger_id = p.passenger_id
JOIN flights f ON b.flight_id = f.flight_id;
```

OUTPUT :

passenger_name	source_city	destination
safiya	mumbai	delhi
safiya	mumbai	banglore
zoya	delhi	chennai
zoya	mumbai	banglore
zoya	hyderabad	pune
hajera	delhi	chennai
hajera	mumbai	delhi
ayesha	banglore	mumbai
ayesha	chennai	pune
azlaan	chennai	pune

-- Task 16 : Show total amount spent by each passenger

SQL query :

```
SELECT p.name, SUM(b.ticket_price) AS total_amount_spent
FROM PASSENGERS p
JOIN BOOKINGS b ON p.passenger_id = b.passenger_id
GROUP BY p.name;
```

OUTPUT :

name	total_amount_spent
safiya	121000
zoya	148000
hajera	105000
ayesha	108000
azlaan	45000