

FlyRes User Manual



Payton Wiscovitch
Informatics Department Spring 23B
Graduate Project

Table of Contents

Introduction	3
phpMyAdmin – User Interface	3
GUI Layout	4
Login Screen.....	4
Basic GUI Layout	5
FlyRes Database Main Page	6
GUI Overview Wrap-up.....	8
Flight Information	9
Overview.....	9
Flight Table.....	9
View All Flights.....	9
Flights by Airport Code	11
Flights by Airline.....	12
Scheduled_flight Table	13
View All Scheduled Flights	13
View Scheduled Flights by Date	13
Customers, Passengers, and Accounts	14
Passenger Table	14
View All Passengers	14
Adding a Passenger.....	15
Edit Passenger Information.....	16
Deleting a Passenger.....	17
Creating an Account.....	18
Linking an Account.....	20
Employees.....	21
Employee Table.....	21
View All Employees.....	21
Adding a New Employee	22
Editing Employee Information	23
Deleting an Employee	25
Reservations	25

Reservation Table	25
View All Reservations.....	25
View Reservation by [parameter]	26
Creating a Reservation.....	27
System Reports	40
Passenger Mailing List.....	40
Customer Email List	42
Employee Mailing List	43
Employee Email List	45
Revenue by Customer	46
Revenue by Employee.....	48
Revenue by Month.....	50

Introduction

The following document is to serve as a user level document for the database administrator/manager and customer representatives to perform user level transactions based on the project requirements. This is a companion document to the overall project technical specifications document. This document will start by walking the user through the phpMyAdmin user interface and then explain how to perform necessary tasks. Based on the project requirements, users need to be able to perform the following tasks/transactions:

- List all flights
- List flights by airport
- View reservations
- View reservations by customer
- Create a reservation
- Revenue report by month
- Revenue generated by customer
- Revenue generated by employee (customer representative)
- Add, edit, and delete passenger/passenger information
- Create a customer mailing list
- View passenger(s) reserved for a specific flight
- Make suggestions for flights based on customer reservations

phpMyAdmin – User Interface

PhpMyAdmin was chosen for this project for the ease of use and the GUI provided for non-technical users. The

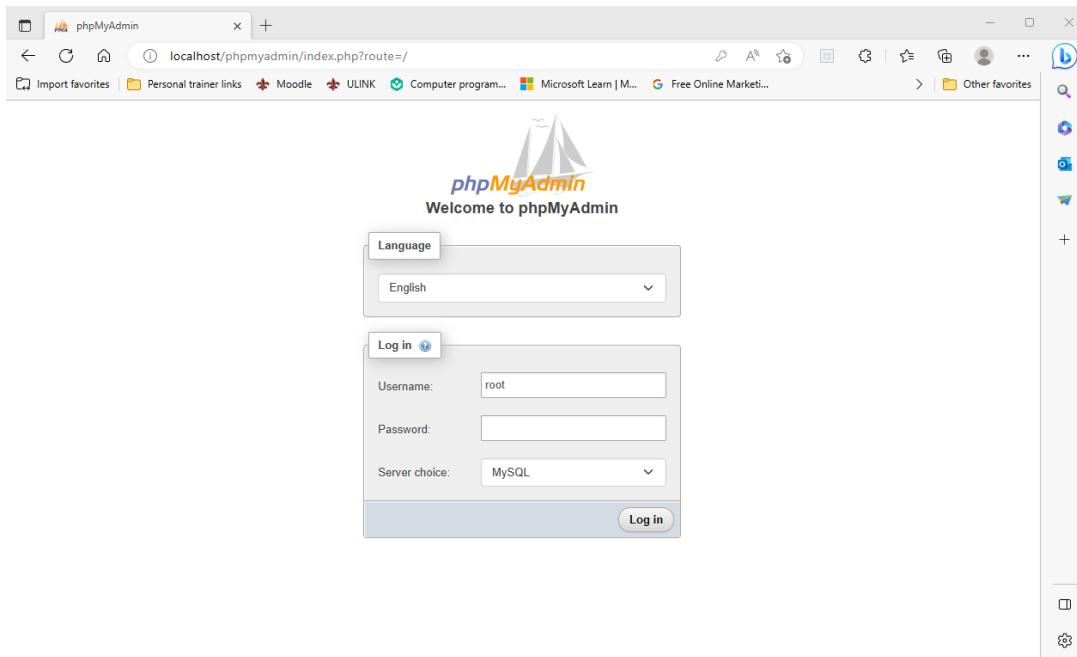
“phpMyAdmin is a free software tool written in [PHP](#), intended to handle the administration of [MySQL](#) over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.”[1]

interface allows for both traditional use of MySQL commands and queries, as well as simple point and click administration. The program can be downloaded directly from www.phpmyadmin.net.

GUI Layout

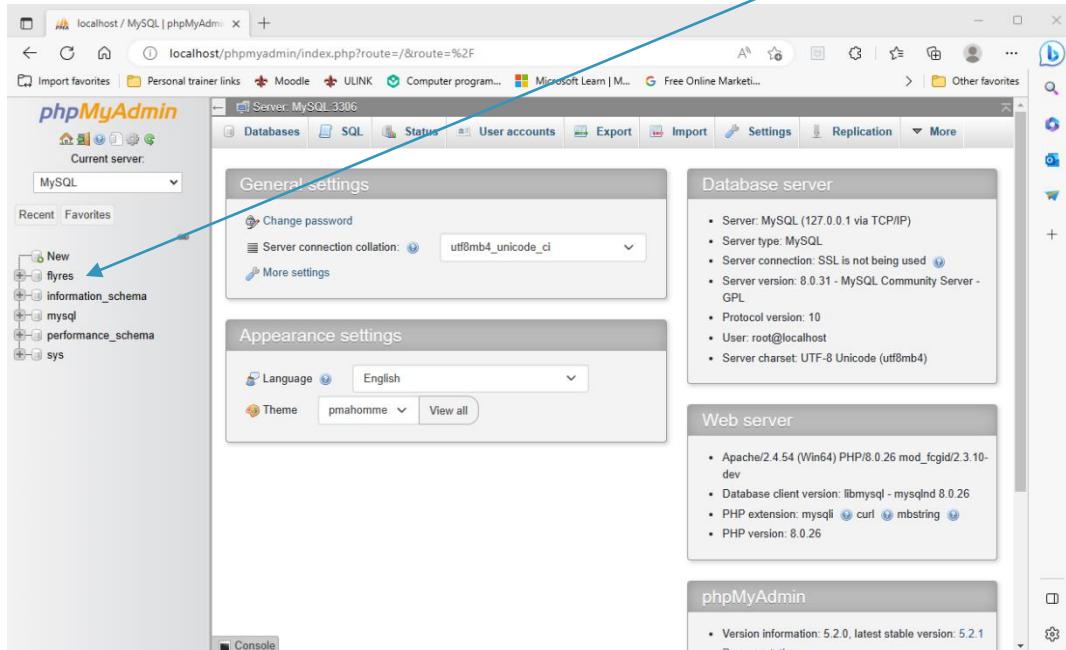
Login Screen

Once the phpMyAdmin software is downloaded and opened a login screen appears. An administrator user name and password for the FlyRes database has been established. FlyRes management will login using the user name: manager and password: flyresmanager. You will notice that it opens as a part of the user's web browser as it is a web-based application.



Basic GUI Layout

This is the main screen of the phpMyAdmin GUI. On the left side panel are all the databases listed. The database we will be working with is flyres. Across the top toolbar the administrator will see the option to view all databases, enter SQL commands, view server status, add/remove/edit user accounts, export information, import information, view and edit settings, and more. The first thing we want to do is select the flyres database so that we will be working within that database.



FlyRes Database Main Page

After opening the FlyRes database, a main page screen will appear showing all tables within the database.

The screenshot shows the phpMyAdmin interface for the FlyRes database. The left sidebar is labeled 'Left side panel' and lists the database structure under 'flyres'. The main area, labeled 'Main page', shows a table of 8 tables with their respective details. A modal window for creating a new table is open in the center.

All tables are listed in the left side panel under the flyres database as well as the center console. In the center of the screen the tables are listed along with the available functions such as browse, view table structure, table search, insert information into the table, empty the table, or drop/delete the table. The main functions the administrator will use are browse, search, and insert.

Browse

The browse function performs a simple SELECT all FROM <table> command. Which returns all entries for a given table. This is a great function to use to see all the information available for any given table. For example, if we wanted to see all the employees in the employee table, we could simply select browse for the employee table.

The screenshot shows the phpMyAdmin interface for the 'employee' table. The table has columns: employee_id, first_name, last_name, address, city, state, zip, phone, email, hire_date, and hourly_rate. Data rows are shown for John Wick, Michael Totaro, Wonder Woman, and Bob Builder. Action buttons for Edit, Copy, and Delete are available for each row.

Search

The search function allows us to search a table for a specific record using the available variables. For example, if we wanted to search for a specific employee by first and last name, we could click search under the employee table and then search this information.

1. Search screen for employee table: enter employee first name and last name to search and click 'Go'.

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. Under the 'Tables' section, the 'employee' table is chosen. In the main pane, there is a search interface titled 'Table search'. It contains a form with fields for various columns: 'employee_id', 'first_name', 'last_name', 'address', 'city', 'state', 'zip', 'phone', 'email', 'hire_date', and 'hourly_rate'. Each field has a dropdown menu for operators like '=', 'LIKE', etc., and a text input field. Below the form is a button labeled 'Extra options'.

2. Search results are generated and displayed as follows.

The screenshot shows the MySQL Workbench interface after a search has been performed. The main pane displays the search results for the 'employee' table. A green banner at the top indicates 'Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)'. Below the banner, the SQL query is shown: 'SELECT * FROM `employee` WHERE `first_name` LIKE 'John' AND `last_name` LIKE 'Wick''. The results table shows one row: employee_id 11123333, first_name John, last_name Wick, address 337 Doggy Lane, city Lafayette, state Louisiana, zip 70506, phone 3375551234, email john.wick@flyres.com, hire_date 2020-04-01, and hourly_rate 25.00. There are buttons for 'Edit', 'Copy', 'Delete', 'Export', and 'Import' below the table. At the bottom, there are buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'.

3. Based on the purpose of the search the administrator can then use the information, edit the information, or delete the information.

Insert

The last function that will be performed from the main database screen is the insert function. For example, if the administrator needs to add a new employee. They will select insert on the employee table row.

- After selecting insert on the employee table the following input screen will appear.

The screenshot shows the phpMyAdmin interface for inserting data into the 'employee' table. The left sidebar lists various databases and tables, with 'employee' selected. The main area has tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert' (which is active), 'Export', 'Import', 'Privileges', 'Operations', and 'Triggers'. The 'Insert' tab displays a form with fields for each column in the 'employee' table:

Column	Type	Function	Null	Value
employee_id	int		Yes	<input type="text"/>
first_name	varchar(50)		Yes	<input type="text"/>
last_name	varchar(50)		Yes	<input type="text"/>
address	varchar(250)		No	<input type="text"/>
city	varchar(50)		Yes	<input type="text"/>
state	varchar(50)		Yes	<input type="text"/>
zip	int		Yes	<input type="text"/>
phone	varchar(10)		Yes	<input type="text"/>
email	varchar(100)		No	<input type="text"/>
hire_date	date		Yes	<input type="text"/>
hourly_rate	decimal(10,2)		Yes	<input type="text"/>

At the bottom right of the form is a 'Go' button.

- All information is required to be filled out before submitting using the 'Go' button to create a new record. Specifics for adding new employees will be detailed later under the adding/editing/deleting employee section.

GUI Overview Wrap-up

The best way to familiarize yourself with the phpMyAdmin graphical user interface is to follow this guide and explore the interface hands on. The interface is designed with ease of use in mind and combining the user-friendly design with the detailed instructions contained in this manual will help the flyres administrator to correctly perform necessary functions.

Flight Information

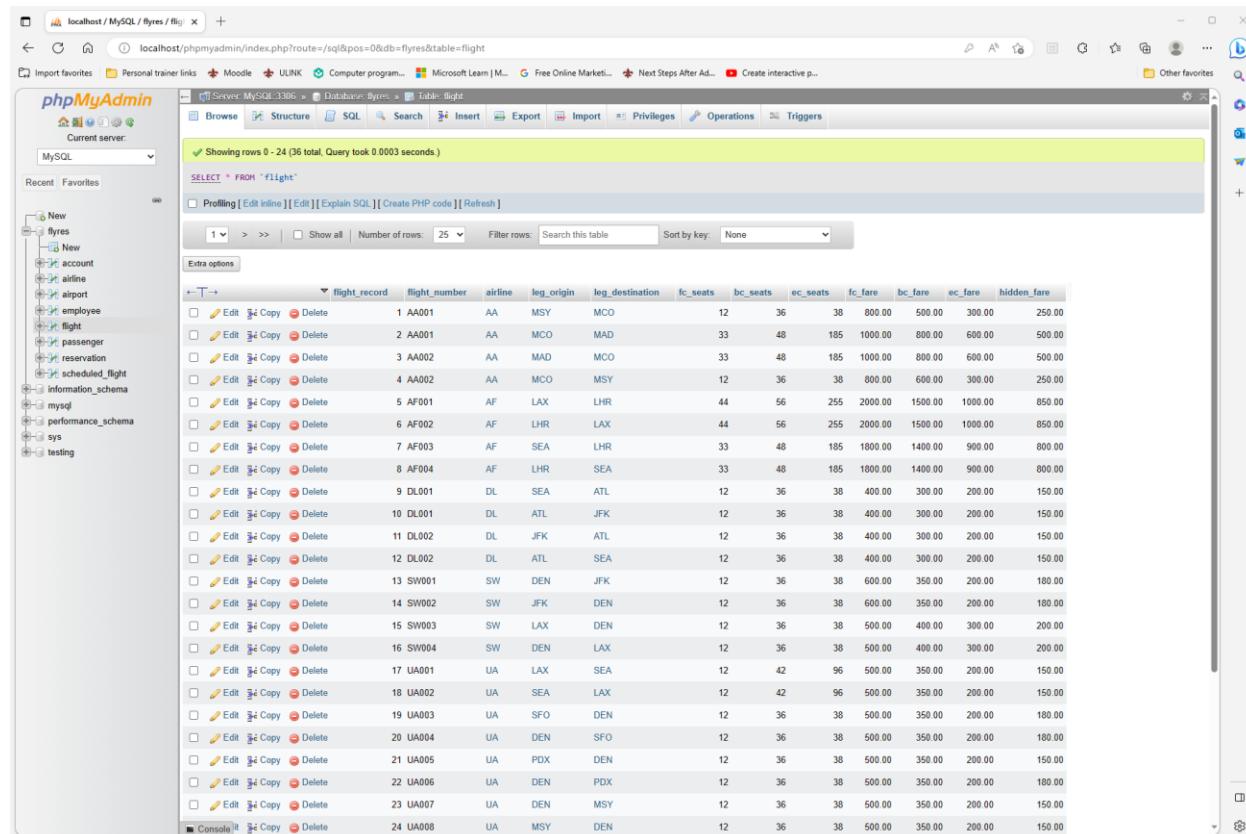
Overview

The following section will explain in detail how to access flight information. This will include how to view all flights, view all scheduled flights, looking up airport codes, viewing all flights by airport code, viewing flights by date, and viewing which passengers are reserved on a scheduled flight. It is important to know that the flight list contains general information about flights as it pertains to the route that the flight operates on and the associated fares for the flight. The scheduled_flight table lists all scheduled flights with dates and times for the flights contained within the flight table. So, the flight table will be used to gather information about routes while the scheduled_flight table is used to search for the information we gathered using the flight table. Flights are referenced using their flight record associated with specified origin and destination.

Flight Table

View All Flights

In order to view all flights, expand/open the flyres database as explained in the GUI introduction section by clicking on flyres in the left-hand side panel of phpMyAdmin. Then click on the flight table listed on the main page. This will perform a SELECT * FROM flight command and return a list of all flights and the corresponding information as shown below.



The screenshot shows the phpMyAdmin interface with the following details:

- Server:** MySQL
- Database:** flyres
- Table:** flight
- Query Results:** Showing rows 0 - 24 (36 total, Query took 0.0003 seconds.)
- SQL Query:** SELECT * FROM `flight`;
- Table Headers:** flight_record, flight_number, airline, leg_origin, leg_destination, fc_seats, bc_seats, ec_seats, fc_fare, bc_fare, ec_fare, hidden_fare
- Data Preview:** A grid of flight records with columns for flight number, airline, origin, destination, and various fare types.

flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
1	AA001	AA	MSY	MCO	12	36	38	800.00	500.00	300.00	250.00
2	AA001	AA	MCO	MAD	33	48	185	1000.00	800.00	600.00	500.00
3	AA002	AA	MAD	MCO	33	48	185	1000.00	800.00	600.00	500.00
4	AA002	AA	MCO	MSY	12	36	38	800.00	600.00	300.00	250.00
5	AF001	AF	LAX	LHR	44	56	255	2000.00	1500.00	1000.00	850.00
6	AF002	AF	LHR	LAX	44	56	255	2000.00	1500.00	1000.00	850.00
7	AF003	AF	SEA	LHR	33	48	185	1800.00	1400.00	900.00	800.00
8	AF004	AF	LHR	SEA	33	48	185	1800.00	1400.00	900.00	800.00
9	DL001	DL	SEA	ATL	12	36	38	400.00	300.00	200.00	150.00
10	DL001	DL	ATL	JFK	12	36	38	400.00	300.00	200.00	150.00
11	DL002	DL	JFK	ATL	12	36	38	400.00	300.00	200.00	150.00
12	DL002	DL	ATL	SEA	12	36	38	400.00	300.00	200.00	150.00
13	SW001	SW	DEN	JFK	12	36	38	600.00	350.00	200.00	180.00
14	SW002	SW	JFK	DEN	12	36	38	600.00	350.00	200.00	180.00
15	SW003	SW	LAX	DEN	12	36	38	500.00	400.00	300.00	200.00
16	SW004	SW	DEN	LAX	12	36	38	500.00	400.00	300.00	200.00
17	UA001	UA	LAX	SEA	12	42	96	500.00	350.00	200.00	150.00
18	UA002	UA	SEA	LAX	12	42	96	500.00	350.00	200.00	150.00
19	UA003	UA	SFO	DEN	12	36	38	500.00	350.00	200.00	180.00
20	UA004	UA	DEN	SFO	12	36	38	500.00	350.00	200.00	180.00
21	UA005	UA	PDX	DEN	12	36	38	500.00	350.00	200.00	150.00
22	UA006	UA	DEN	PDX	12	36	38	500.00	350.00	200.00	180.00
23	UA007	UA	DEN	MSY	12	36	38	500.00	350.00	200.00	150.00
24	UA008	UA	MSY	DEN	12	36	38	500.00	350.00	200.00	150.00

All flights have a flight number, an airline, origin, destination, number of seats in each class, and the fares associated with each class as well as a hidden fare which is the lowest economy class fare that the airline would accept in a passenger bid. Airline, leg_origin, and leg_destination are linked to the airline and airport tables respectively. To get further information on the airline or airport, simply click the blue text. For example, flight AA001 is operated by airline AA, if we click the blue text AA, we get the following result.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. The 'airline' table is selected. A query has been run: `SELECT * FROM 'flyres'.'airline' WHERE 'airline_id' = 'AA'`. The results show one row: `airline_id: AA, airline_name: American Airlines`. The 'American Airlines' link is highlighted in blue, indicating it can be clicked for more information.

The result shows that flight AA001 is operated by airline AA = American Airlines. To return to the previous screen and view all flights, simply click the back button on the web browser.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. The 'flight' table is selected. A query has been run: `SELECT * FROM 'flyres'.'flight'`. The results show 24 rows of flight data. The first few rows are:

	flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
1	AA001	AA	MSY	MCO		12	36	38	800.00	500.00	300.00	250.00
2	AA001	AA	MCO	MAD		33	48	185	1000.00	800.00	600.00	500.00
3	AA002	AA	MAD	MCO		33	48	185	1000.00	800.00	600.00	500.00
4	AA002	AA	MCO	MSY		12	36	38	800.00	600.00	300.00	250.00
5	AF001	AF	LAX	LHR		44	56	255	2000.00	1500.00	1000.00	850.00
6	AF002	AF	LHR	LAX		44	56	255	2000.00	1500.00	1000.00	850.00
7	AF003	AF	SEA	LHR		33	48	185	1800.00	1400.00	900.00	800.00
8	AF004	AF	LHR	SEA		33	48	185	1800.00	1400.00	900.00	800.00
9	DL001	DL	SEA	ATL		12	36	38	400.00	300.00	200.00	150.00
10	DL001	DL	ATL	JFK		12	36	38	400.00	300.00	200.00	150.00
11	DL002	DL	JFK	ATL		12	36	38	400.00	300.00	200.00	150.00
12	DL002	DL	ATL	SEA		12	36	38	400.00	300.00	200.00	150.00
13	SW001	SW	DEN	JFK		12	36	38	600.00	350.00	200.00	180.00
14	SW002	SW	JFK	DEN		12	36	38	600.00	350.00	200.00	180.00
15	SW003	SW	LAX	DEN		12	36	38	500.00	400.00	300.00	200.00
16	SW004	SW	DEN	LAX		12	36	38	500.00	400.00	300.00	200.00
17	UA001	UA	LAX	SEA		12	42	96	500.00	350.00	200.00	150.00
18	UA002	UA	SEA	LAX		12	42	96	500.00	350.00	200.00	150.00
19	UA003	UA	SFO	DEN		12	36	38	500.00	350.00	200.00	180.00
20	UA004	UA	DEN	SFO		12	36	38	500.00	350.00	200.00	180.00
21	UA005	UA	PDX	DEN		12	36	38	500.00	350.00	200.00	150.00
22	UA006	UA	DEN	PDX		12	36	38	500.00	350.00	200.00	180.00
23	UA007	UA	DEN	MSY		12	36	38	500.00	350.00	200.00	150.00
24	UA008	UA	MSY	DEN		12	36	38	500.00	350.00	200.00	150.00

Once back at the main flight screen we can also click through the origin or destination to see information regarding the airports. For this example, we will click through the MSY link to view information for this airport code.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The left sidebar lists databases like 'flyres', 'account', 'airline', 'employee', 'flight', 'passenger', 'reservation', 'scheduled_flight', 'information_schema', 'mysql', 'performance_schema', 'sys', and 'testing'. The 'flyres' database is selected. Inside 'flyres', the 'airport' table is selected. The table has columns: 'airport_id', 'airport_name', 'city', 'state', and 'country'. One row is displayed: 'MSY' (Louis Armstrong New Orleans International Airport) located in New Orleans, Louisiana, United States. Below the table, there are buttons for 'Edit', 'Copy', 'Delete', and 'Export'. At the bottom of the page, there is a 'Console' section.

By clicking through the leg_origin airport code we see the information about this airport. MSY is the airport code for Louis Armstrong New Orleans International Airport in New Orleans, Louisiana, United States. To return to the previous flight list we simply click the back button on the web browser to return to the previous page.

Flights by Airport Code

From the main flight list screen, we can search all flights that operate into or out of a specified airport or all flights operated by a specified airline. From the main flight list, above the flight table there is an input to filter rows. In this input field we can enter information to filter the results shown. Let's say we wanted to search all flights departing from or arriving at MSY. We would simply type MSY into this search box.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flights'. A search bar at the top contains the placeholder 'Search this table'. An arrow points from this search bar to the 'Filter rows' input field located just below the search bar in the main query results area. The results table displays flight records with various columns like flight_record, flight_number, airline, leg_origin, leg_destination, etc. The 'Filter rows' input field is highlighted.

Type search constraints into the input field to filter the table. For this example, we will type MSY into the box which returns the following result.

This screenshot shows the same flight table after filtering by 'MSY'. The results now include only four flights: two departing from MSY (AA001 and AA002) and two arriving at MSY (UA007 and UA008). The 'Filter rows' input field now contains 'MSY'.

	flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
<input type="checkbox"/>	1	AA001	AA	MSY	MCO	12	36	38	800.00	500.00	300.00	250.00
<input type="checkbox"/>	4	AA002	AA	MCO	MSY	12	36	38	800.00	600.00	400.00	250.00
<input type="checkbox"/>	23	UA007	UA	DEN	MSY	12	36	38	500.00	350.00	200.00	150.00
<input type="checkbox"/>	24	UA008	UA	MSY	DEN	12	36	38	500.00	350.00	200.00	150.00

We can see from the screen shot that there are two flights that depart from MSY and two flights which arrive at MSY. We will return to the main flight list by clicking the back button in the web browser.

Flights by Airline

Once back at the main flight list we can repeat the steps above using an airline code. So, in the filter rows input we can type in AA for American Airlines to view all flights operated by American Airlines. The results show that American Airlines operated four flights and all the information for these four flights.

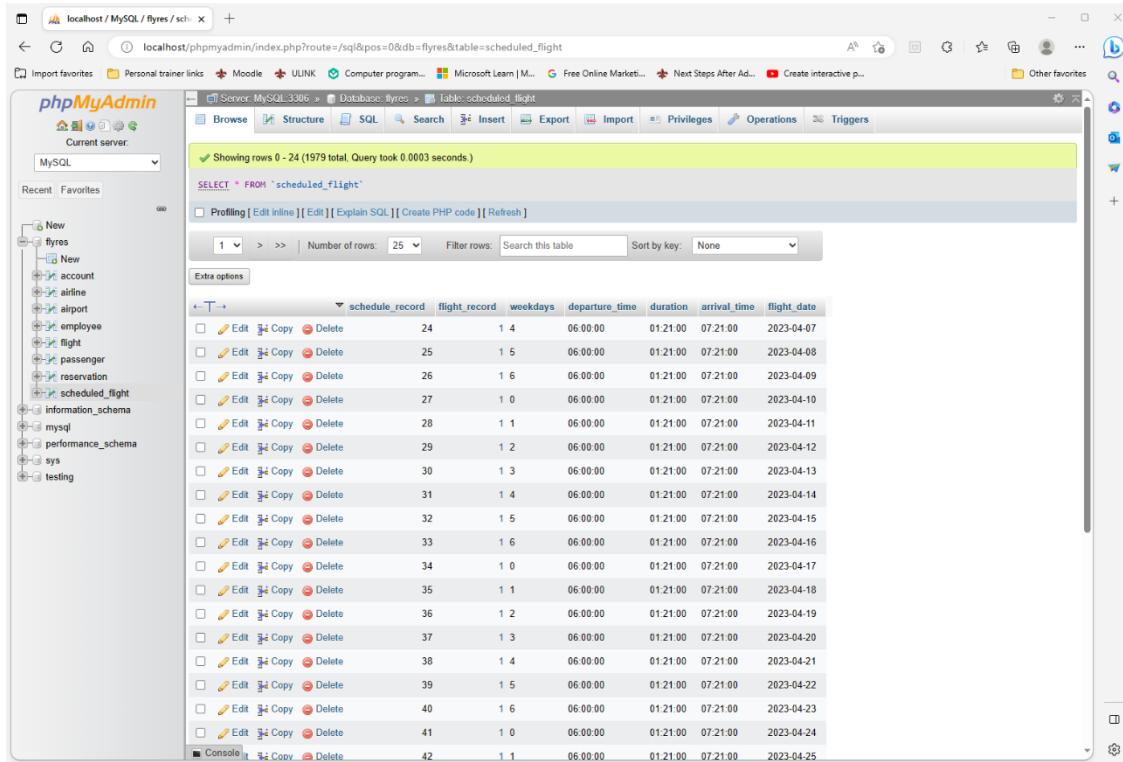
This screenshot shows the flight table again, but this time filtered by the airline code 'AA'. The results now include only four flights operated by American Airlines (AA001, AA002, AA003, and AA004). The 'Filter rows' input field now contains 'AA'.

	flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
<input type="checkbox"/>	1	AA001	AA	MSY	MCO	12	36	38	800.00	500.00	300.00	250.00
<input type="checkbox"/>	2	AA001	AA	MCO	MAD	33	48	185	1000.00	800.00	600.00	500.00
<input type="checkbox"/>	3	AA002	AA	MAD	MCO	33	48	185	1000.00	800.00	600.00	500.00
<input type="checkbox"/>	4	AA002	AA	MCO	MSY	12	36	38	800.00	600.00	300.00	250.00

Scheduled_flight Table

View All Scheduled Flights

The scheduled flight table contains flight schedule information such as flight date, departure times, flight duration, and scheduled arrival times. To view all of the scheduled flights we simply click on scheduled_flight from the main page of the flyres database.



The screenshot shows the phpMyAdmin interface for the 'flyres' database. The left sidebar shows the database structure with tables like account, airline, airport, employee, flight, passenger, reservation, and scheduled_flight. The 'scheduled_flight' table is selected. The main area displays the table data with the following columns: schedule_record, flight_record, weekdays, departure_time, duration, arrival_time, and flight_date. The data shows 42 rows of flight schedules for April 2023.

schedule_record	flight_record	weekdays	departure_time	duration	arrival_time	flight_date
24	1 4	06:00:00	01:21:00	07:21:00	2023-04-07	
25	1 5	06:00:00	01:21:00	07:21:00	2023-04-08	
26	1 6	06:00:00	01:21:00	07:21:00	2023-04-09	
27	1 0	06:00:00	01:21:00	07:21:00	2023-04-10	
28	1 1	06:00:00	01:21:00	07:21:00	2023-04-11	
29	1 2	06:00:00	01:21:00	07:21:00	2023-04-12	
30	1 3	06:00:00	01:21:00	07:21:00	2023-04-13	
31	1 4	06:00:00	01:21:00	07:21:00	2023-04-14	
32	1 5	06:00:00	01:21:00	07:21:00	2023-04-15	
33	1 6	06:00:00	01:21:00	07:21:00	2023-04-16	
34	1 0	06:00:00	01:21:00	07:21:00	2023-04-17	
35	1 1	06:00:00	01:21:00	07:21:00	2023-04-18	
36	1 2	06:00:00	01:21:00	07:21:00	2023-04-19	
37	1 3	06:00:00	01:21:00	07:21:00	2023-04-20	
38	1 4	06:00:00	01:21:00	07:21:00	2023-04-21	
39	1 5	06:00:00	01:21:00	07:21:00	2023-04-22	
40	1 6	06:00:00	01:21:00	07:21:00	2023-04-23	
41	1 0	06:00:00	01:21:00	07:21:00	2023-04-24	
42	1 1	06:00:00	01:21:00	07:21:00	2023-04-25	

View Scheduled Flights by Date

From the scheduled_flight list we can filter and sort results based on our needs. Let's say for example we want to see all flights departing on 2023-04-07. We simply type this information into the filter by search input and get the following results.

	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	24	1 4	06:00:00	01:21:00	07:21:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	59	32 4	13:15:00	02:45:00	16:00:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	60	31 4	02:00:00	10:15:00	12:15:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	61	30 4	04:00:00	10:15:00	14:15:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	62	29 4	00:15:00	02:45:00	03:00:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	63	28 4	13:00:00	10:30:00	23:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	64	27 4	20:00:00	10:30:00	06:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	65	26 4	05:00:00	10:00:00	15:00:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	66	25 4	01:30:00	10:00:00	11:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	67	24 4	10:00:00	02:55:00	12:55:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	68	23 4	06:00:00	02:55:00	08:55:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	69	22 4	15:00:00	02:30:00	17:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	70	21 4	19:00:00	02:30:00	21:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	71	20 4	12:15:00	02:30:00	14:45:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	72	19 4	09:00:00	02:30:00	11:30:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	73	18 4	13:00:00	02:55:00	15:55:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	74	17 4	05:40:00	02:55:00	08:35:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	75	16 4	18:30:00	02:20:00	20:50:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	76	15 4	17:00:00	02:20:00	19:20:00	2023-04-07
	<input type="checkbox"/>	Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	77	14 4	14:00:00	03:45:00	17:45:00	2023-04-07

Console Copy Delete

There are a number of ways to sort and filter flight information. The best way is to follow the information provided in this guide and adjust the input information for your specific scenario.

Customers, Passengers, and Accounts

Passenger Table

View All Passengers

The passenger table contains information about passengers. Not all passengers are customers. This is the case when one family member is a customer and makes reservations for flights for their family members. Any person traveling on a flight will have an entry in the passenger table. We can view the passenger table in its entirety by clicking on the passenger table from the main page of the flyres database or at any time selecting it from the left side panel under flyres.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The left sidebar lists various tables, and the main area displays the 'passenger' table. The table has columns for passenger_id, first_name, last_name, gender, date_of_birth, address, city, state, zip, phone, email, account, is_customer, meal_pref, seat_pref, and class_pref. There are 7 rows of data. Below the table, there are buttons for 'Check all', 'Edit', 'Copy', 'Delete', and 'Export'. The top navigation bar includes 'Import favorites', 'Personal trainer links', 'Moodle', 'ULINK', 'Computer program...', 'Microsoft Learn [M...]', 'Free Online Market...', 'Next Steps After Ad...', 'Create interactive p...', and 'Other favorites'.

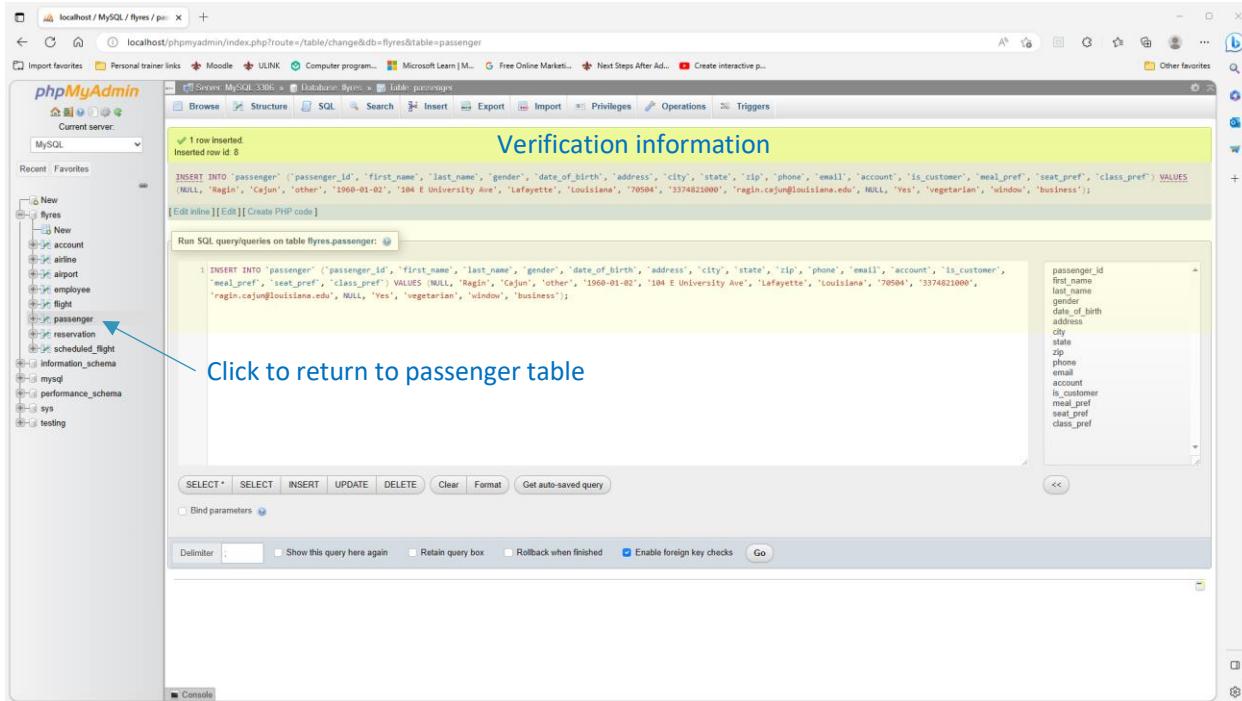
Adding a Passenger

From the main passenger table list we can add a new passenger by selecting the insert tab across the top toolbar. After selecting insert we are presented with the following screen for inputting the passenger information.

The screenshot shows the 'Insert' form for the 'passenger' table. The form contains fields for first_name (Ragin), last_name (Cajun), gender (other), date_of_birth (1960-01-02), address (104 E University Ave), city (Lafayette), state (Louisiana), zip (70504), phone (3374821000), and email (ragin.cajun@louisiana.edu). The 'passenger_id' field is empty, indicated by a red asterisk. The top navigation bar is identical to the previous screenshot.

Enter the information as described leaving the passenger_id field blank as this information is automatically generated. Leave the account field blank as we will address creating a new account and linking it to the

customer later in this section. Under the `is_customer` field select 'Yes' if the passenger is the one creating the reservation. Select 'No' if this entry is for a passenger who is not the individual creating the reservation. When finished, click 'Go' to save the entry in the passenger table. After selecting 'Go' we are presented with the following screen which confirms all of the information we just entered. To see the newly updated passenger table we can again click on the passenger table on the left side panel under flyres to view the entire passenger table.



The updated passenger table with our new customer/passenger is shown below.

	Edit	Copy	Delete	passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email	account	is_customer	meal_pref	seat_pref	class_pref
<input type="checkbox"/>	Edit	Copy	Delete	1	Robert	Smith	male	1980-09-03	456 Family Street	Lafayette	Louisiana	70506	3185554567	bob.smith@aol.com	1	Yes	turkey sandwich	aisle	economy
<input type="checkbox"/>	Edit	Copy	Delete	2	Susan	Smith	female	1982-04-20	456 Family Street	Lafayette	Louisiana	70506	3185554567	NULL	NULL	No	veggie plate	any	economy
<input type="checkbox"/>	Edit	Copy	Delete	3	Annie	Smith	female	2010-12-25	456 Family Street	Lafayette	Louisiana	70506	NULL	NULL	NULL	No	kids meal	window	economy
<input type="checkbox"/>	Edit	Copy	Delete	4	Donald	Duck	male	1976-03-03	100 Cartoon Lane	Hollywood	California	90210	9095551000	donnytheduck@quack.com	2	Yes	fruit plate	window	first
<input type="checkbox"/>	Edit	Copy	Delete	5	Tweety	Bird	other	2020-07-04	424 Twitter Rd	Denver	Colorado	80123	3037204242	ogwtwitter@aol.com	3	Yes	vegetarian	aisle	economy
<input type="checkbox"/>	Edit	Copy	Delete	6	Roger	Rabbit	male	1959-09-21	8 Bunnyhop Lane	Seattle	Washington	98101	2065551959	r.rabbit@verizon.net	4	Yes	NULL	NULL	business
<input type="checkbox"/>	Edit	Copy	Delete	7	Payton	Wiscowitch	male	1986-12-23	123 I Moved Street	San Francisco	California	94117	9099421495	C00122694@louisiana.edu	NULL	Yes	turkey sandwich	aisle	economy
<input type="checkbox"/>	Edit	Copy	Delete	8	Ragin	Cajun	other	1960-01-02	104 E University Ave	Lafayette	Louisiana	70504	3374821000	ragin.cajun@louisiana.edu	NULL	Yes	vegetarian	window	business

Edit Passenger Information

From the main passenger table, we can edit passenger information. Let's take our newest passenger Ragin Cajun for example. The passenger has moved to a new address, and we need to change the associated information. We can edit the passenger information by clicking the Edit link located next to the pencil icon to the left of the passenger entry. This brings us back to our insert page where we can adjust the information accordingly and select 'Go' to save.

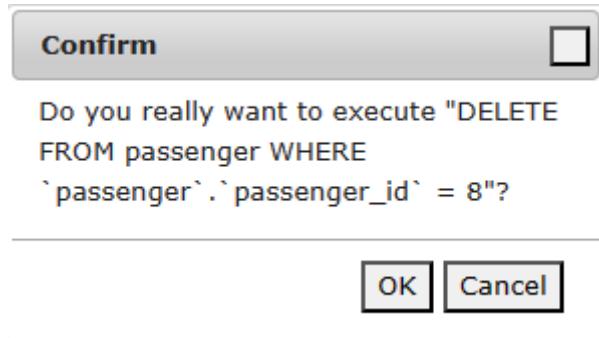
The entry has now been updated with the new address for the passenger.

passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email	account	is_customer	meal_pref	seat_pref	class_pref
8	Ragin	Cajun	other	1960-01-02	444 Cajundome Blvd 104 E University Ave	Lafayette	Louisiana	70506	3374821000	ragin.cajun@louisiana.edu	1	Yes	turkey sandwich	aisle	economy

Deleting a Passenger

Passengers can be deleted from the passenger table using the Delete link to the left of the passenger entry. However, once this information is deleted it cannot be recovered. So, it is important that we are sure this is

necessary before performing a delete procedure. We are presented with the following prompt when clicking Delete.



Again, it is important to be sure that we want to delete the selected information before confirming using the OK button shown above. Once OK is clicked the record will be permanently deleted.

Creating an Account

Each customer can have one or more accounts. This may be the case if the same customer makes reservations for both leisure travel and business travel. We can create a new account by first opening the account table by selecting account from the left side panel or from the main page of the flyres database.

account_id	owner	cc_number	open_date
1	1	4347123443219876	2023-01-02
2	4	3456738249089631	2023-02-08
3	5	3902491247894487	2023-04-03
4	6	4347091234987401	2023-02-07

From here, we can select the Insert tab along the top toolbar to create a new account. We need to know the passenger_id of the person we are creating the account for to link the account to the owner. In this example, we are creating an account for our newest customer Ragin Cajun passenger_id 8. You will enter the information as described, leaving the account_id field blank as this information is autogenerated. Under owner, select the correct passenger_id from the drop-down menu (in this case 8 with the corresponding street address of Ragin

Cajun). Type in the customer's credit card number and then select the current date from the calendar and select 'Go' to save.

We are then taken to a screen which verifies the information we have just entered. To return to the account table select account from the left side panel under flyres.

Our newest entry is now shown at the bottom of the account table.

	account_id	owner	cc_number	open_date
<input type="checkbox"/>	1	1	4347123443219876	2023-01-02
<input type="checkbox"/>	2	4	3456738249089831	2023-02-08
<input type="checkbox"/>	3	5	3902491247894487	2023-04-03
<input type="checkbox"/>	4	6	4347091234987401	2023-02-07
<input type="checkbox"/>	5	8	4445895234569876	2023-04-17

Linking an Account

Now that we have created a new account, we need to link the account to the passenger table. In our working example, our new account_id is 5 and belongs to passenger_id 8. We need to go back to the passenger table and edit the entry for passenger_id 8 to link the new account to the customer that owns the account. We open the passenger table by selecting passenger from the left side panel under flyres. We then select Edit for passenger_id 8.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. The left sidebar shows tables like 'flyres', 'New', 'account', 'airline', 'airport', 'employee', 'flight', 'passenger', 'reservation', 'scheduled_flight', 'information_schema', 'mysql', 'performance_schema', 'sys', and 'testing'. The 'passenger' table is selected. The main area displays the following data:

	passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email
<input type="checkbox"/>	1	Robert	Smith	male	1980-09-03	455 Family Street	Lafayette	Louisiana	70506	3185554567	bob.smith@aol.com
<input type="checkbox"/>	2	Susan	Smith	female	1982-04-20	456 Family Street	Lafayette	Louisiana	70506	3185554567	NULL
<input type="checkbox"/>	3	Annie	Smith	female	2010-12-25	456 Family Street	Lafayette	Louisiana	70506	NULL	NULL
<input type="checkbox"/>	4	Donald	Duck	male	1976-03-03	100 Cartoon Lane	Hollywood	California	90210	9095551000	donnytheduck@quack.com
<input type="checkbox"/>	5	Tweety	Bird	other	2020-07-04	424 Twitter Rd	Denver	Colorado	80123	3037204242	ogtwitter@aol.com
<input type="checkbox"/>	6	Roger	Rabbit	male	1959-09-21	8 Bunnyhop Lane	Seattle	Washington	98101	2065551969	rabbit@verizon.net
<input type="checkbox"/>	7	Payton	Wiscovitch	male	1986-12-23	123 I Moved Street	San Francisco	California	94117	9099421495	C00122694@louisiana.edu
<input checked="" type="checkbox"/>	8	Raghn	Cajun	other	1960-01-02	444 Cajundome Blvd	Lafayette	Louisiana	70506	3374821000	ragin.cajun@louisiana.edu

Under the insert page we select account_id 5 from the drop-down menu to associate account 5 with passenger 8.

account	int	<input type="button" value="▼"/>	<input type="checkbox"/>	4445895234569876 - 5 <input type="button" value="▼"/>
---------	-----	----------------------------------	--------------------------	---

Then we select 'Go' to save this information.

Verification information

Showing rows 0 - 7 (8 total). Query took 0.0003 seconds.

SELECT * FROM `passenger`

passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email	account	is_customer	meal_pref	seat_pref	class_pref
1	Robert	Smith	male	1980-09-03	456 Family Street	Lafayette	Louisiana	70506	3185554567	bob.smith@aol.com	1	Yes	turkey sandwich	aisle	economy
2	Susan	Smith	female	1982-04-20	456 Family Street	Lafayette	Louisiana	70506	3185554567	NULL	NULL	No	veggie plate	any	economy
3	Annie	Smith	female	2010-12-25	456 Family Street	Lafayette	Louisiana	70506	NULL	NULL	NULL	No	kids meal	window	economy
4	Donald	Duck	male	1976-03-03	100 Cartoon Lane	Hollywood	California	90210	9099551900	donnytheduck@quack.com	2	Yes	fruit plate	window	first
5	Tweety	Bird	other	2020-07-04	424 Twitter Rd	Denver	Colorado	80120	3037204242	ogthwne@aol.com	3	Yes	vegetarian	aisle	economy
6	Roger	Rabbit	male	1959-09-21	8 Bunnyhop Lane	Seattle	Washington	98101	2065551969	r.rabbit@verizon.net	4	Yes	NULL	NULL	business
7	Payton	Wisniewich	male	1986-12-23	123 Moved Street	San Francisco	California	94111	0999421495	C00122694@louisiana.edu	5	Yes	turkey sandwich	aisle	economy
8	Ragin	Cajun	other	1969-01-02	444 Cajundome Blvd	Lafayette	Louisiana	70506	3374821000	ragin.cajun@louisiana.edu	NULL	Yes	vegetarian	window	business

We have now created a new passenger/customer entry, edited the passenger information, created an account for the new customer, and linked the account to the customer.

Employees

Employee Table

The employee table has information about the customer representatives employed by FlyRes. The employee_id field is the employee's social security number and is a unique identifier for the employee table. All fields within the employee table are required fields and will require input for adding and saving new employees to the table.

View All Employees

To view all employees, we can select the employee table by clicking on employee in the left side panel or from the main page of the flyres database.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. The left sidebar lists databases and tables, with 'flyres' selected. The main area shows the 'employee' table with several rows of data. At the top, the 'Insert' tab is highlighted in blue. Below it, the table structure is shown with columns: employee_id, first_name, last_name, address, city, state, zip, phone, email, hire_date, and hourly_rate.

Adding a New Employee

After selecting the employee table from the left side panel or the main page of the flyres database, a new employee can be added by clicking the insert tab from the employee table main page.

The screenshot shows the 'Insert' form for the 'employee' table. The 'employee_id' field is filled with '46182534'. The 'first_name' field contains 'Alfred', 'last_name' contains 'Hitchcock', and 'address' contains '911 Bates Motel Road'. In the 'city' field, 'Death Valley' is typed. The 'state' field has 'California' selected. The 'zip' field contains '92328'. The 'phone' field contains '5556689111'. The 'email' field contains 'alfred.hitchcock@flyres.com'. The 'hire_date' field has '2023-04-18' selected. The 'hourly_rate' field contains '25.00'. At the bottom right, there is a 'Go' button.

The above input screen appears for adding a new employee. All fields are required. Enter all information as indicated and click 'Go' to save.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. A successful INSERT operation has been performed on the 'employee' table. The query inserted a new row with the following values:

```
INSERT INTO `employee`(`employee_id`, `first_name`, `last_name`, `address`, `city`, `state`, `zip`, `phone`, `email`, `hire_date`, `hourly_rate`) VALUES ('461825354', 'Alfred', 'Hitchcock', '911 Bates Hotel Road', 'Death Valley', 'California', '92328', '5556669111', 'alfred.hitchcock@flyres.com', '2023-04-18', '25.00');
```

A blue arrow points from the text "Return to employee table" to the 'employee' table entry in the database tree on the left.

Editing Employee Information

Employee information can be edited by selecting Edit from the employee table main page. For example, if one of our existing employees has received a raise and we need to change their hourly pay rate. To demonstrate this we will change the pay rate of our newest employee Alfred Hitchcock by selecting Edit next to the entry for this employee.

The screenshot shows the phpMyAdmin interface displaying the 'employee' table. The table contains five rows of data. The fifth row, which corresponds to Alfred Hitchcock, is selected and highlighted with a yellow background. A blue arrow points from the text "Edit" in the table header to the 'Edit' link next to Alfred Hitchcock's row.

	employee_id	first_name	last_name	address	city	state	zip	phone	email	hire_date	hourly_rate
<input type="checkbox"/>	11223333	John	Wick	337 Doggy Lane	Lafayette	Louisiana	70506	3375551234	john.wick@flyres.com	2020-04-01	25.00
<input type="checkbox"/>	337649834	Michael	Totoro	1 University Lane	Lafayette	Louisiana	70503	3375555649	michael.totoro@flyres.com	2020-01-01	30.00
<input type="checkbox"/>	459082345	Wonder	Woman	911 Hero St	Lafayette	Louisiana	70506	3375559911	wonder.woman@flyres.com	2021-01-04	22.00
<input type="checkbox"/>	461825354	Alfred	Hitchcock	911 Bates Hotel Road	Death Valley	California	92328	5556669111	alfred.hitchcock@flyres.com	2023-04-18	25.00
<input type="checkbox"/>	587654321	Bob	Builder	540 Construction Rd	Oakland	California	94037	5165559876	bob.builder@flyres.com	2020-07-03	20.00

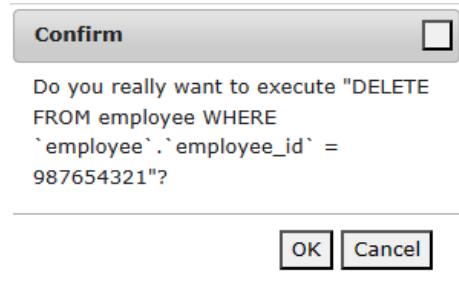
Once selecting edit next to the employee's entry, we are taken back to the insert screen where we can edit the current information. We will change the hourly rate and then select 'Go' to save the change.

The screenshot shows the phpMyAdmin interface for the 'employee' table. A specific row for employee_id 461825354 has been selected for editing. The 'hourly_rate' field is currently set to 25.00. A blue arrow labeled 'Changed rate' points to the input field where the value has been changed to 30.00. The 'Go' button is visible at the bottom right of the edit form. Below the edit form, a 'Verification information' box displays the SQL query used to update the record: `UPDATE `employee` SET `hourly_rate` = '30.00' WHERE `employee`.`employee_id` = 461825354;`. The main table view shows the updated hourly rate for this employee as 30.00. A blue text overlay 'New rate' is placed near the updated row in the table.

employee_id	first_name	last_name	address	city	state	zip	phone	email	hire_date	hourly_rate
111223333	John	Wick	337 Doggy Lane	Lafayette	Louisiana	70506	3375551234	john.wick@flyres.com	2020-04-01	25.00
337549834	Michael	Totaro	1 University Lane	Lafayette	Louisiana	70503	337555540	michael.totaro@flyres.com	2020-01-01	30.00
459082345	Wonder	Woman	911 Hero St	Lafayette	Louisiana	70506	3375559911	wonder.woman@flyres.com	2021-01-04	22.00
461825354	Alfred	Hitchcock	911 Bates Motel Road	Death Valley	California	92328	5556669111	alfred.hitchcock@flyres.com	2023-04-18	30.00
987654321	Bob	Builder	540 Construction Rd	Oakland	California	94607	5105559876	bob.builder@flyres.com	2020-07-03	20.00

Deleting an Employee

From the main page of the employee table an employee can be permanently deleted by selecting the Delete link next to the desired entry. This action is permanent, and we need to be certain that this is the desired outcome before completing this action. After selecting Delete we will be presented with the following prompt:



Before selecting OK it is important to be absolutely sure that the desired result is to permanently remove all information for this employee from the database.

Reservations

Reservation Table

The reservation table shows all reservations in the FlyRes system. The reservation table houses information from multiple different tables such as the passenger table, scheduled flight table, and the employee table. Each reservation is for a passenger, for a specific scheduled flight, and created by a customer representative. Reservations linked to passengers who are also customers will have fare information, booking fees, and total revenue gained from this specific reservation. A reservation record is a specific instance of a reservation while a reservation number can be shared among passengers traveling together under one reservation. For example, if a family is traveling together the reservation number is shared among all passengers in this group.

View All Reservations

In order to view all reservations in the system, we can simply select the reservation table under flyres from the left side panel or from the main page of the flyres database.

reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue
1	FR38RS1	2023-04-12	1	38	900.00	50.00	337549834	105.00
2	FR38RS1	2023-04-12	2	38	0.00	0.00	337549834	0.00
3	FR38RS1	2023-04-12	3	38	0.00	0.00	337549834	0.00
4	FR775RS2	2023-04-12	1	775	900.00	50.00	337549834	105.00
5	FR775RS2	2023-04-12	2	775	0.00	0.00	337549834	0.00
6	FR775RS2	2023-04-12	3	775	0.00	0.00	337549834	0.00
7	FR414DD1	2023-04-13	4	414	2000.00	30.00	459002345	330.00
8	FR557DD2	2023-04-13	4	557	2000.00	30.00	459002345	330.00
9	FR650DD3	2023-04-13	4	650	500.00	30.00	111223333	105.00
10	FR662DD3	2023-04-13	4	662	3000.00	30.00	111223333	480.00
11	FR841DD4	2023-04-13	4	841	3000.00	30.00	111223333	480.00
12	FR865DD4	2023-04-13	4	865	500.00	30.00	111223333	105.00
13	FR627TB1	2023-04-10	5	627	180.00	50.00	337549834	77.00
14	FR936TB2	2023-04-10	5	936	180.00	50.00	337549834	77.00
15	FR334TB3	2023-04-24	5	1328	150.00	50.00	337549834	72.50
16	FR1435TB4	2023-04-24	5	1435	150.00	50.00	337549834	72.50
17	FR334RR1	2023-04-13	6	334	300.00	30.00	987654321	75.00
18	FR333RR2	2023-04-13	6	333	300.00	30.00	987654321	75.00
19	FR512RR3	2023-04-13	6	512	300.00	30.00	987654321	75.00
20	FR511RR4	2023-04-13	6	511	300.00	30.00	987654321	75.00
23	FR678RR5	2023-04-13	6	678	1800.00	30.00	987654321	300.00
24	FR929RR6	2023-04-13	6	929	1800.00	30.00	987654321	300.00
27	FR1092RR7	2023-04-13	6	1092	2500.00	30.00	337549834	405.00
28	FR1343RR8	2023-04-13	6	1343	2500.00	30.00	337549834	405.00

View Reservation by [parameter]

We can view all the reservations for a particular parameter by searching for the parameter in the reservation table. For example, if we want to view all reservations for a specific passenger. We have a passenger named Donald Duck his passenger_id is 4. We can view all of Donald's reservations by selecting the reservation table in the left side panel and then using the search tab along the top toolbar of the reservation table.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The left sidebar lists various tables, including 'reservation'. The main area displays the 'reservation' table with 26 rows. The 'Search' button is highlighted with a blue arrow.

	reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue
1	FR38RS1	2023-04-12	1	38	900.00	50.00	337549834	185.00	
2	FR38RS1	2023-04-12	2	38	0.00	0.00	337549834	0.00	
3	FR38RS1	2023-04-12	3	38	0.00	0.00	337549834	0.00	
4	FR77RS2	2023-04-12	1	775	900.00	50.00	337549834	185.00	
5	FR77RS2	2023-04-12	2	775	0.00	0.00	337549834	0.00	
6	FR77RS2	2023-04-12	3	775	0.00	0.00	337549834	0.00	
7	FR414D01	2023-04-13	4	414	2000.00	30.00	459082345	330.00	
8	FR557D02	2023-04-13	4	557	2000.00	30.00	459082345	330.00	
9	FR662D03	2023-04-13	4	650	500.00	30.00	111223333	105.00	
10	FR662D03	2023-04-13	4	662	3000.00	30.00	111223333	480.00	
11	FR841D04	2023-04-13	4	841	3000.00	30.00	111223333	480.00	
12	FR865D04	2023-04-13	4	865	500.00	30.00	111223333	105.00	
13	FR827TB1	2023-04-10	5	827	180.00	50.00	337549834	77.00	
14	FR936T82	2023-04-10	5	936	180.00	50.00	337549834	77.00	
15	FR334TB3	2023-04-24	5	1328	150.00	50.00	337549834	72.50	
16	FR1435TB4	2023-04-24	5	1435	150.00	50.00	337549834	72.50	
17	FR334RR1	2023-04-13	6	334	300.00	30.00	987654321	75.00	
18	FR333R2	2023-04-13	6	333	300.00	30.00	987654321	75.00	

On the search input the passenger_id by selecting from the drop down menu for passenger. Then, select 'Go' to execute the search which will return all reservations for passenger_id 4, Donald Duck.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The left sidebar lists various tables, including 'reservation'. The main area displays the 'reservation' table with 5 rows. A search filter 'passenger' is set to '4', and the 'Go' button has been clicked. The results show all reservations for passenger_id 4.

	reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue
7	FR414D01	2023-04-13	4	414	2000.00	30.00	459082345	330.00	
8	FR557D02	2023-04-13	4	557	2000.00	30.00	459082345	330.00	
9	FR662D03	2023-04-13	4	650	500.00	30.00	111223333	105.00	
10	FR662D03	2023-04-13	4	662	3000.00	30.00	111223333	480.00	
11	FR841D04	2023-04-13	4	841	3000.00	30.00	111223333	480.00	

Creating a Reservation

Creating a reservation requires gathering information from multiple tables. Each reservation will require knowing the passenger_id of each passenger, the flight_record for the preferred flight and flight date, and the employee_id of the customer representative creating the reservation. We will also need to note the fare for

each flight reserved. Booking fees are based on the fare. Any fare < 500 has a \$50 booking fee while any fare > 500 has a booking fee of \$30. Fares and booking fees are only charged to the customer. So, passengers that are not also customers will not have any charges associated with their reservation.

Reservation Scenario

Passenger Payton Wiscovitch wants to make a roundtrip reservation from San Francisco to Denver and back leaving on 5/01/2023 and returning 5/07/2023.

Passenger Look-up

We need to gather the required information about our passenger. We will open the passenger table and search for our passenger. We can use the passenger's name to look them up, but there may be multiple passengers with the same name. So, we will try a unique identifier such as the phone number or email address associated with the passenger. In this case, Payton Wiscovitch is a unique name, but for demonstration purposes, let's look him up using the phone number 9099421495.

1. Open the passenger table by selecting passenger on the left side panel listed under flyres.
2. In the Filter rows: input where it says "Search this table" type in the passenger phone number.

The screenshot shows the phpMyAdmin interface with the 'passenger' table selected. A blue arrow points to the 'Filter rows' input field, which contains the phone number '9099421495'. The results table displays 8 rows of passenger data, including columns like passenger_id, first_name, last_name, gender, date_of_birth, address, city, state, zip, phone, email, account, is_customer, meal_pref, seat_pref, and class_pref. The 7th row, corresponding to Payton Wiscovitch, is highlighted.

passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email	account	is_customer	meal_pref	seat_pref	class_pref
1	Robert	Smith	male	1980-09-03	456 Family Street	Lafayette	Louisiana	70506	3185554567	bob.smith@aol.com	1	Yes	turkey sandwich	aisle	economy
2	Susan	Smith	female	1982-04-20	456 Family Street	Lafayette	Louisiana	70506	3185554567	NULL	NULL	No	veggie plate	any	economy
3	Annie	Smith	female	2010-12-25	456 Family Street	Lafayette	Louisiana	70506	NULL	NULL	NULL	No	kids meal	window	economy
4	Donald	Duck	male	1976-03-03	100 Cartoon Lane	Hollywood	California	90210	9099551000	donnytheduck@quack.com	2	Yes	fruit plate	window	first
5	Tweety	Bird	other	2020-07-04	424 Twitter Rd	Denver	Colorado	80123	3037204242	ogwitter@aol.com	3	Yes	vegetarian	aisle	economy
6	Roger	Rabbit	male	1959-09-21	8 Bunnyhop Lane	Seattle	Washington	98101	2065551959	r.rabbit@verizon.net	4	Yes	NULL	NULL	business
7	Payton	Wiscovitch	male	1986-12-23	123 I Moved Street	San Francisco	California	94117	9099421495	C00122694@louisiana.edu	6	Yes	turkey sandwich	aisle	economy
8	Ragin	Cajun	other	1960-01-02	444 Cajundome	Lafayette	Louisiana	70506	3374821000	ragin.cajun@louisiana.edu	5	Yes	vegetarian	window	business

3. Note the results of the passenger information.

The screenshot shows the passenger table results again, but this time the 7th row (Payton Wiscovitch) is highlighted in yellow. The rest of the table rows are greyed out.

passenger_id	first_name	last_name	gender	date_of_birth	address	city	state	zip	phone	email	account	is_customer	meal_pref	seat_pref	class_pref
7	Payton	Wiscovitch	male	1986-12-23	123 I Moved Street	San Francisco	California	94117	9099421495	C00122694@louisiana.edu	6	Yes	turkey sandwich	aisle	economy

4. Passenger_id = 7, class preference is economy

Gathering Flight Information

We are looking for flights from San Francisco to Denver departing on 5/1/2023 and flights from Denver to San Francisco departing on 5/7/2023.

Airport Code

1. The first step is noting the airport codes for San Francisco and Denver. If you know the airport codes skip to the next section. To do this, open the airport table by selecting airport from the left side panel.

Showing rows 0 - 24 (94 total, Query took 0.0004 seconds)

Select search

Search

Extra options

	airport_id	airport_name	city	state	country
<input type="checkbox"/>	ABQ	Albuquerque International Sunport	Albuquerque	New Mexico	United States
<input type="checkbox"/>	ANR	Ted Stevens Anchorage International Airport	Anchorage	Alaska	United States
<input type="checkbox"/>	ATH	Athens International Airport	Athens	NULL	Greece
<input type="checkbox"/>	ATL	Hartsfield-Jackson Atlanta International Airport	Atlanta	Georgia	United States
<input type="checkbox"/>	BCN	Barcelona International Airport	Barcelona	NULL	Spain
<input type="checkbox"/>	BDL	Bradley International Airport	Hartford	Connecticut	United States
<input type="checkbox"/>	BHM	Birmingham Shuttlesworth International Airport	Birmingham	Alabama	United States
<input type="checkbox"/>	BKK	Suvarnabhumi Airport	Bangkok	NULL	Thailand
<input type="checkbox"/>	BNA	Nashville International Airport	Nashville	Tennessee	United States
<input type="checkbox"/>	BOS	Gen. Edward Lawrence Logue International Airport	Boston	Massachusetts	United States
<input type="checkbox"/>	BTR	Baton Rouge Metropolitan Airport	Baton Rouge	Louisiana	United States
<input type="checkbox"/>	BTW	Burlington International Airport	Burlington	Vermont	United States
<input type="checkbox"/>	BWI	Baltimore/Washington International Airport	Baltimore	Maryland	United States
<input type="checkbox"/>	BZN	Bozeman Yellowstone International Airport	Bozeman	Montana	United States
<input type="checkbox"/>	CAI	Cairo International Airport	Cairo	NULL	Egypt
<input type="checkbox"/>	CDG	Paris Charles de Gaulle Airport	Paris	NULL	France
<input type="checkbox"/>	CHS	Charleston International Airport	Charleston	South Carolina	United States
<input type="checkbox"/>	CLE	Cleveland Hopkins International Airport	Cleveland	Ohio	United States
<input type="checkbox"/>	CLT	Charlotte Douglas International Airport	Charlotte	North Carolina	United States
<input type="checkbox"/>	CMH	John Glenn Columbus International Airport	Columbus	Ohio	United States
<input type="checkbox"/>	CUN	Cancun International Airport	Cancun	NULL	Mexico
<input type="checkbox"/>	DCA	Ronald Reagan Washington National Airport	Washington, D.C.	Virginia	United States
<input type="checkbox"/>	DEN	Denver International Airport	Denver	Colorado	United States
<input type="checkbox"/>	DFW	Dallas Fort Worth International Airport	Dallas	Texas	United States

Then select the search tab along the top toolbar. From the search input screen under city type in San

Do a "query by example" (wildcard: "%")

Column	Type	Collation	Operator	Value
airport_id	varchar(3)	utf8mb4_0900_ai_ci	LIKE	<input type="text"/>
airport_name	varchar(50)	utf8mb4_0900_ai_ci	LIKE	<input type="text"/>
city	varchar(50)	utf8mb4_0900_ai_ci	LIKE	<input type="text"/> San Francisco
state	varchar(50)	utf8mb4_0900_ai_ci	LIKE	<input type="text"/>
country	varchar(50)	utf8mb4_0900_ai_ci	LIKE	<input type="text"/>

Go

Francisco and select 'Go' to search.

2. In the results, note the airport code for San Francisco. To search for the Denver airport code repeat steps 1 and 2 entering Denver in the city search.

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT * FROM `airport` WHERE `city` LIKE 'San Francisco'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

airport_id	airport_name	city	state	country
SFO	San Francisco International Airport	San Francisco	California	United States

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT * FROM `airport` WHERE `city` LIKE 'Denver'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

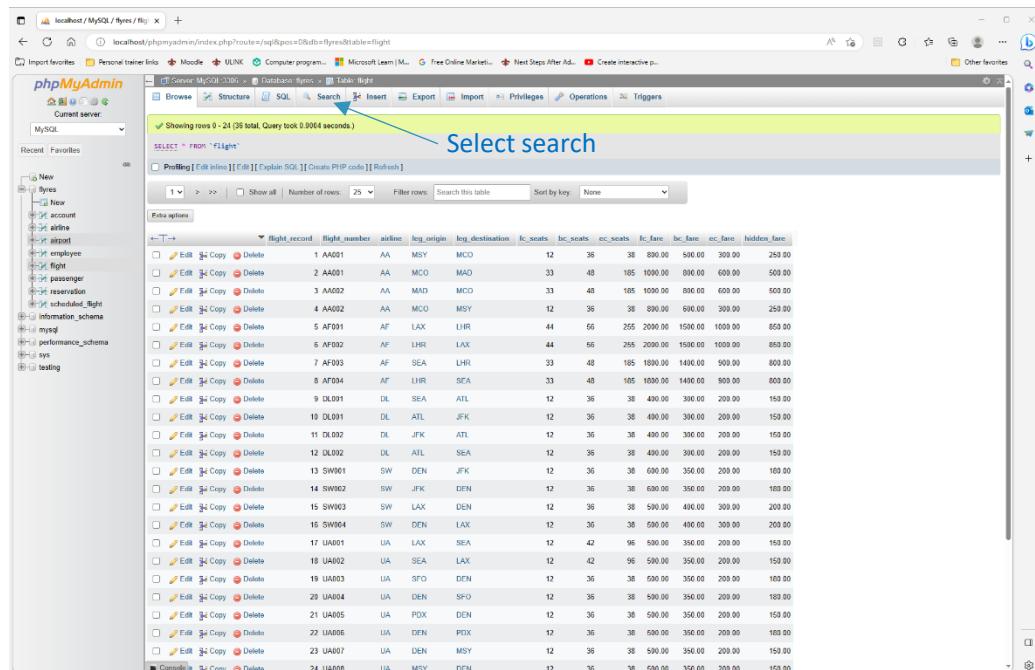
airport_id	airport_name	city	state	country
DEN	Denver International Airport	Denver	Colorado	United States

- Now we know we are looking for flights with origin/destination of SFO and DEN.

Flight Record

Our next step is figuring out which flights offer service between SFO and DEN. We are looking for a specific flight_record number for each leg of the trip.

- Open the flight table by selecting flight from the left side panel.
- Select the search tab along the top toolbar.



Showing rows 0 - 24 (39 total, Query took 0.0064 seconds.)

```
SELECT * FROM `flight`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	cc_seats	fc_fare	bc_fare	cc_fare	hidden_fare
1	AA001	AA	MSY	MCO	12	36	38	800.00	500.00	300.00	250.00
2	AA001	AA	MCO	MAD	33	48	105	1000.00	800.00	600.00	500.00
3	AA002	AA	MAD	MCO	33	48	105	1000.00	800.00	600.00	500.00
4	AA002	AA	MCO	MSY	12	36	38	800.00	600.00	300.00	250.00
5	AF001	AF	LAX	LHR	44	56	255	2000.00	1500.00	1000.00	850.00
6	AF002	AF	LHR	LAX	44	56	255	2000.00	1500.00	1000.00	850.00
7	AF003	AF	SEA	LHR	33	48	105	1800.00	1400.00	900.00	800.00
8	AF004	AF	LHR	SEA	33	48	105	1800.00	1400.00	900.00	800.00
9	DL001	DL	SEA	ATL	12	36	38	400.00	300.00	200.00	150.00
10	DL001	DL	ATL	JFK	12	36	38	400.00	300.00	200.00	150.00
11	DL002	DL	JFK	ATL	12	36	38	400.00	300.00	200.00	150.00
12	DL002	DL	ATL	SEA	12	36	38	400.00	300.00	200.00	150.00
13	SW001	SW	DEN	JFK	12	36	38	600.00	350.00	200.00	160.00
14	SW002	SW	JFK	DEN	12	36	38	600.00	350.00	200.00	160.00
15	SW003	SW	LAX	DEN	12	36	38	500.00	400.00	300.00	200.00
16	SW004	SW	DEN	LAX	12	36	38	500.00	400.00	300.00	200.00
17	UA001	UA	LAX	SEA	12	42	96	500.00	350.00	200.00	150.00
18	UA002	UA	SEA	LAX	12	42	96	500.00	350.00	200.00	150.00
19	UA003	UA	SFO	DEN	12	36	38	600.00	350.00	200.00	160.00
20	UA004	UA	DEN	SFO	12	36	38	500.00	350.00	200.00	160.00
21	UA005	UA	PDX	DEN	12	36	38	500.00	350.00	200.00	160.00
22	UA006	UA	DEN	PDX	12	36	38	500.00	350.00	200.00	160.00
23	UA007	UA	DEN	MSY	12	36	38	500.00	350.00	200.00	160.00
24	UA008	UA	MSY	DEN	12	36	38	500.00	350.00	200.00	160.00

- In the search input select SFO from the drop-down menu for leg_origin and DEN from the drop down menu for leg_destination. Then select 'Go' to perform the search.

Showing rows 0 - 0 (1 total, Query took 0.0009 seconds.)

```
SELECT * FROM `flight` WHERE `leg_origin` LIKE 'SFO' AND `leg_destination` LIKE 'DEN'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
<input type="checkbox"/>	19	UA003	UA	SFO	DEN	12	36	38	500.00	350.00	200.00	180.00

4. Based on the search results we will be looking for flight_record 19 when searching scheduled flights. This is for flight UA003 operating from SFO to DEN. We also note that the cost of an economy class ticket is \$200. So, our associated booking fee for this flight would be \$50.
5. Repeat steps 1-3 changing the leg_origin to DEN and leg_destination to SFO to search for the return flight for this roundtrip.

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT * FROM `flight` WHERE `leg_origin` LIKE 'DEN' AND `leg_destination` LIKE 'SFO'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	flight_record	flight_number	airline	leg_origin	leg_destination	fc_seats	bc_seats	ec_seats	fc_fare	bc_fare	ec_fare	hidden_fare
<input type="checkbox"/>	20	UA004	UA	DEN	SFO	12	36	38	500.00	350.00	200.00	180.00

6. We note from the results that we are looking up flight_record 20 when searching for scheduled flights for the passenger's return trip. This is for flight UA004 from DEN to SFO. The economy class fare is \$200 and will have a \$50 booking fee.

Searching for Scheduled Flights

Now that we know the flight_record number for each of the flights we can use this information to search for scheduled flights on the desired dates. So, we will be searching the scheduled flight table for flight_record 19 departing on 5/1/2023 and flight_record 20 departing on 5/7/2023.

1. Open the scheduled flight table by selecting scheduled_flight from the left side panel of the flyres database. Then select the search tab along the top toolbar to search the scheduled flights.

Showing rows 0 - 24 (1579 total. Query took 0.0004 seconds.)

Select search

	schedule_record	flight_record	weekdays	departure_time	duration	arrival_time	flight_date
<input type="checkbox"/>	24	1 4	06:00:00	01:21:00	07:21:00	2023-04-07	
<input type="checkbox"/>	25	1 5	06:00:00	01:21:00	07:21:00	2023-04-08	
<input type="checkbox"/>	26	1 6	06:00:00	01:21:00	07:21:00	2023-04-09	
<input type="checkbox"/>	27	1 0	06:00:00	01:21:00	07:21:00	2023-04-10	
<input type="checkbox"/>	28	1 1	06:00:00	01:21:00	07:21:00	2023-04-11	
<input type="checkbox"/>	29	1 2	06:00:00	01:21:00	07:21:00	2023-04-12	
<input type="checkbox"/>	30	1 3	06:00:00	01:21:00	07:21:00	2023-04-13	
<input type="checkbox"/>	31	1 4	06:00:00	01:21:00	07:21:00	2023-04-14	
<input type="checkbox"/>	32	1 5	06:00:00	01:21:00	07:21:00	2023-04-15	
<input type="checkbox"/>	33	1 6	06:00:00	01:21:00	07:21:00	2023-04-16	
<input type="checkbox"/>	34	1 0	06:00:00	01:21:00	07:21:00	2023-04-17	
<input type="checkbox"/>	35	1 1	06:00:00	01:21:00	07:21:00	2023-04-18	
<input type="checkbox"/>	36	1 2	06:00:00	01:21:00	07:21:00	2023-04-19	
<input type="checkbox"/>	37	1 3	06:00:00	01:21:00	07:21:00	2023-04-20	
<input type="checkbox"/>	38	1 4	06:00:00	01:21:00	07:21:00	2023-04-21	
<input type="checkbox"/>	39	1 5	06:00:00	01:21:00	07:21:00	2023-04-22	
<input type="checkbox"/>	40	1 6	06:00:00	01:21:00	07:21:00	2023-04-23	
<input type="checkbox"/>	41	1 0	06:00:00	01:21:00	07:21:00	2023-04-24	
<input type="checkbox"/>	42	1 1	06:00:00	01:21:00	07:21:00	2023-04-25	
<input type="checkbox"/>	43	1 2	06:00:00	01:21:00	07:21:00	2023-04-26	
<input type="checkbox"/>	44	1 3	06:00:00	01:21:00	07:21:00	2023-04-27	
<input type="checkbox"/>	45	1 4	06:00:00	01:21:00	07:21:00	2023-04-28	
<input type="checkbox"/>	46	1 5	06:00:00	01:21:00	07:21:00	2023-04-29	
<input type="checkbox"/>	47	1 6	06:00:00	01:21:00	07:21:00	2023-04-30	

2. From the search input screen select flight_record 19 from the dropdown menu and then select 5/1/2023 from the calendar for flight_date. Then select 'Go' to search.

Do a "query by example" (wildcard: "%")

Column	Type	Collation	Operator	Value
schedule_record	int		=	
flight_record	int		=	UA-19
weekdays	varchar(29)	utf8mb4_0900_ai_ci	LIKE	
departure_time	time		=	
duration	time		=	
arrival_time	time		=	
flight_date	date		=	2023-05-01

Extra options

Go

3. Results show that there is one flight from SFO to DEN (flight record 19) on 5/1/2023. The schedule_record is 936. We will use this information when making the reservation.

Showing rows 0 - 0 (1 total, Query took 0.0013 seconds.)

```
SELECT * FROM `scheduled_flight` WHERE `flight_record` = 19 AND `flight_date` = '2023-05-01'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	schedule_record	flight_record	weekdays	departure_time	duration	arrival_time	flight_date
<input type="checkbox"/>   	936	19	0	09:00:00	02:30:00	11:30:00	2023-05-01

4. Repeat steps 1-3 for flight_record 20 to search for a return trip on 5/7/2023.

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM `scheduled_flight` WHERE `flight_record` = 20 AND `flight_date` = '2023-05-07'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	schedule_record	flight_record	weekdays	departure_time	duration	arrival_time	flight_date
<input type="checkbox"/>   	1151	20	6	12:15:00	02:30:00	14:45:00	2023-05-07

5. Results show there is one flight from DEN to SFO (flight record 20) on 5/7/2023. The schedule record is 1151. We will use this information to make the return flight reservation.

Reviewing all information

From the above searches we have compiled all of the information needed to make the necessary reservations. The **passenger_id** for Payton Wiscovitch is **7**. He is traveling from **SFO to DEN** on **schedule_record 936** on 5/1/2023 on an **economy fare of \$200** with a **\$50 booking fee** for this flight. He will return on **schedule_record 1151** from **DEN to SFO** on 5/7/2023 on an **economy fare of \$200** with a **booking fee of \$50**.

Creating the Reservations

We now have all our input information for creating the reservations for Payton Wiscovitch. We will now create a reservation by inputting this information into the reservation table.

1. Open the reservation table by selecting reservation from the left side panel under flyres.
2. Select the insert tab across the top toolbar of the reservation table.

Select Insert

- On the insert page, leave reservation record blank as this information is automatically generated.

- Reservation number: FR[insert the schedule record number][passenger first and last initial][#of reservation being created in this session]. For this scenario, our first reservation will be for the flight from SFO to DEN on 5/1/23. Reservation number: FR936PW1.
- Select current date as booking date.
- Select passenger number 7 for Payton Wiscovitch from the drop down menu for passenger.

- For scheduled flight type in the flight record number for the first flight: 936.
- For the fare type in the economy class fare for this flight: 200.
- For the booking fee type in the booking fee for this flight: 50.
- Under customer_rep select your employee id.
- Revenue = leave blank this will be calculated later.

Select 'Go' to save.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The 'reservation' table is selected. A success message at the top states '1 row inserted. Inserted row id: 29'. Below it, the SQL query is displayed:

```
INSERT INTO `reservation` (`reservation_record`, `reservation_number`, `booking_date`, `passenger`, `scheduled_flight`, `fare`, `booking_fee`, `customer_rep`, `revenue`) VALUES (NULL, 'FR936PW1', '2023-04-18', '7', '936', '200', '50', '111223333', '(fare*.15) + booking_fee');
```

The right side of the interface shows the table structure with columns: reservation_record, reservation_number, booking_date, passenger, scheduled_flight, fare, booking_fee, customer_rep, and revenue.

4. This is verification of a saved reservation. Now, repeat steps 1-3 to create the reservation for the return trip for flight record 1151 reservation number FR1151PW2.

localhost / MySQL / flyres / reservation

localhost/phpmyadmin/index.php?route=/table/change&db=flyres&table=reservation

Import favorites Personal trainer links Moodle ULINK Computer program... Microsoft Learn | M... Free Online Marketi... Next Steps After Ad... Create interactive p... Other favorites

phpMyAdmin

Current server: MySQL

Recent Favorites

New flyres New account airline airport employee flight passenger reservation scheduled_flight information_schema mysql performance_schema sys testing

Server: MySQL 3306 Database: flyres Table: reservation

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Column	Type	Function	Null	Value
reservation_record	int			
reservation_number	varchar(10)			FR1151PW2
booking_date	date			2023-04-18
passenger	int			123 I Moved Street - 7
scheduled_flight	int			1151
fare	decimal(10,2)			200
booking_fee	decimal(10,2)			50
customer_rep	int			337 Doggy Lane - 111223333
revenue	decimal(10,2)			

Go

Ignore

Column	Type	Function	Null	Value
reservation_record	int			
reservation_number	varchar(10)			

Console

localhost / MySQL / flyres / reservation

localhost/phpmyadmin/index.php?route=/table/change&db=flyres&table=reservation

Import favorites Personal trainer links Moodle ULINK Computer program... Microsoft Learn | M... Free Online Marketi... Next Steps After Ad... Create interactive p... Other favorites

phpMyAdmin

Current server: MySQL

Recent Favorites

New flyres New account airline airport employee flight passenger reservation scheduled_flight information_schema mysql performance_schema sys testing

Server: MySQL 3306 Database: flyres Table: reservation

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

✓ 1 row inserted.
Inserted row id: 30

INSERT INTO `reservation` (`reservation_record`, `reservation_number`, `booking_date`, `passenger`, `scheduled_flight`, `fare`, `booking_fee`, `customer_rep`, `revenue`) VALUES (NULL, 'FR1151PW2', '2023-04-18', '7', '1151', '200', '50', '111223333', NULL);

[Edit inline] [Edit] [Create PHP code]

Run SQL query/queries on table flyres.reservation:

```
1 INSERT INTO `reservation` (`reservation_record`, `reservation_number`, `booking_date`, `passenger`, `scheduled_flight`, `fare`, `booking_fee`, `customer_rep`, `revenue`) VALUES (NULL, 'FR1151PW2', '2023-04-18', '7', '1151', '200', '50', '111223333', NULL);
```

reservation_record
reservation_number
booking_date
passenger
scheduled_flight
fare
booking_fee
customer_rep
revenue

SELECT * SELECT INSERT UPDATE DELETE Clear Format Get auto-saved query

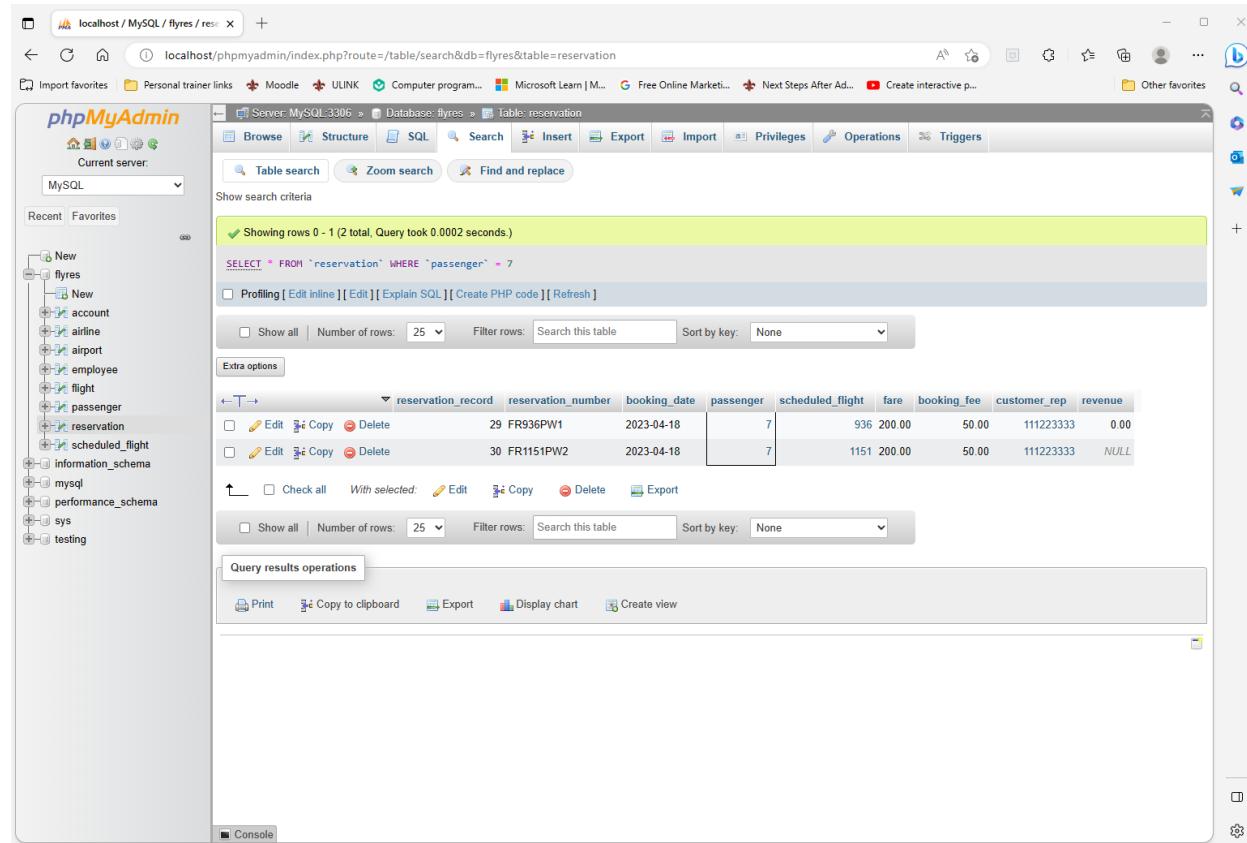
Bind parameters

Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks Go

Console

Viewing New Reservations

We can now view our new reservations by returning to the main reservation table and performing a search for passenger 7, Payton Wiscovitch. This will return all reservations for this passenger.



The screenshot shows the phpMyAdmin interface for the 'flyres' database. The left sidebar lists various tables: New, flyres, account, airline, airport, employee, flight, passenger, reservation, scheduled_flight, information_schema, mysql, performance_schema, sys, and testing. The 'reservation' table is selected in the main area. The SQL tab contains the query: `SELECT * FROM `reservation` WHERE `passenger` = 7`. The results grid shows two rows of data:

	reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue
29 FR936PW1	2023-04-18	7	936	200.00	50.00	111223333	0.00		
30 FR1151PW2	2023-04-18	7	1151	200.00	50.00	111223333	NULL		

Calculating Revenue

In order to calculate revenue for new reservations, we need to update the revenue column in the reservation table. This is done by performing an UPDATE on the reservation table.

1. Open the reservation table by clicking reservation on the left side panel and then click the SQL tab across the top toolbar.
2. Copy and paste or type the following command:

```
UPDATE reservation SET revenue = (fare*.15) + booking_fee;
```

3. Select 'Go' to execute the command.

Showing rows 0 - 24 (25 total, Query took 0.0003 seconds.)

SELECT * FROM `reservation`

	reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue	
<input type="checkbox"/>	Edit	Copy	Delete	1	FR38RS1	2023-04-12	38	900.00	50.00	337549834 185.00
<input type="checkbox"/>	Edit	Copy	Delete	2	FR38RS1	2023-04-12	2	38.00	0.00	337549834 0.00
<input type="checkbox"/>	Edit	Copy	Delete	3	FR38RS1	2023-04-12	3	38.00	0.00	337549834 0.00
<input type="checkbox"/>	Edit	Copy	Delete	4	FR775RS2	2023-04-12	1	775.00	50.00	337549834 185.00
<input type="checkbox"/>	Edit	Copy	Delete	5	FR775RS2	2023-04-12	2	775.00	0.00	337549834 0.00
<input type="checkbox"/>	Edit	Copy	Delete	6	FR775RS2	2023-04-12	3	775.00	0.00	337549834 0.00
<input type="checkbox"/>	Edit	Copy	Delete	7	FR414DD1	2023-04-13	4	414.00	30.00	459082345 330.00
<input type="checkbox"/>	Edit	Copy	Delete	8	FR657DD2	2023-04-13	4	557.00	30.00	459082345 330.00
<input type="checkbox"/>	Edit	Copy	Delete	9	FR650DD3	2023-04-13	4	650.00	30.00	111223333 105.00
<input type="checkbox"/>	Edit	Copy	Delete	10	FR662DD3	2023-04-13	4	662.00	30.00	111223333 105.00
<input type="checkbox"/>	Edit	Copy	Delete	11	FR841DD4	2023-04-13	4	841.00	30.00	111223333 105.00
<input type="checkbox"/>	Edit	Copy	Delete	12	FR865DD4	2023-04-13	4	865.00	30.00	111223333 105.00
<input type="checkbox"/>	Edit	Copy	Delete	13	FR827TB1	2023-04-10	5	827.00	50.00	337549834 77.00
<input type="checkbox"/>	Edit	Copy	Delete	14	FR936TB2	2023-04-10	5	936.00	50.00	337549834 77.00
<input type="checkbox"/>	Edit	Copy	Delete	15	FR334TB3	2023-04-24	5	1328.00	50.00	337549834 72.50
<input type="checkbox"/>	Edit	Copy	Delete	16	FR1436TB4	2023-04-24	5	1435.00	50.00	337549834 72.50
<input type="checkbox"/>	Edit	Copy	Delete	17	FR334RR1	2023-04-13	6	334.00	30.00	987654321 75.00
<input type="checkbox"/>	Edit	Copy	Delete	18	FR333RR2	2023-04-13	6	333.00	30.00	987654321 75.00

- If prompted, confirm that you would like to execute the UPDATE command. You can now go back to the reservation table and view the new reservations with the updated revenue generated by these transactions.

Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

SELECT * FROM `reservation` WHERE `passenger` = 7

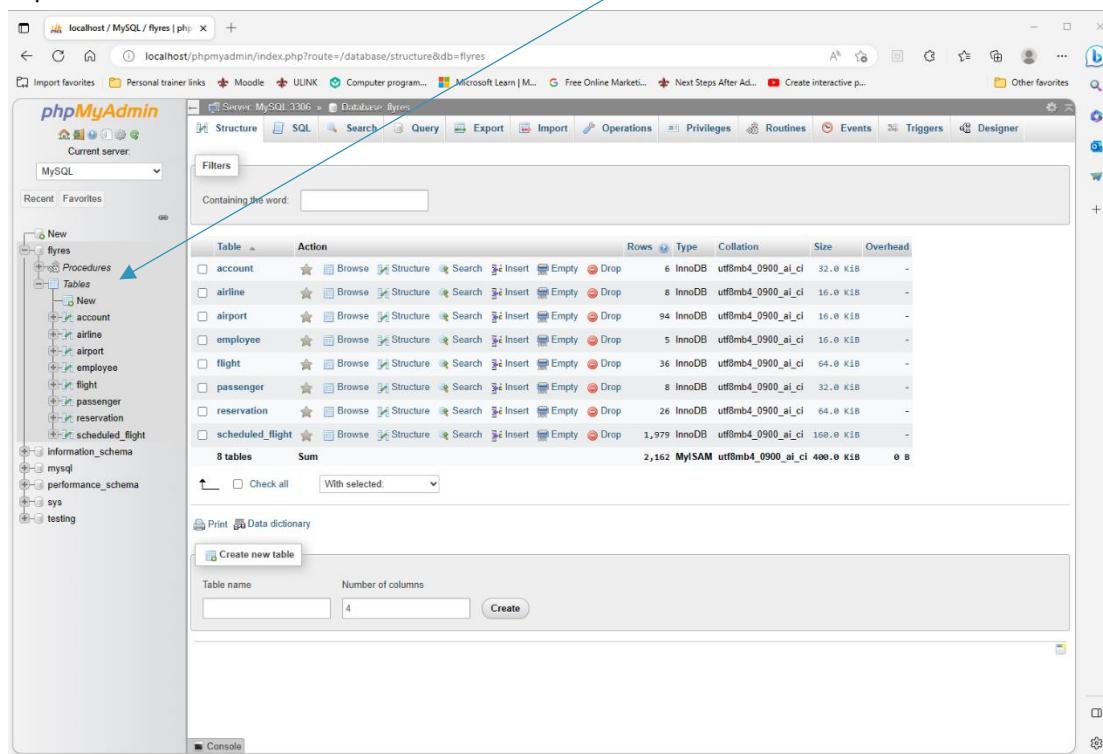
	reservation_record	reservation_number	booking_date	passenger	scheduled_flight	fare	booking_fee	customer_rep	revenue	
<input type="checkbox"/>	Edit	Copy	Delete	29	FR936PW1	2023-04-18	7	936.00	50.00	111223333 80.00
<input type="checkbox"/>	Edit	Copy	Delete	30	FR1151PW2	2023-04-18	7	1151.00	50.00	111223333 80.00

System Reports

There are various reports that can be generated using the phpMyAdmin graphical user interface for FlyRes. This section will show how to create and view each of these reports. Some of these reports are generated using stored procedures. The list of stored procedures can be accessed from the left side panel of the FlyRes database. Simply click on Procedures listed under flyres to view this list. Then, select execute under the desired report. Some reports will require executing commands in the SQL command line. All instructions for reports are listed below.

Passenger Mailing List

1. To generate a passenger mailing list, click on the Procedures link directly under flyres in the left side panel.



The screenshot shows the phpMyAdmin interface for the FlyRes database. The left sidebar shows the database structure with the 'flyres' schema expanded, revealing tables like account, airline, airport, employee, flight, passenger, reservation, and scheduled_flight. Under the 'Procedures' link, there are several stored procedures listed: New, account, airline, airport, employee, flight, passenger, reservation, and scheduled_flight. The main pane displays a table of these procedures with columns for Table, Action, Rows, Type, Collation, Size, and Overhead. At the bottom of the main pane, there is a 'Create new table' form. The top navigation bar includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, and Designer.

2. The stored procedures list will appear allowing you to select which report you would like to generate.
3. Locate the procedure named '**passenger mailing list**' and select **Execute**.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. In the left sidebar, under 'Procedures', there is a list of stored procedures. One of them, 'passenger mailing list', has its 'Execute' link highlighted with a blue arrow.

Name	Type	Returns
Customer Email List	PROCEDURE	Edit Execute Export Drop
Employee email list	PROCEDURE	Edit Execute Export Drop
Employee mailing list	PROCEDURE	Edit Execute Export Drop
passenger mailing list	PROCEDURE	Edit Execute Export Drop
revenue by customer	PROCEDURE	Edit Execute Export Drop
revenue by employee	PROCEDURE	Edit Execute Export Drop

4. The report is generated.

The screenshot shows the phpMyAdmin interface after executing the 'passenger mailing list' procedure. The message bar indicates success: 'Your SQL query has been executed successfully. 8 rows affected by the last statement inside the procedure.' Below this, the execution results are displayed in a table:

first_name	last_name	address	city	state	zip
Robert	Smith	456 Family Street	Lafayette	Louisiana	70506
Susan	Smith	456 Family Street	Lafayette	Louisiana	70506
Annie	Smith	456 Family Street	Lafayette	Louisiana	70506
Donald	Duck	100 Cartoon Lane	Hollywood	California	90210
Tweety	Bird	424 Twitter Rd	Denver	Colorado	80123
Roger	Rabbit	8 Bunnyhop Lane	Seattle	Washington	98101
Payton	Wicowitch	123 Moved Street	San Francisco	California	94117
Ragin	Cajun	444 Cajundome Blvd	Lafayette	Louisiana	70506

A blue bracket on the right side of the results table is labeled 'Passenger mailing list'.

Customer Email List

1. To generate a customer email list, click on Procedures under flyres from the left side panel.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. On the left sidebar, under the 'flyres' schema, the 'Procedures' node is selected, indicated by a blue arrow. The main pane displays a table of 8 tables with their respective details. A search bar at the top is set to 'Containing the word:'. At the bottom, there are buttons for 'Check all' and 'With selected'. A 'Create new table' dialog is open in the bottom right corner.

Table	Action	Rows	Type	Collation	Size	Overhead
account	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci	32.0 KIB	-
airline	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	16.0 KIB	-
airport	Browse Structure Search Insert Empty Drop	94	InnoDB	utf8mb4_0900_ai_ci	16.0 KIB	-
employee	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_0900_ai_ci	16.0 KIB	-
flight	Browse Structure Search Insert Empty Drop	36	InnoDB	utf8mb4_0900_ai_ci	64.0 KIB	-
passenger	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	32.0 KIB	-
reservation	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_0900_ai_ci	64.0 KIB	-
scheduled_flight	Browse Structure Search Insert Empty Drop	1,979	InnoDB	utf8mb4_0900_ai_ci	168.0 KIB	-
Sum		2,162	MyISAM	utf8mb4_0900_ai_ci	400.0 KIB	0 B

2. The stored procedures list will appear allowing you to select which report you would like to generate.
3. Locate the procedure named '**Customer Email List**' and select **Execute**.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. On the left sidebar, under the 'flyres' schema, the 'Routines' node is selected, indicated by a blue arrow. The main pane displays a table of stored procedures. A blue arrow points to the 'Customer Email List' row. The 'Type' column shows 'PROCEDURE' for all rows. The 'Returns' column is empty. There are 'Edit', 'Execute', 'Export', and 'Drop' buttons for each row. A search bar at the top right is set to 'Search'.

Name	Type	Returns
Customer Email List	PROCEDURE	
Employee email list	PROCEDURE	
Employee mailing list	PROCEDURE	
passenger mailing list	PROCEDURE	
revenue by customer	PROCEDURE	
revenue by employee	PROCEDURE	

4. The customer email list is generated.

Customer email list

first_name	last_name	email
Robert	Smith	bob.smith@aol.com
Donald	Duck	donaldtheduck@quack.com
Tweety	Bird	tweetybird@aol.com
Roger	Rabbit	rabbit@verizon.net
Peyton	Wisecratch	C00122694@louisiana.edu
Ragin	Cajun	ragin.cajun@louisiana.edu

Employee Mailing List

- To generate an employee mailing list, click on Procedures under flyres from the left side panel.

Table	Action	Rows	Type	Collation	Size	Overhead
account	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci	32.0 Kib	-
airline	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
airport	Browse Structure Search Insert Empty Drop	94	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
employee	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
flight	Browse Structure Search Insert Empty Drop	36	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	-
passenger	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	32.0 Kib	-
reservation	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	-
scheduled_flight	Browse Structure Search Insert Empty Drop	1,979	InnoDB	utf8mb4_0900_ai_ci	160.0 Kib	-
8 tables	Sum	2,162	MySAM	utf8mb4_0900_ai_ci	400.0 Kib	0 B

- The stored procedures list will appear allowing you to select which report you would like to generate.
- Locate the procedure named '**Employee mailing list**' and select **Execute**.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. The left sidebar shows tables like 'airline', 'account', 'flight', etc. The main area displays a list of routines:

Name	Type	Returns
Customer Email List	PROCEDURE	
Employee email list	PROCEDURE	
Employee mailing list	PROCEDURE	
passenger mailing list	PROCEDURE	
revenue by customer	PROCEDURE	
revenue by employee	PROCEDURE	

A blue arrow points to the 'Execute' button for the 'Employee mailing list' row.

4. The employee mailing list is generated.

The screenshot shows the phpMyAdmin interface after executing the 'Employee mailing list' procedure. The results are displayed in a table:

first_name	last_name	address	city	state	zip
John	Wick	337 Doggy Lane	Lafayette	Louisiana	70596
Michael	Totoro	1 University Lane	Lafayette	Louisiana	70593
Wonder	Woman	911 Hero St	Lafayette	Louisiana	70596
Alfred	Hitchcock	911 Bates Motel Road	Death Valley	California	92328
Bob	Builder	540 Construction Rd	Oakland	California	94607

A blue bracket groups the table output, which is labeled 'Employee mailing list'.

Employee Email List

1. To generate an employee email list, click on Procedures under flyres from the left side panel.

The screenshot shows the phpMyAdmin interface for the MySQL 'flyres' database. On the left sidebar, under the 'flyres' schema, the 'Procedures' node is selected, indicated by a blue arrow. The main pane displays a table of stored procedures:

Table	Action	Rows	Type	Collation	Size	Overhead
account	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci	32.0 Kib	-
airline	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
airport	Browse Structure Search Insert Empty Drop	94	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
employee	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
flight	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	-
passenger	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_0900_ai_ci	32.0 Kib	-
reservation	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_0900_ai_ci	64.0 Kib	-
scheduled_flight	Browse Structure Search Insert Empty Drop	1,979	InnoDB	utf8mb4_0900_ai_ci	160.0 Kib	-
8 tables	Sum	2,162	MyISAM	utf8mb4_0900_ai_ci	400.0 Kib	0 B

2. The stored procedures list will appear allowing you to select which report you would like to generate.
3. Locate the procedure named '**Employee email list**' and select **Execute**.

The screenshot shows the phpMyAdmin interface for the MySQL 'flyres' database. On the left sidebar, under the 'flyres' schema, the 'Routines' node is selected. The main pane displays a table of stored routines:

Name	Type	Returns	Action
Customer Email List	PROCEDURE		Edit Execute Export Drop
Employee email list	PROCEDURE		Edit Execute Export Drop
Employee mailing list	PROCEDURE		Edit Execute Export Drop
passenger mailing list	PROCEDURE		Edit Execute Export Drop
revenue by customer	PROCEDURE		Edit Execute Export Drop
revenue by employee	PROCEDURE		Edit Execute Export Drop

A blue arrow points to the 'Execute' button for the 'Employee email list' row.

4. The employee email list is generated.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. In the left sidebar, under the 'Procedures' section of the 'flyres' schema, there is a list of stored procedures. One of them, 'Employee_email_list', is highlighted. Below the list, a modal window titled 'Execution results of routine 'Employee_email_list'' is open, showing the output of the procedure. The output table has columns 'first_name', 'last_name', and 'email'. The data is as follows:

first_name	last_name	email
John	Wick	john.wick@flyres.com
Michael	Tarantino	michael.tarantino@flyres.com
Wonder	Woman	wonder.woman@flyres.com
Alfred	Hitchcock	alfred.hitchcock@flyres.com
Bob	Bulldozer	bob.bulldozer@flyres.com

Revenue by Customer

1. To generate a revenue report by customer, click on Procedures under flyres on the left side panel.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. In the left sidebar, under the 'Tables' section of the 'flyres' schema, there is a list of tables. One of them, 'account', is highlighted. Below the list, a modal window titled 'Filters' is open, showing a search bar with the placeholder 'Containing the word:' and a table of results. The table has columns 'Table' and 'Action'. The data is as follows:

Table	Action
account	Browse Structure Search Insert Empty Drop
airline	Browse Structure Search Insert Empty Drop
airport	Browse Structure Search Insert Empty Drop
employee	Browse Structure Search Insert Empty Drop
flight	Browse Structure Search Insert Empty Drop
passenger	Browse Structure Search Insert Empty Drop
reservation	Browse Structure Search Insert Empty Drop
scheduled_flight	Browse Structure Search Insert Empty Drop

2. The stored procedures list will appear allowing you to select which report you would like to generate.

3. Locate the procedure named '**revenue by customer**' and select **Execute**.

The screenshot shows the phpMyAdmin interface with the 'Routines' tab selected. In the main pane, there is a table titled 'Routines' with columns 'Name', 'Type', and 'Returns'. The table lists several procedures, including 'Customer Email List', 'Employee email list', 'Employee mailing list', 'passenger mailing list', 'revenue by customer', and 'revenue by employee'. The 'revenue by customer' row has its 'Execute' button highlighted with a blue oval and a blue arrow pointing to it from the right.

4. You will be presented with a Routine parameters input and will need to provide the passenger_id for which you are generating the report. For our example, we will calculate the total revenue generated by customer/passenger 1, Robert Smith. Type the pid (passenger id) number into the value box. Then select 'Go' to execute the report.

A modal dialog box titled 'Execute routine `revenue by customer`' is shown. It contains a 'Routine parameters' section with a table. The table has columns 'Name', 'Type', 'Function', and 'Value'. A single row is present with 'pid' as the name, 'INT' as the type, and '1' as the value. A blue arrow points from the 'Value' field to the '1' value. At the bottom of the dialog are 'Go' and 'Close' buttons.

5. Total revenue generated for passenger 1 is calculated and returned.

A modal dialog box titled 'Execution results of routine `revenue by customer`' is shown. It displays the output of the procedure. The output is a single row with the column name 'passenger_revenue' and the value '370.00'.

Revenue by Employee

1. To generate a revenue report by employee, click on Procedures under flyres on the left side panel.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. On the left sidebar, under the 'Procedures' section, there is a list of stored procedures: Customer Email List, Employee email list, Employee mailing list, passenger mailing list, revenue by customer, and revenue by employee. A blue arrow points from the text in step 1 to the 'Procedures' link in the sidebar.

2. The stored procedures list will appear allowing you to select which report you would like to generate.
3. Locate the procedure named '**revenue by employee**' and select **Execute**.

The screenshot shows the phpMyAdmin interface for the 'flyres' database. On the left sidebar, under the 'Routines' section, there is a list of stored procedures: Customer Email List, Employee email list, Employee mailing list, passenger mailing list, revenue by customer, and revenue by employee. The 'revenue by employee' procedure is highlighted with a blue circle around the 'Execute' button. A blue arrow points from the text in step 3 to the 'Execute' button for the 'revenue by employee' procedure.

4. You will be presented with a Routine parameters input and will need to provide the employee_id for which you are generating the report. For our example, we will calculate the total revenue generated

by employee John Wick with employee_id 111223333. Type the eid (employee id) number into the value box. Then select 'Go' to execute the report.

Execute routine 'revenue by employee'

Routine parameters

Name	Type	Function	Value
eid	INT		111223333

Go Close

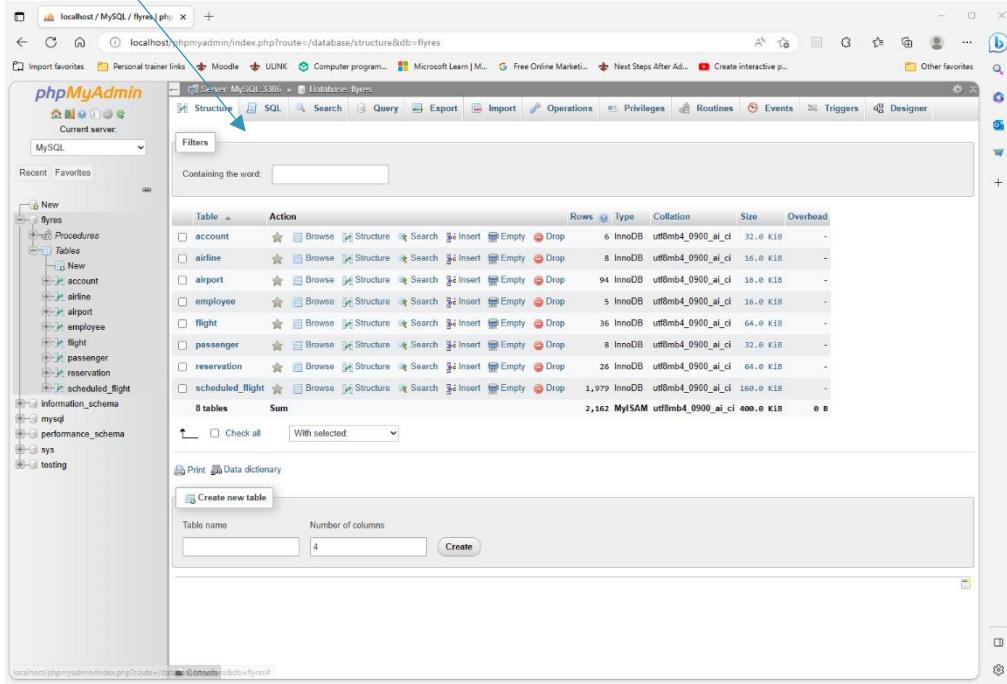
5. The total revenue generated by employee John Wick 111223333 is calculated.

Execution results of routine 'revenue by employee'

employee_revenue
1330.00

Revenue by Month

1. Select the SQL tab along the top toolbar from any page.



The screenshot shows the phpMyAdmin interface for a MySQL database named 'flyres'. The 'Structure' tab is selected. On the left, there's a tree view of tables: account, airline, airport, employee, flight, passenger, reservation, and scheduled_flight. Below the tree, there's a table showing the structure of the 'reservation' table. At the bottom, there's a 'Create new table' form with 'Table name' set to 'reservation' and 'Number of columns' set to '4'. The 'SQL' tab is highlighted with a blue arrow.

2. On the SQL query screen input the following:

```
SELECT SUM(revenue) AS month_revenue  
FROM reservation  
WHERE booking_date LIKE [insert date parameter]
```

3. The **date parameter** needs to be entered in the following format:

- 'YYYY-MM-%' example: '2023-04-%' will return the total revenue generated in April 2023.
- The date must be enclosed in the quotation marks. The % symbol is used to represent all days in the month and year provided.

The screenshot shows the phpMyAdmin interface for a database named 'flyres'. In the left sidebar, under the 'Tables' section, there are several tables listed: account, airline, airport, employee, flight, passenger, reservation, and scheduled_light. The 'reservation' table is currently selected. In the main query editor area, a SQL query is entered:

```
SELECT SUM(revenue)
FROM reservation
WHERE booking_date LIKE '2023-04-%';
```

The 'Go' button at the bottom of the query editor is highlighted with a blue arrow.

4. After inputting the command and appropriately providing the date parameter, click 'Go' to execute.
5. The total revenue for the month and year provided will be calculated.

The screenshot shows the results of the executed SQL query. The results table displays one row with the column name 'SUM(revenue)' and its value '4369.00'. A blue arrow points from the 'Go' button in the previous screenshot to this result row.

SUM(revenue)
4369.00