Air Cargo Analysis

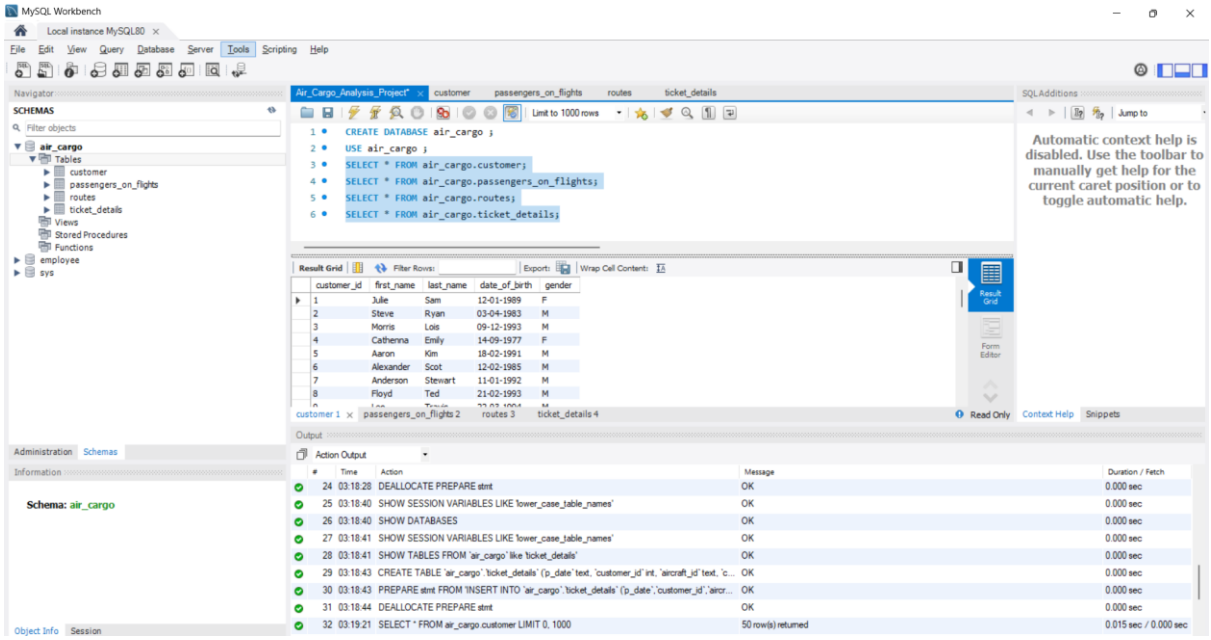
Problem Statement Scenario:

Air Cargo is an aviation company that provides air transportation services for passengers and freight. Air Cargo uses its aircraft to provide different services with the help of partnerships or alliances with other airlines. The company wants to prepare reports on regular passengers, busiest routes, ticket sales details, and other scenarios to improve the ease of travel and booking for customers.

Project Objective:

We, as a DBA expert, need to focus on identifying the regular customers to provide offers, analyse the busiest route which helps to increase the number of aircraft required and prepare an analysis to determine the ticket sales details. This will ensure that the company improves its operability and becomes more customer-centric and a favourable choice for air travel.

- **Importing Datasets:**
- customer



Conclusion:

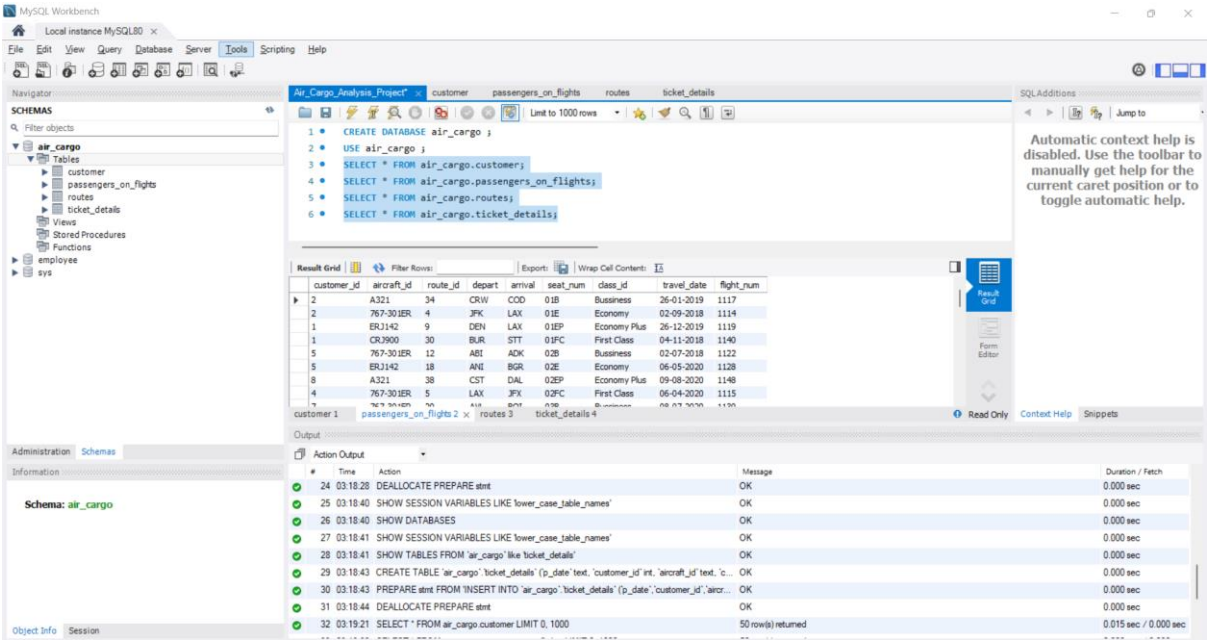
In this project, the *Air Cargo* database was created and analysed through a series of SQL operations designed to enhance the company's route and customer management. The tasks included:

- Designing an ER diagram and defining primary and foreign key relationships.

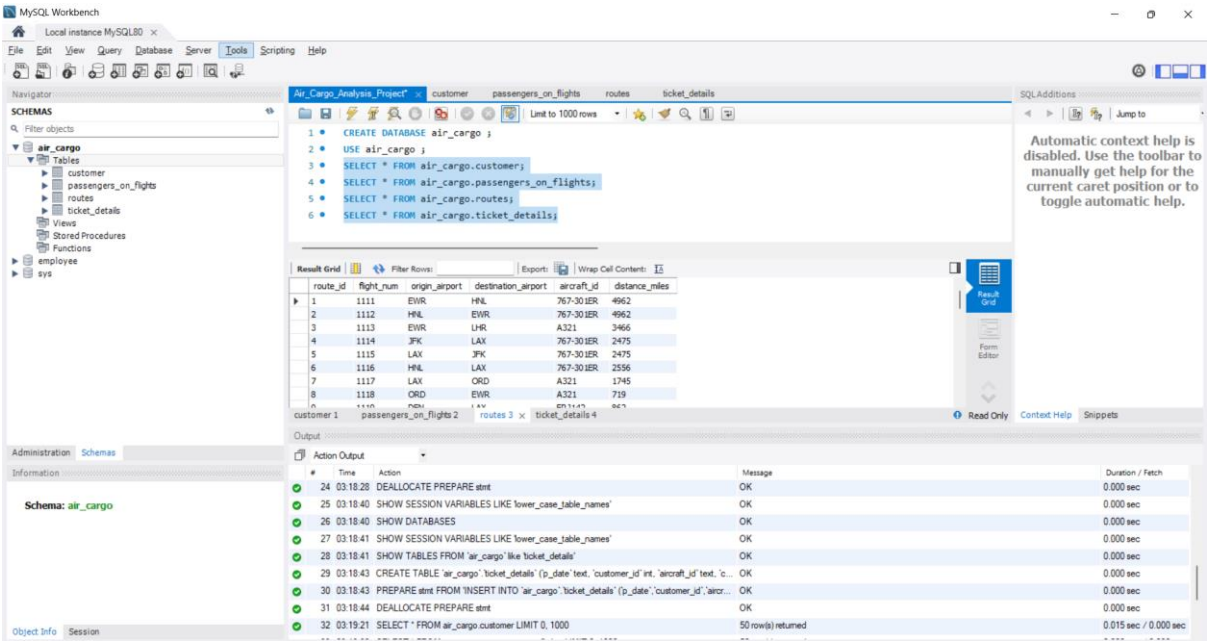
- Writing queries to extract passenger, route, and ticket details.
- Applying conditional logic using IF, CASE, and aggregation functions.
- Creating views, stored procedures, and stored functions for automated and reusable operations.
- Using indexing and the EXPLAIN plan to improve query performance.
- Implementing advanced SQL concepts such as ROLLUP, WINDOW FUNCTIONS, and runtime parameterised procedures.

Through these implementations, the project demonstrates practical applications of SQL in database design, performance optimisation, and analytical reporting — enabling more efficient decision-making for route planning, customer offers, and revenue tracking in the aviation domain.

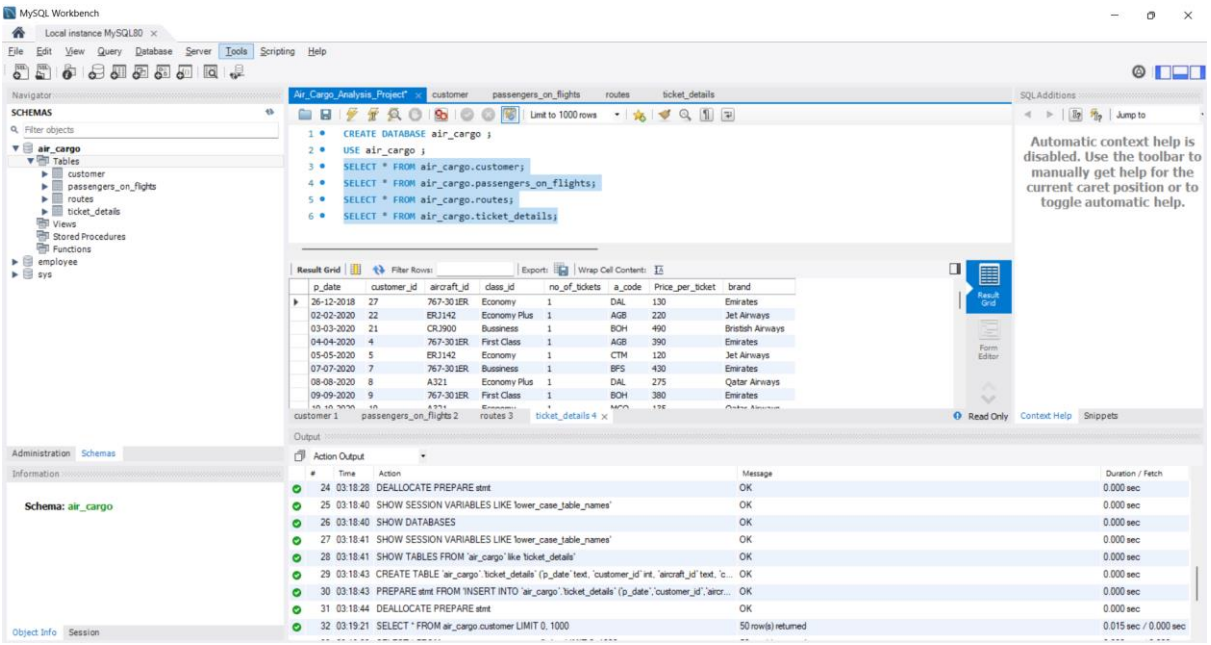
- passengers_on_flights



- routes



- ticket_details



Following operations should be performed:

1. Create an ER diagram for the given airlines database.

Changed the data types of columns and defined primary keys during the ER diagram creation phase (these can also be modified later directly in SQL if needed) :

customer - Table

Table Name: customer Schema: air_cargo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
customer_id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
first_name	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
last_name	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
date_of_birth	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL

routes - Table

Table Name: routes Schema: air_cargo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
route_id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
flight_num	INT	☐	☐	☐	☐	☐	☐	☐	☐	
origin_airport	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
destination_airport	DECIMAL(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

passengers_on_flights - Table

Table Name: passengers_on_flights Schema: air_cargo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
travel_date	DATE	☑	☑	☐	☐	☐	☐	☐	☐	
flight_num	INT	☑	☑	☐	☐	☐	☐	☐	☐	
passengers_on_flightscol	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

ticket_details - Table

Table Name: ticket_details Schema: air_cargo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
p_date	DATE	☑	☑	☐	☐	☐	☐	☐	☐	
customer_id	INT	☑	☑	☐	☐	☐	☐	☐	☐	
aircraft_id	VARCHAR(20)	☑	☑	☐	☐	☐	☐	☐	☐	
class_id	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Defined Foreign keys:

passengers_on_flights - Table

Table Name: passengers_on_flights Schema: air_cargo

Foreign Key Name	Referenced Table
fk_passenger_customer	`air_cargo`.`customer`
fk_passenger_route	`air_cargo`.`routes`

Column	Referenced Column
☐ customer_id	
☐ aircraft_id	
☑ route_id	route_id
☐ depart	
☐ arrival	
☐ seat_num	
☐ class_id	
☐ travel_date	
☐ flight_num	
☐ passengers_on_...	

Foreign Key Options

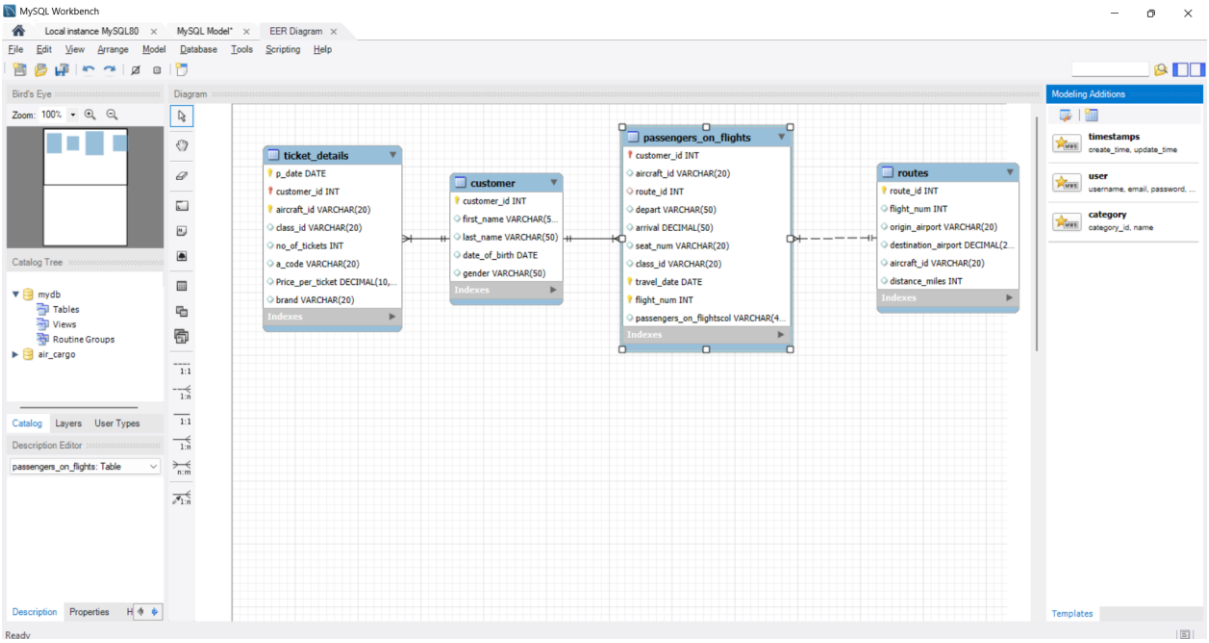
On Update: NO ACTION

On Delete: NO ACTION

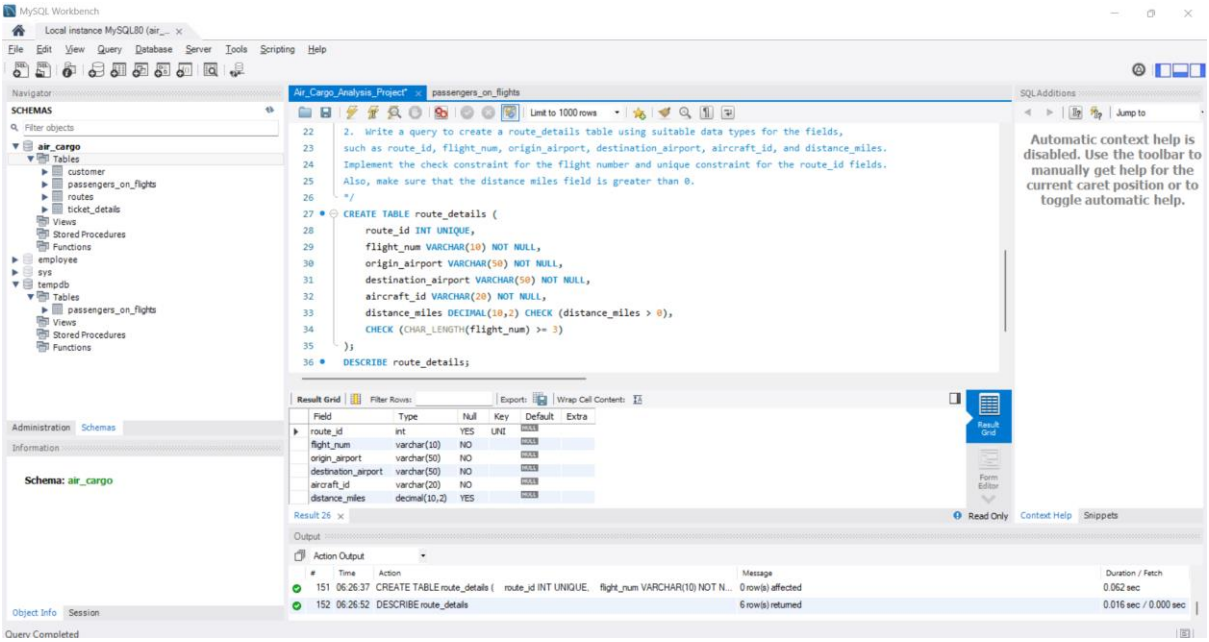
☐ Skip in SQL generation

Foreign Key Comment

Final ER Diagram:



2. Write a query to create a **route_details** table using suitable data types for the fields, such as **route_id**, **flight_num**, **origin_airport**, **destination_airport, aircraft_id**, and **distance_miles**. Implement the check constraint for the flight number and unique constraint for the **route_id** fields. Also, make sure that the distance miles field is greater than 0.

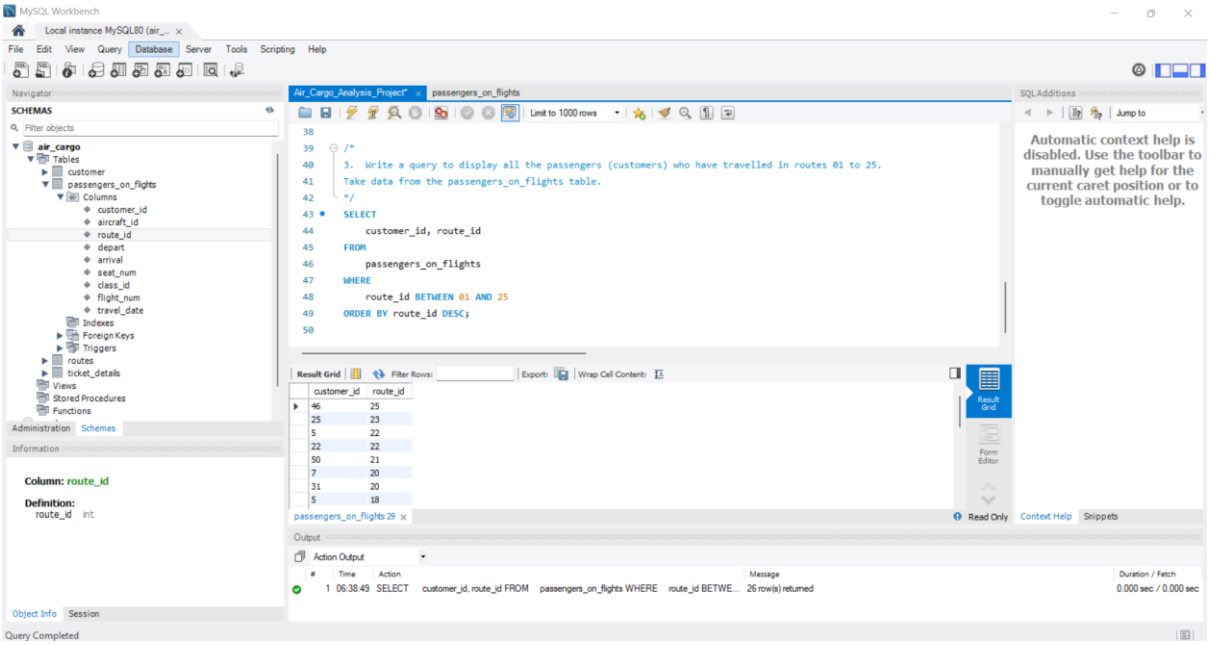


Note:

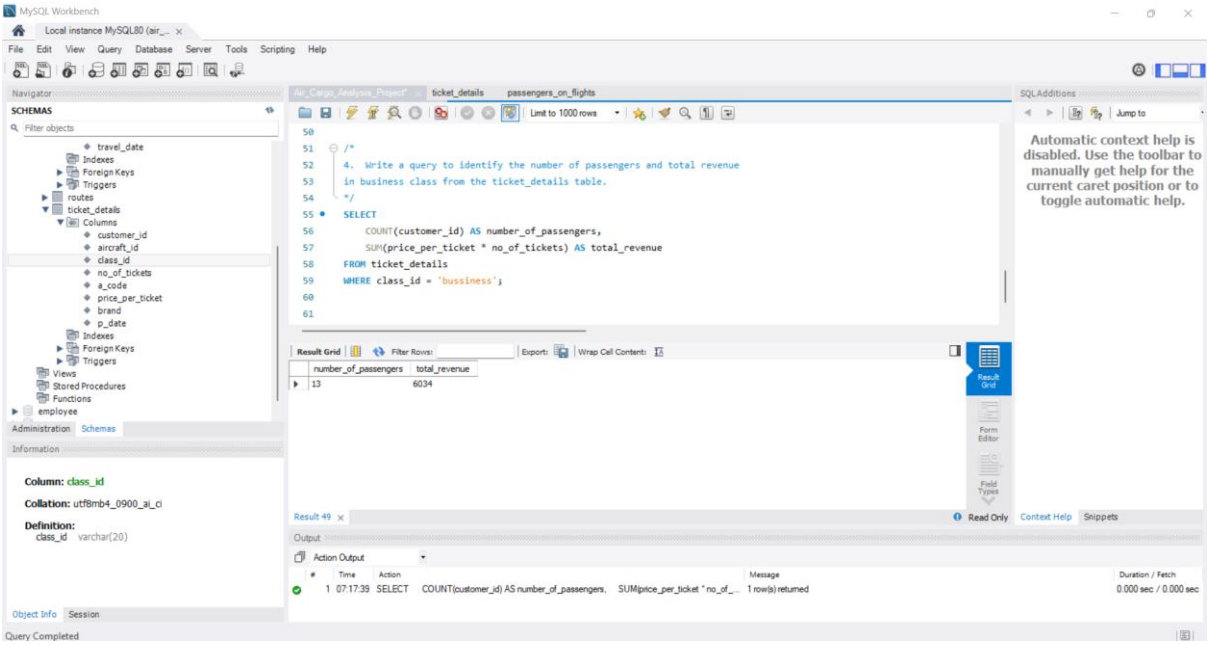
The route_id field has been defined with a **UNIQUE constraint** as per the problem statement.

Alternatively, it could also be defined as a **PRIMARY KEY**, which would inherently enforce uniqueness and not allow NULL values at the same time.

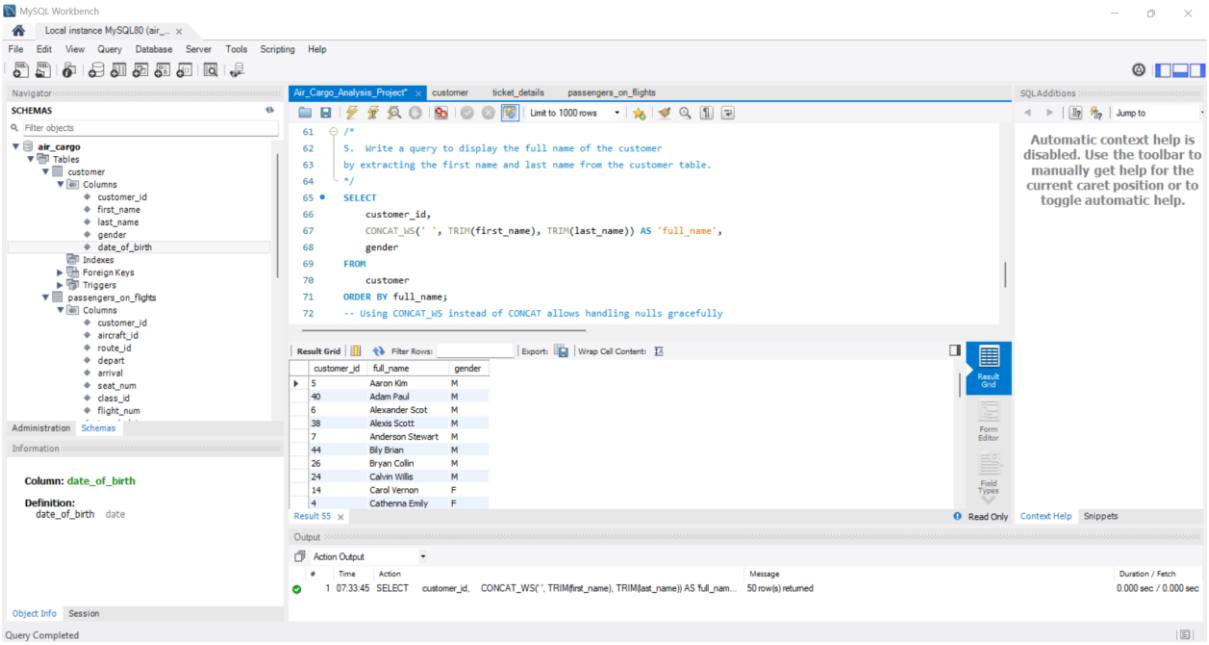
3. Write a query to display all the passengers (customers) who have travelled in routes 01 to 25. Take data from the passengers_on_flights table.



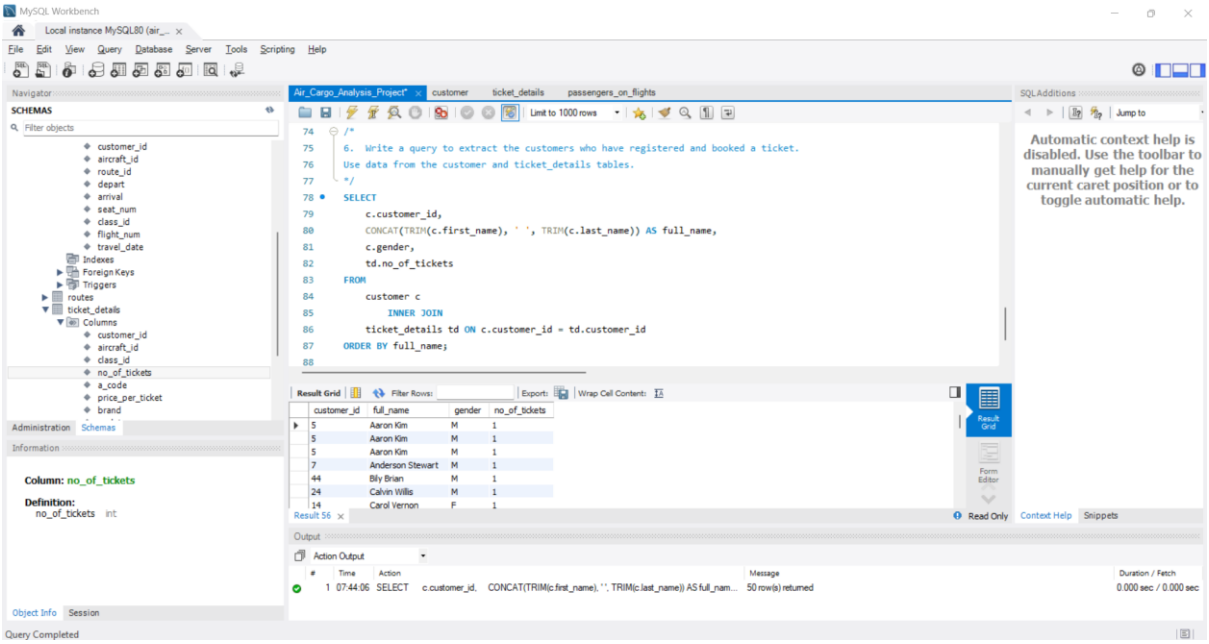
4. Write a query to identify the number of passengers and total revenue in business class from the ticket_details table.



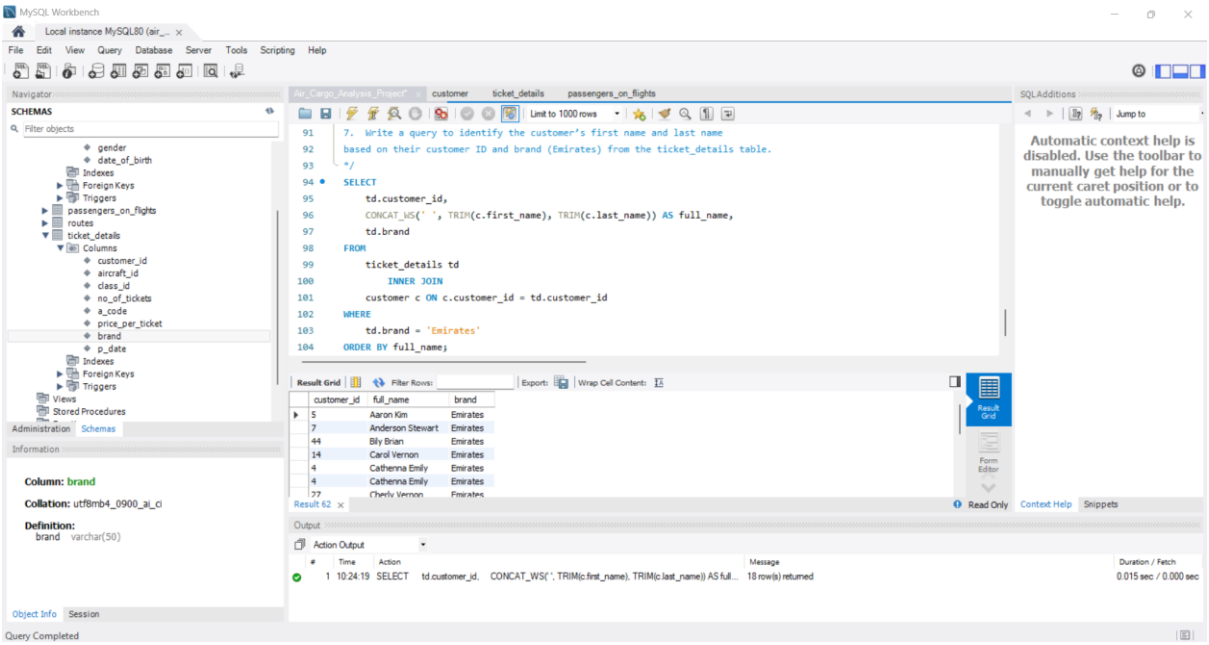
5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.



6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket_details tables.

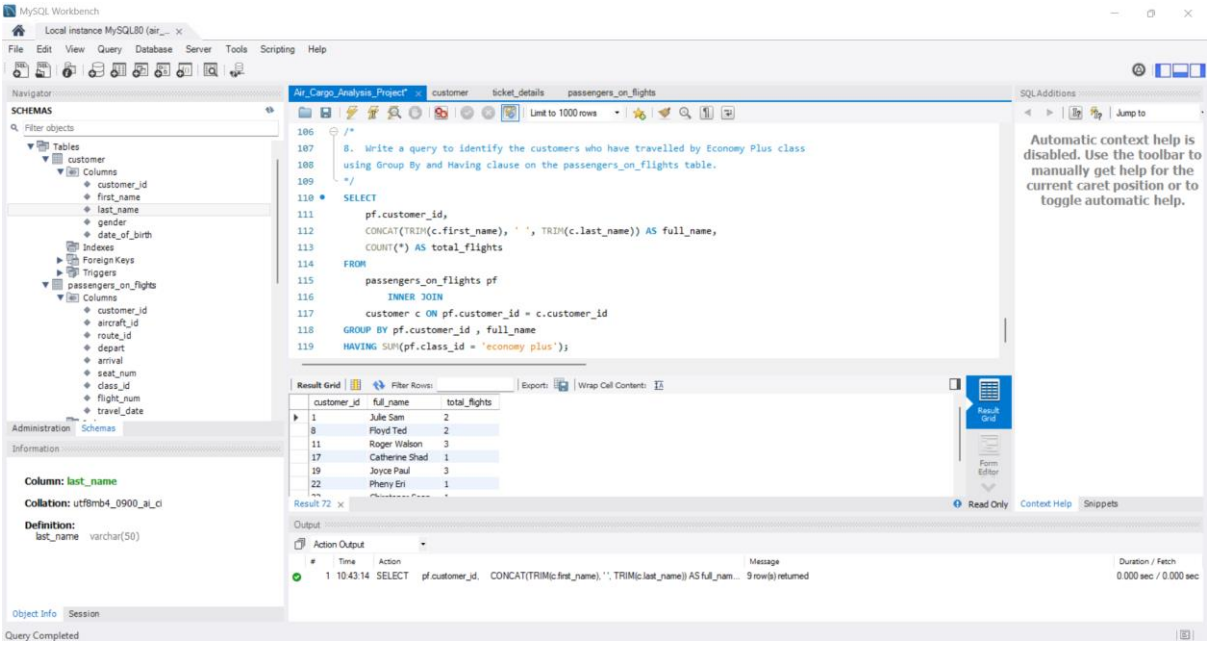


7. Write a query to identify the customer's first name and last name based on their customer ID and brand (Emirates) from the ticket_details table.

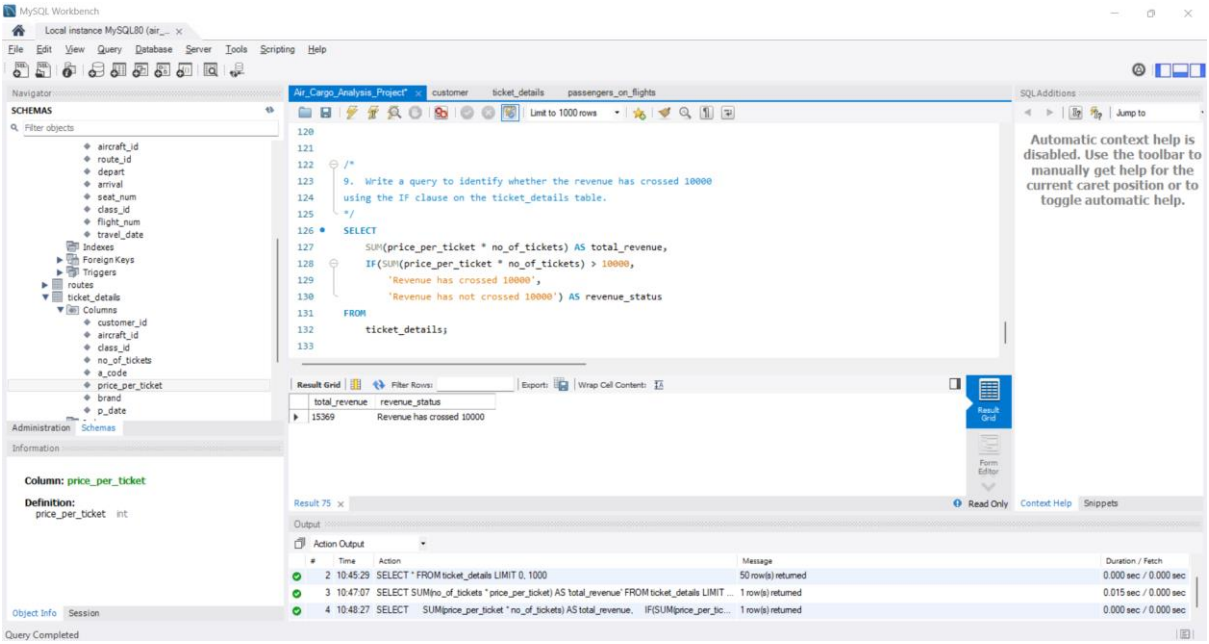


.

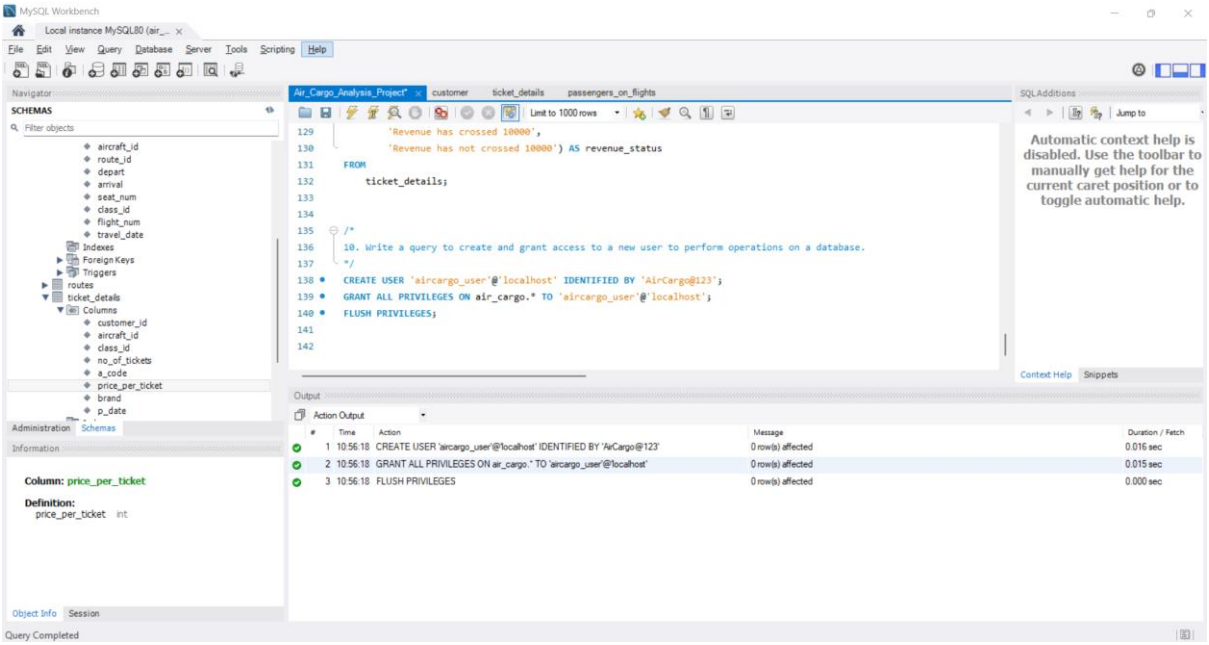
8. Write a query to identify the customers who have travelled by *Economy Plus* class using Group By and Having clause on the passengers_on_flights table.



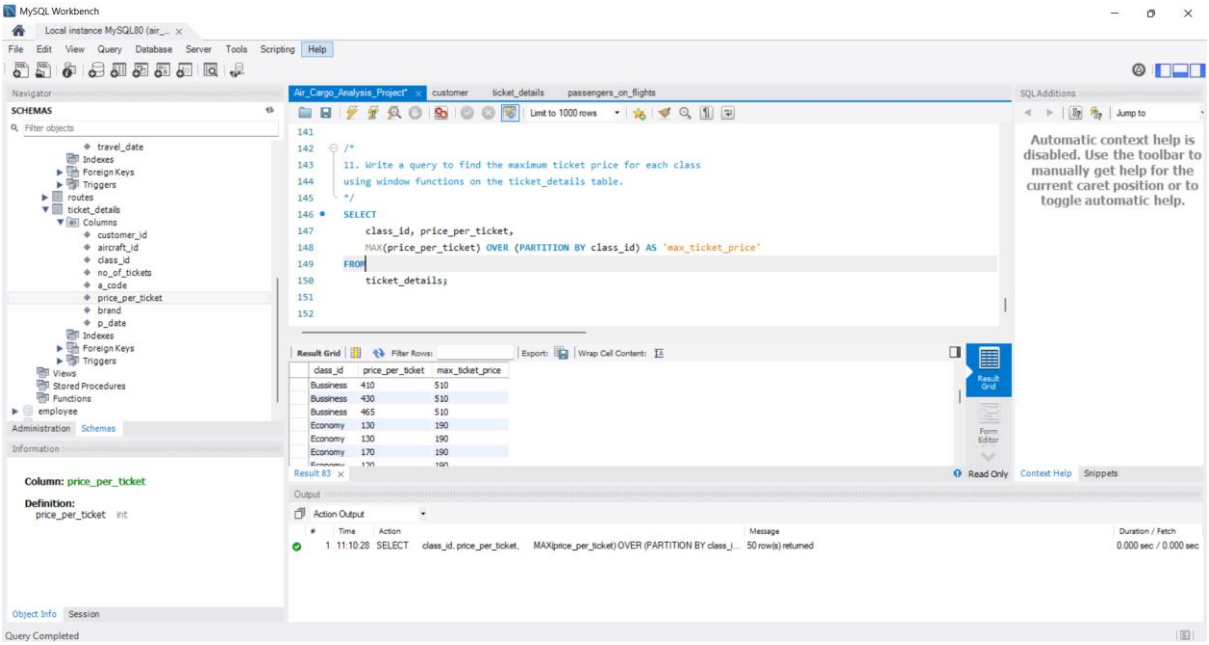
9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.



10. Write a query to create and grant access to a new user to perform operations on a database.

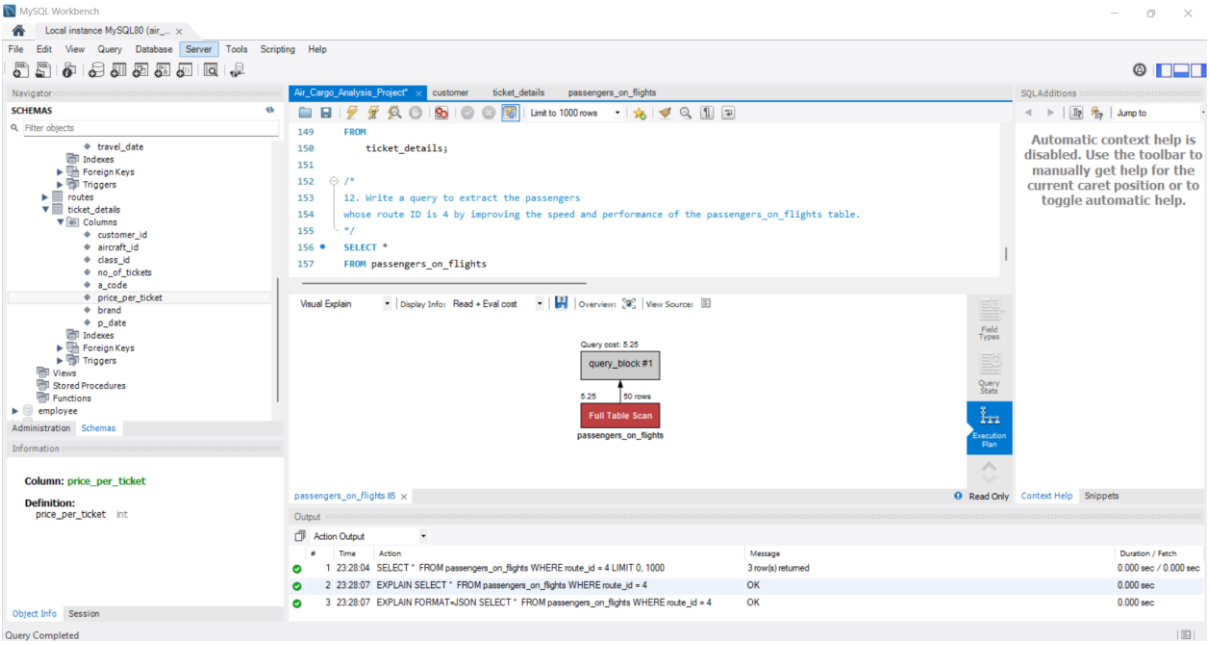


11. Write a query to find the maximum ticket price for each class using window functions on the ticket_details table.

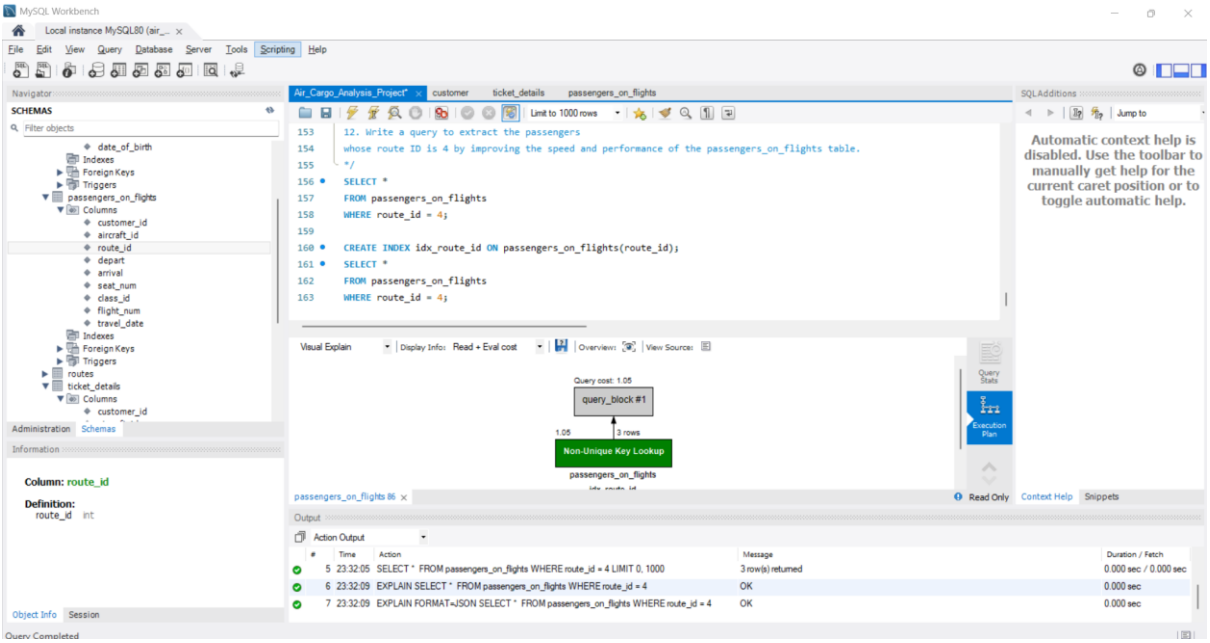


12. Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers_on_flights table.

Performance before indexing:



Performance before indexing:

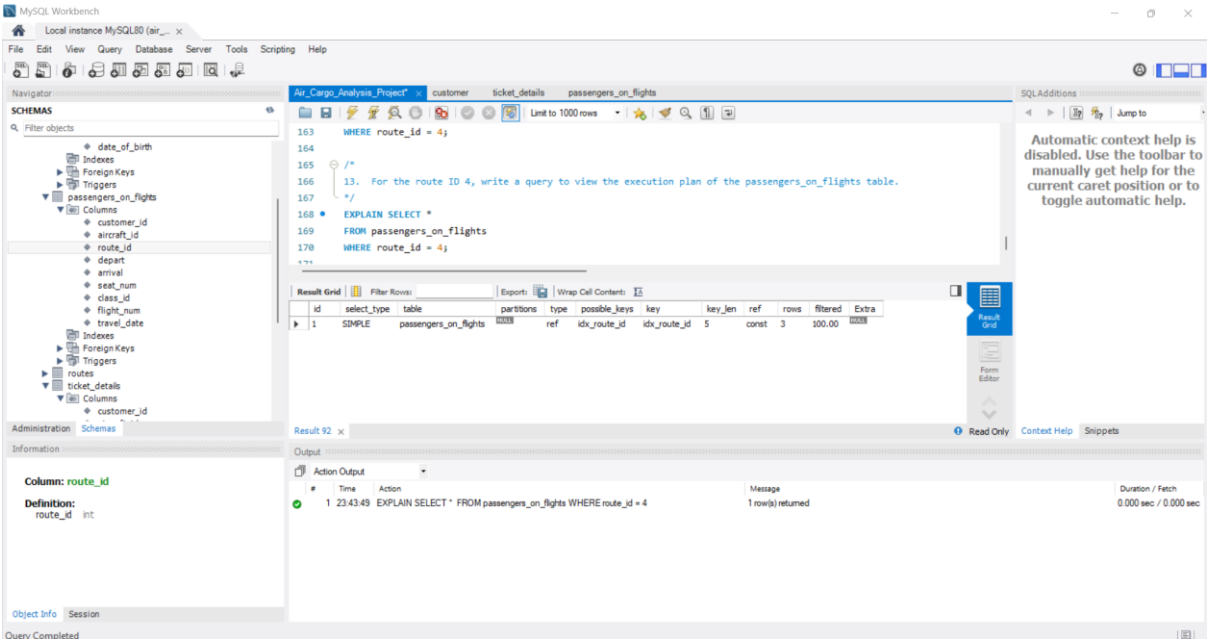


Observation:

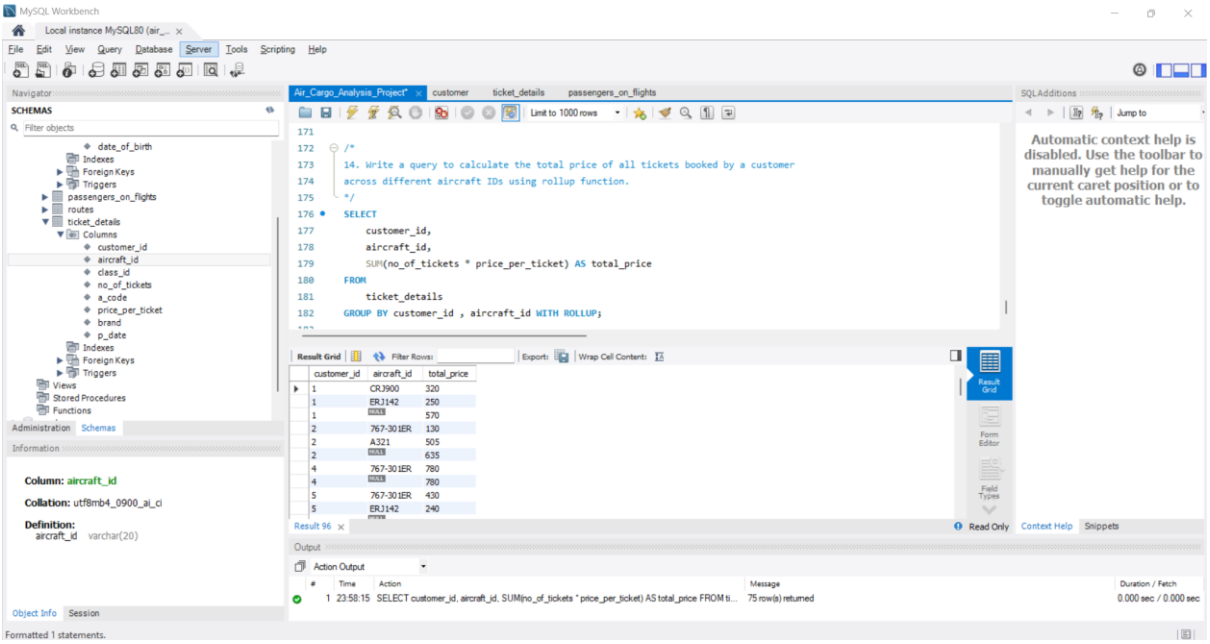
Before indexing, the query to extract passengers whose route_id = 4 performed a **full table scan**, resulting in a higher execution cost (**5.25**) and slower performance as the entire dataset had to be scanned sequentially.

After creating an **index** on the route_id column, the execution plan showed that the query used the index which significantly **reduced the number of rows scanned** and improved the overall query execution speed (1.05).

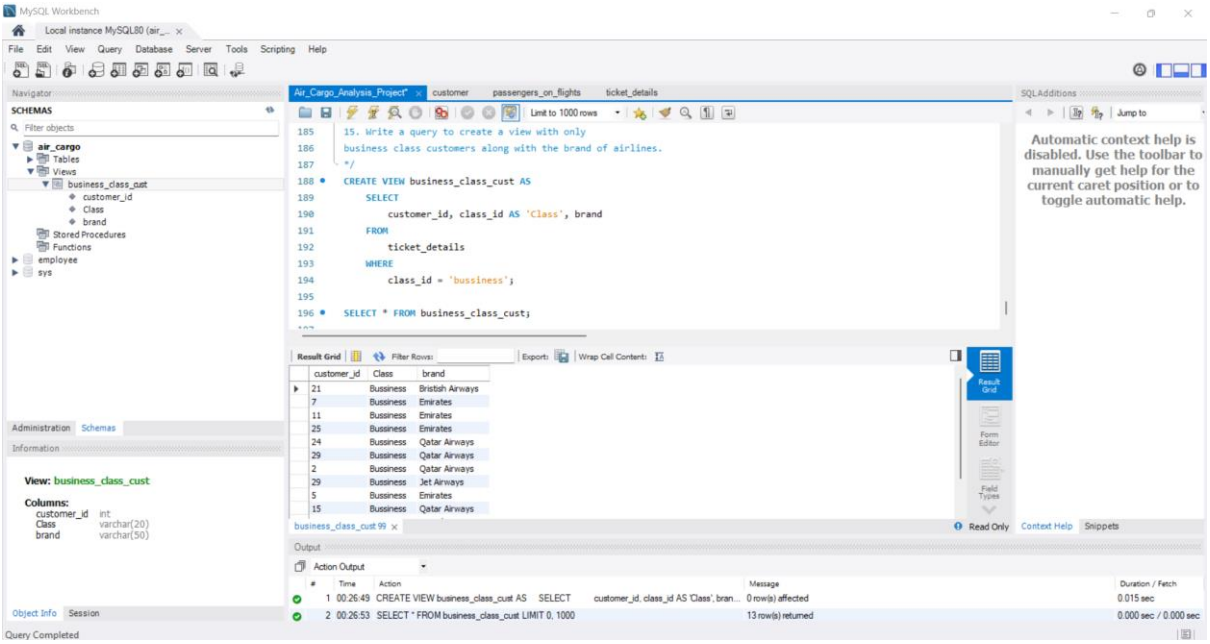
13. For the route ID 4, write a query to view the execution plan of the passengers_on_flights table.



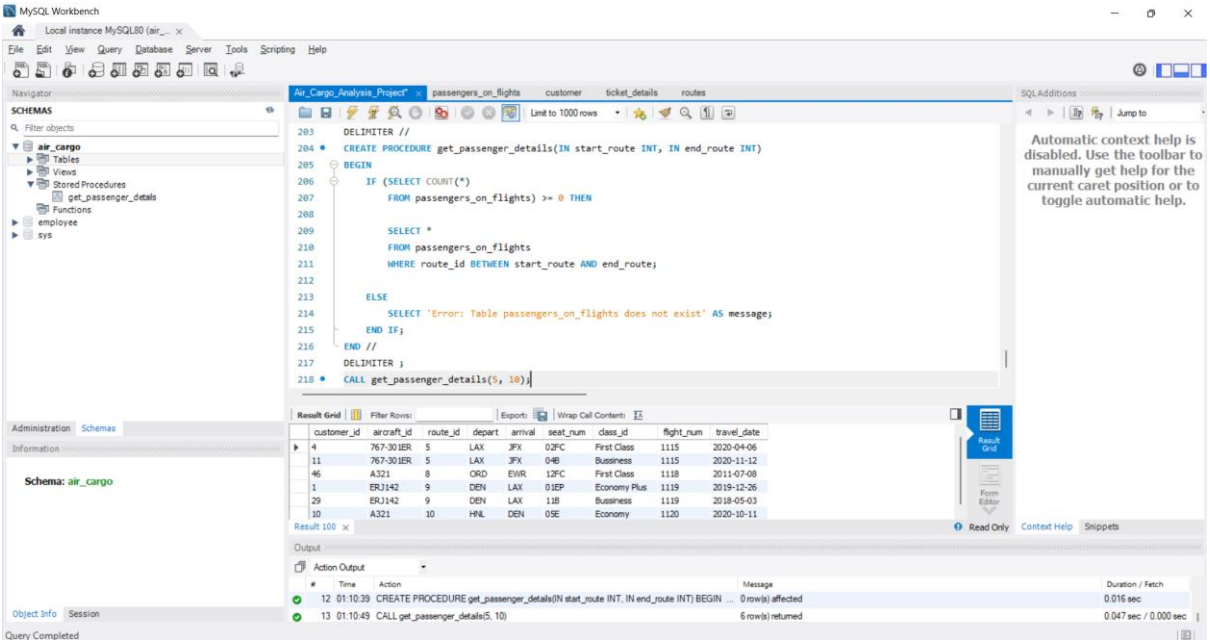
14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.



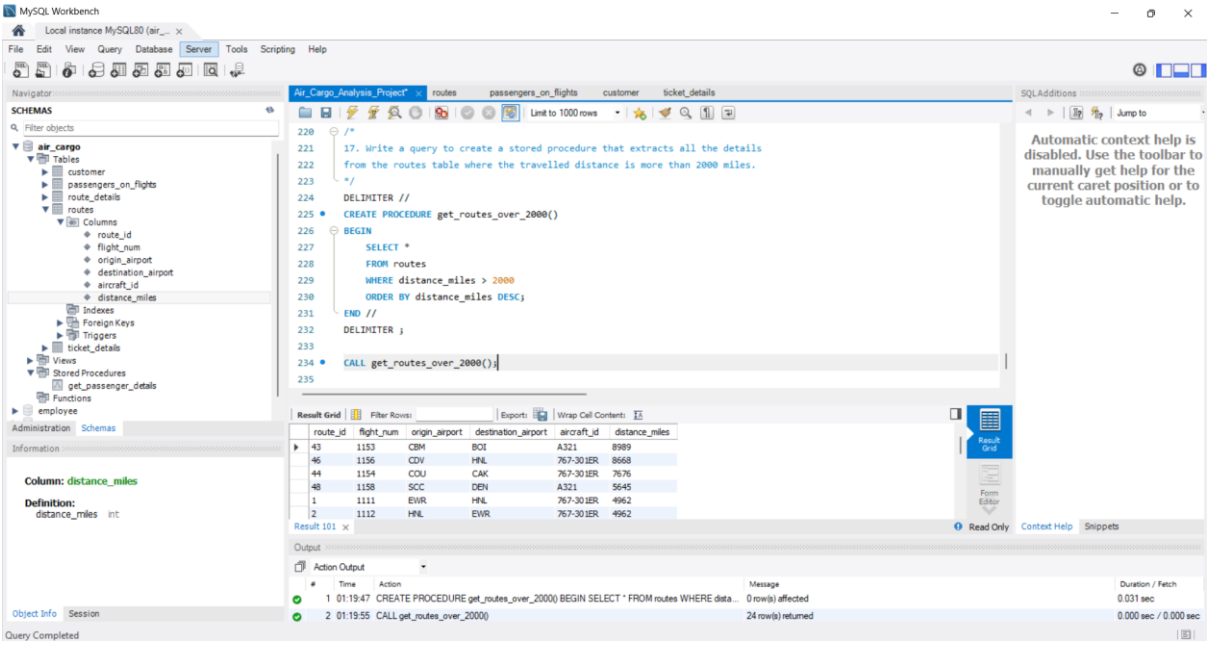
15. Write a query to create a view with only business class customers along with the brand of airlines.



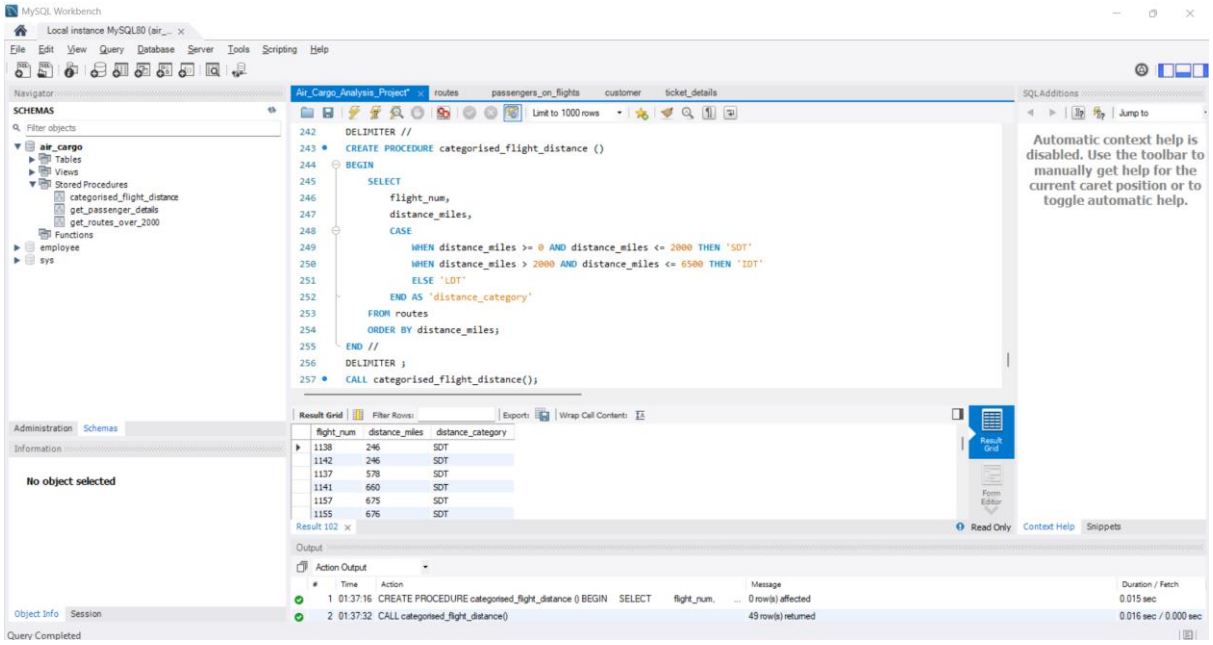
16. Write a query to create a stored procedure to get the details of all passengers flying between a range of routes defined in run time. Also, return an error message if the table doesn't exist.



17. Write a query to create a stored procedure that extracts all the details from the routes table where the travelled distance is more than 2000 miles.



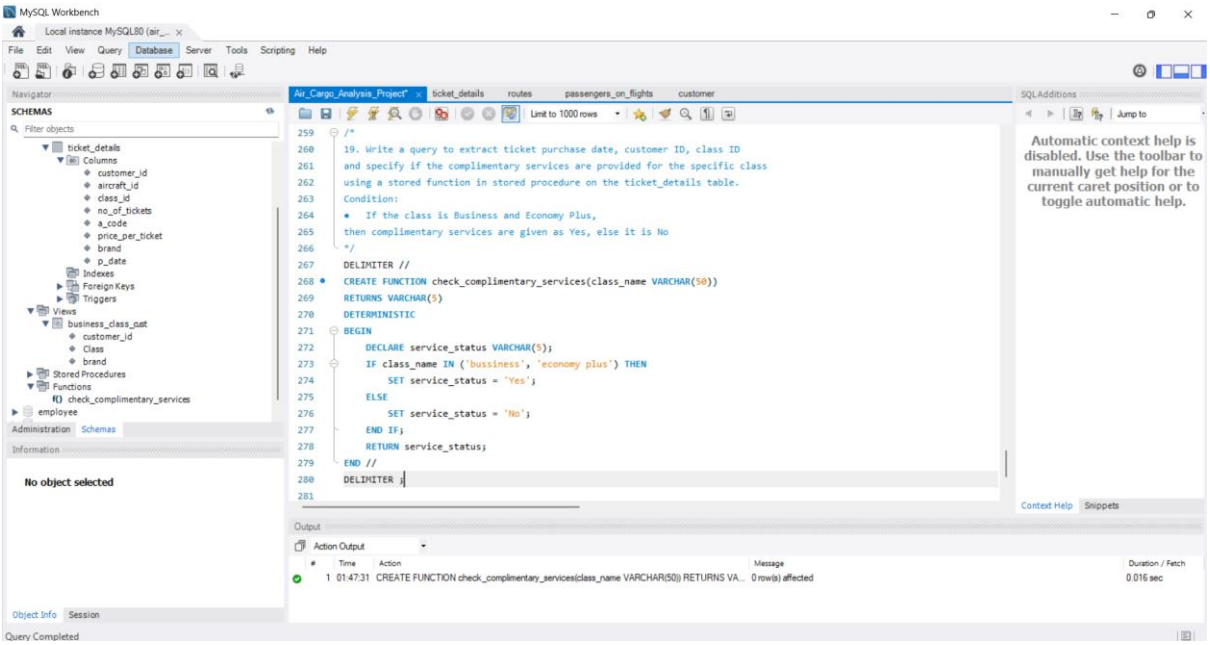
18. Write a query to create a stored procedure that groups the distance travelled by each flight into three categories. The categories are, short distance travel (SDT) for >=0 AND <= 2000 miles, intermediate distance travel (IDT) for >2000 AND <=6500, and long-distance travel (LDT) for >6500.



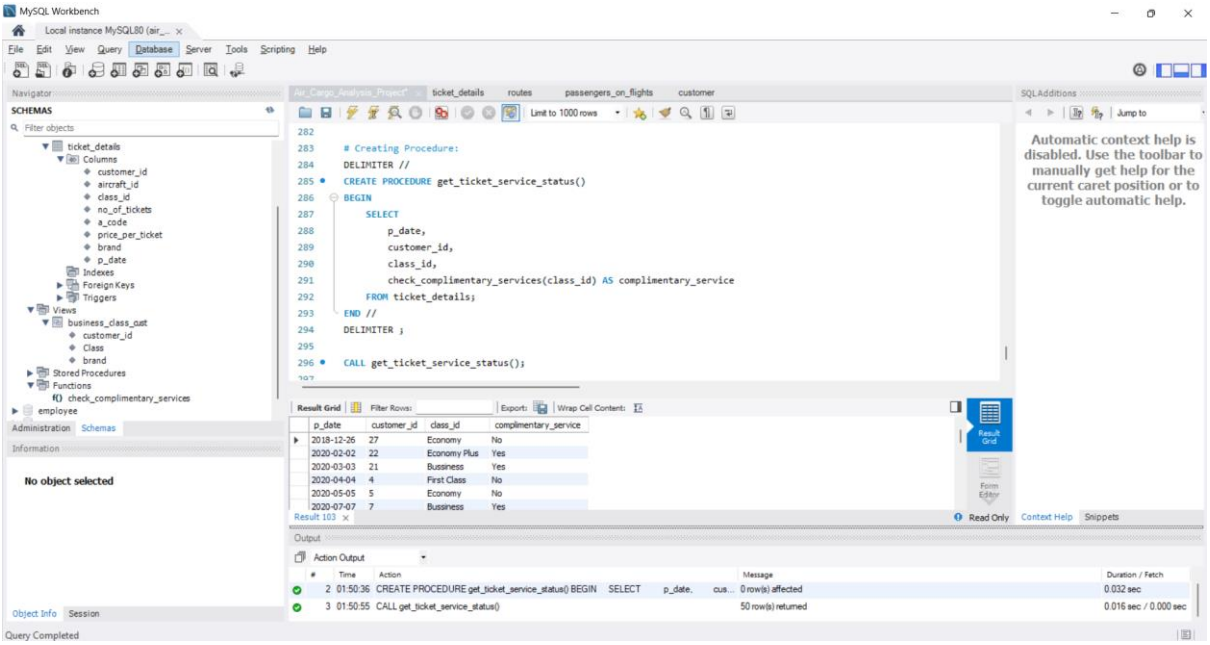
19. Write a query to extract ticket purchase date, customer ID, class ID and specify if the complimentary services are provided for the specific class using a stored function in stored procedure on the ticket_details table. Condition:

- If the class is *Business* and *Economy Plus,* then complimentary services are given as *Yes,* else it is *No*

Creating Function:



Creating PROCEDURE:



20. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

NOTE: Cursor operations were not covered during Live Class, hence this query was not implemented

Conclusion:

In this project, the *Air Cargo* database was created and analysed through a series of SQL operations designed to enhance the company's route and customer management. The tasks included:

- Designing an ER diagram and defining primary and foreign key relationships.
- Writing queries to extract passenger, route, and ticket details.
- Applying conditional logic using IF, CASE, and aggregation functions.
- Creating views, stored procedures, and stored functions for automated and reusable operations.
- Using indexing and the EXPLAIN plan to improve query performance.
- Implementing advanced SQL concepts such as ROLLUP, WINDOW FUNCTIONS, and runtime parameterised procedures.

Through these implementations, the project demonstrates practical applications of SQL in database design, performance optimisation, and analytical reporting — enabling more efficient decision-making for route planning, customer offers, and revenue tracking in the aviation domain.