🛫 FL_Management Database Summary

1️⃣ Overview

The **FL_Management** database manages all operations related to flights, airports, aircrafts, passengers, ticketing, finances, and admin security.
It enforces **referential integrity**, **business logic**, and **automation** through **foreign keys, triggers, stored procedures, and functions**.

2️⃣ Entities (Tables) and Their Roles

Table	Primary Key	Purpose	Key Attributes (Summary)
Flight	`Flight_No`	Stores flight details and timing information	`Dept_Time`, `Arr_Time`, `Duration (derived)`
Airport	`Airport_ID`	Information about airports	`Name`, `Location`
Aircraft	`Aircraft_ID`	Holds aircraft details	`Model`, `Capacity`, `Manufacturer`
Tickets	`Ticket_ID`	Ticket booking information linked to flights and passengers	`Class`, `Seat_No`, `Booking_Date`
Passenger	`Passenger_ID`	Stores passenger info	`Name (First, Last)`, `Gender`, `Nationality`, `Contact_No (multivalued)`, `Total_Tickets`

Finance	`Transaction_ID`	Transaction data for ticket payments	`Amount`, `Date`, `Transaction_Type`, `Passenger_ID`, `Ticket_ID`
Admin	`Admin_ID`	Authorized system users	`Roles (Employee/Manager/CEO)`, `Username`, `Password`
2FA *(Weak Entity)*	None (depends on `Admin_ID`)	Stores OTP-based two-factor authentication details	`Contact_No`, `OTP`

Data inserted: ~10 sample records per table, with realistic entries (flights, airports, passengers, admins, etc.) ensuring relational consistency.

3️⃣ Relationships Implemented

- Each **flight** departs and arrives at an **airport** (2 foreign keys).
- Each **aircraft** operates **multiple flights** (1:N).
- A **manager admin** manages all flight details.
- Each **ticket** is booked for one **flight** and one **passenger**.
- A **passenger** can book multiple tickets (with `Booking_Date`).
- **Finance** connects **passengers** and **tickets** (payment details).
- **Admin** is secured by **2FA (weak entity)** via OTP verification.

4️⃣ Functions (3 Total)

Function Name	Purpose	Test Example

`GetFlightDuration(flight_no)`	Calculates flight duration in minutes from departure and arrival times.	`SELECT GetFlightDuration('FL001');`
`CountPassengerTickets(passenger_id)`	Counts total tickets booked by a passenger.	`SELECT CountPassengerTickets('P01');`
`TotalRevenueByPassenger(passenger_id)`	Calculates total payment made by a passenger across all bookings.	`SELECT TotalRevenueByPassenger('P01');`

5️⃣ Stored Procedures (3 Total)

Procedure Name	Purpose	Test Example
`BookTicket(ticket_id, flight_no, passenger_id, class, seat, date)`	Inserts a new ticket and updates related info.	`CALL BookTicket('T12','FL001','P01','Eco','16A','2025-10-02');`

`AddPayment(trans_id, passenger_id, ticket_id, amount, date, type)`	Adds a transaction record into Finance.	`CALL AddPayment('F13','P01','T01',500,'2025-10-01','Card');`
`UpdateFlightTimes(flight_no, dept, arr)`	Updates flight times safely and keeps data consistent.	`CALL UpdateFlightTimes('FL001','2025-10-01 11:00:00','2025-10-01 15:30:00');`

---

6 Triggers (3 Total)

Trigger Name	Table	When	Purpose
`trg_UpdateFlightDuration`	`Flight`	BEFORE UPDATE	Automatically updates the `Duration` column whenever `Dept_Time` or `Arr_Time` changes.
`trg_ValidateFinanceAmount`	`Finance`	BEFORE INSERT	Validates that the inserted `Amount` is greater than zero. Throws an error if invalid.
`trg_UpdateTicketsAfterInsert/Delete`	`Tickets`	AFTER INSERT / DELETE	Automatically increments or decrements the `Passenger.Total_Tickets`field whenever tickets are booked or deleted.

---

7 Testing Summary

To test all logic in a clean order:

1. **Insert flights, passengers, and tickets.**
2. Use **stored procedures** to add bookings and payments.
3. Run **functions** to fetch computed data (`Duration`, `Revenue`, etc.).
4. Update or delete entries to **see triggers fire automatically.**
5. Verify consistency with `SELECT * FROM` queries across tables.

8️⃣ Highlights

✅ Fully normalized relational design (3NF)
✅ Derived attributes & constraints handled via functions and triggers
✅ Data integrity via foreign keys and validation triggers
✅ Automation via stored procedures
✅ Secure login structure using Admin + 2FA

SYNTAX:

FUNCTIONS:

a) GetFlightDuration

Calculates total flight duration (in minutes) based on Departure and Arrival times.

```
DELIMITER $$

CREATE FUNCTION GetFlightDuration(flight_no VARCHAR(10))

RETURNS INT

DETERMINISTIC

BEGIN

    DECLARE duration INT;

    SELECT TIMESTAMPDIFF(MINUTE, Dept_Time, Arr_Time)

    INTO duration
```

```
    FROM Flight
    WHERE Flight_No = flight_no;
    RETURN duration;
END$$
DELIMITER ;
```

Test:

```
SELECT GetFlightDuration('FL001');
```

(b) CountPassengerTickets

Returns the total number of tickets a passenger has booked.

```
DELIMITER $$
CREATE FUNCTION CountPassengerTickets(pid VARCHAR(10))
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE ticket_count INT;
    SELECT COUNT(*) INTO ticket_count
    FROM Tickets
    WHERE Passenger_ID = pid;
    RETURN ticket_count;
END$$
```

```
DELIMITER ;
```

Test:

```
SELECT CountPassengerTickets('P01');
```

(c) TotalRevenueByPassenger

Calculates total payment amount made by a specific passenger.

```
DELIMITER $$

CREATE FUNCTION TotalRevenueByPassenger(pid VARCHAR(10))

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

    DECLARE total DECIMAL(10,2);

    SELECT SUM(Amount) INTO total

    FROM Finance

    WHERE Passenger_ID = pid;

    RETURN IFNULL(total, 0);

END$$

DELIMITER ;
```

Test:

```
SELECT TotalRevenueByPassenger('P01');
```

⚙️2️⃣ STORED PROCEDURES

(a) BookTicket

Inserts a new ticket record.

```
DELIMITER $$

CREATE PROCEDURE BookTicket(

    IN tid VARCHAR(10),

    IN flight_no VARCHAR(10),

    IN pid VARCHAR(10),

    IN t_class VARCHAR(20),

    IN seat VARCHAR(10),

    IN book_date DATE

)

BEGIN

    INSERT INTO Tickets (Ticket_ID, Flight_No, Passenger_ID, Class,
Seat_No, Booking_Date)

    VALUES (tid, flight_no, pid, t_class, seat, book_date);

END$$

DELIMITER ;
```

Test:

```
CALL BookTicket('T12','FL001','P01','Eco','16A','2025-10-02');
```

(b) AddPayment

Adds a finance transaction for a ticket.

```
DELIMITER $$

CREATE PROCEDURE AddPayment(

    IN trans_id VARCHAR(10),

    IN pid VARCHAR(10),

    IN tid VARCHAR(10),

    IN amt DECIMAL(10,2),

    IN pay_date DATE,

    IN trans_type VARCHAR(10)

)

BEGIN

    INSERT INTO Finance (Transaction_ID, Passenger_ID, Ticket_ID,
Amount, Date, Transaction_Type)

    VALUES (trans_id, pid, tid, amt, pay_date, trans_type);

END$$

DELIMITER ;
```

Test:

```
CALL AddPayment('F13','P01','T01',500,'2025-10-01','Card');
```

(c) UpdateFlightTimes

Updates flight departure and arrival times.

```
DELIMITER $$

CREATE PROCEDURE UpdateFlightTimes(

    IN flight_no VARCHAR(10),

    IN new_dept DATETIME,

    IN new_arr DATETIME

)

BEGIN

    UPDATE Flight

    SET Dept_Time = new_dept,

        Arr_Time = new_arr

    WHERE Flight_No = flight_no;

END$$

DELIMITER ;
```

Test:

```
CALL UpdateFlightTimes('FL001','2025-10-01 11:00:00','2025-10-01
15:30:00');
```

⚡3️⃣TRIGGERS

(a) Auto-update Flight Duration

Automatically updates duration (in minutes) when flight times are changed.

```
-- Add derived Duration column if not added already

ALTER TABLE Flight ADD COLUMN Duration INT;


DELIMITER $$

CREATE TRIGGER trg_UpdateFlightDuration

BEFORE UPDATE ON Flight

FOR EACH ROW

BEGIN

    SET NEW.Duration = TIMESTAMPDIFF(MINUTE, NEW.Dept_Time,
NEW.Arr_Time);

END$$

DELIMITER ;
```

Test:

```
UPDATE Flight

SET Dept_Time='2025-10-01 12:00:00', Arr_Time='2025-10-01 16:15:00'

WHERE Flight_No='FL001';
```

```
SELECT Flight_No, Duration FROM Flight WHERE Flight_No='FL001';
```

(b) Validate Finance Amount

Prevents inserting finance records with invalid amounts.

```
DELIMITER $$

CREATE TRIGGER trg_ValidateFinanceAmount

BEFORE INSERT ON Finance

FOR EACH ROW

BEGIN

    IF NEW.Amount <= 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Amount must be greater than 0';

    END IF;

END$$

DELIMITER ;
```

Test:

```
CALL AddPayment('F15','P01','T01',0,'2025-10-01','Cash');  -- will
throw error
```

(c) Auto-update Passenger Total_Tickets

Keeps passenger ticket count updated automatically.

```
-- Add Total_Tickets column if not added already

ALTER TABLE Passenger ADD COLUMN Total_Tickets INT DEFAULT 0;


-- After INSERT Trigger

DELIMITER $$

CREATE TRIGGER trg_UpdateTicketsAfterInsert

AFTER INSERT ON Tickets

FOR EACH ROW

BEGIN

    UPDATE Passenger

    SET Total_Tickets = Total_Tickets + 1

    WHERE Passenger_ID = NEW.Passenger_ID;

END$$

DELIMITER ;


-- After DELETE Trigger

DELIMITER $$

CREATE TRIGGER trg_UpdateTicketsAfterDelete

AFTER DELETE ON Tickets

FOR EACH ROW

BEGIN
```

```
    UPDATE Passenger
    SET Total_Tickets = Total_Tickets - 1
    WHERE Passenger_ID = OLD.Passenger_ID;
END$$
DELIMITER ;
```

Test:

```
CALL BookTicket('T13','FL002','P01','Business','4A','2025-10-02');
SELECT Passenger_ID, Total_Tickets FROM Passenger WHERE
Passenger_ID='P01';

DELETE FROM Tickets WHERE Ticket_ID='T13';
SELECT Passenger_ID, Total_Tickets FROM Passenger WHERE
Passenger_ID='P01';
```

🧩1️⃣ Airport

Airport_ID	Name	Location
A001	Indira Gandhi Intl	Delhi
A002	Chhatrapati Shivaji Intl	Mumbai
A003	Kempegowda Intl	Bangalore
A004	Netaji Subhas Chandra Bose Intl	Kolkata
A005	Chennai Intl	Chennai
A006	Rajiv Gandhi Intl	Hyderabad
A007	Cochin Intl	Kochi
A008	Jaipur Intl	Jaipur
A009	Goa Dabolim Airport	Goa
A010	Pune Intl	Pune

✈️2️⃣ Aircraft

Aircraft_ID	Model	Capacity	Manufacturer
AC001	A320	180	Airbus
AC002	A350	300	Airbus
AC003	B737	160	Boeing
AC004	B787	250	Boeing
AC005	E190	100	Embraer
AC006	ARJ21	90	Comac
AC007	A321neo	200	Airbus
AC008	B777	350	Boeing
AC009	E195	120	Embraer

AC010	C919	150	Comac

🕒3 Flight

Flight_No	Dept_Time	Arr_Time	Duration	Dept_Airport	Arr_Airport	Aircraft_ID
FL001	2025-10-01 08:00:00	2025-10-01 10:00:00	120	A001	A002	AC001
FL002	2025-10-01 09:00:00	2025-10-01 12:00:00	180	A002	A003	AC002
FL003	2025-10-01 11:00:00	2025-10-01 13:30:00	150	A003	A004	AC003
FL004	2025-10-01 14:00:00	2025-10-01 16:00:00	120	A004	A005	AC004
FL005	2025-10-01 07:00:00	2025-10-01 09:00:00	120	A005	A006	AC005
FL006	2025-10-01 17:00:00	2025-10-01 19:30:00	150	A006	A007	AC006
FL007	2025-10-01 10:00:00	2025-10-01 13:00:00	180	A007	A008	AC007
FL008	2025-10-01 12:00:00	2025-10-01 14:00:00	120	A008	A009	AC008
FL009	2025-10-01 18:00:00	2025-10-01 20:30:00	150	A009	A010	AC009
FL010	2025-10-01 06:00:00	2025-10-01 08:00:00	120	A010	A001	AC010

🎟️4 Tickets

Ticket_ID	Flight_No	Passenger_ID	Class	Seat_No	Booking_Date
T001	FL001	P01	Economy	12A	2025-09-20

T002	FL001	P02	Business	4B	2025-09-20
T003	FL002	P03	Premium Eco	8C	2025-09-21
T004	FL003	P04	Economy	15D	2025-09-22
T005	FL004	P05	Business	3A	2025-09-22
T006	FL005	P06	First	1A	2025-09-23
T007	FL006	P07	Economy	14F	2025-09-23
T008	FL007	P08	Premium Eco	9B	2025-09-24
T009	FL008	P09	Business	5C	2025-09-24
T010	FL009	P10	Economy	17D	2025-09-25

🧍5️⃣Passenger

Passenger_ID	First_Name	Last_Name	Gender	Nationality	Contact_No	Total_Tickets
P01	Aarav	Mehta	M	Indian	9876543210	1
P02	Priya	Nair	F	Indian	9823456712	1
P03	Arjun	Verma	M	Indian	9123456789	1
P04	Kavya	Iyer	F	Indian	9988776655	1
P05	Rohan	Singh	M	Indian	9811122233	1
P06	Meera	Das	F	Indian	9898989898	1
P07	Aditya	Rao	M	Indian	9000011111	1
P08	Sneha	Gupta	F	Indian	9333312345	1

P09	Dev	Sharma	M	Indian	9123412341	1
P10	Ananya	Bose	F	Indian	9432112334	1

💳6 Finance

Transaction_ID	Passenger_ID	Ticket_ID	Amount	Date	Transaction_Type
F01	P01	T001	4500.00	2025-09-20	Card
F02	P02	T002	15000.00	2025-09-20	UPI
F03	P03	T003	7500.00	2025-09-21	Cash
F04	P04	T004	4000.00	2025-09-22	Card
F05	P05	T005	12000.00	2025-09-22	UPI
F06	P06	T006	20000.00	2025-09-23	Card
F07	P07	T007	3500.00	2025-09-23	UPI
F08	P08	T008	8000.00	2025-09-24	Card
F09	P09	T009	11000.00	2025-09-24	Cash
F10	P10	T010	3800.00	2025-09-25	UPI

🧑‍💼7 Admin

Admin_ID	Username	Password	Roles
AD001	emp_ravi	emp123	Employee
AD002	emp_sneha	emp456	Employee
AD003	mgr_rahul	mgr123	Airport_Manager

AD004	ceo_anil	ceo321	CEO
AD005	emp_karan	emp999	Employee
AD006	emp_kavya	emp777	Employee
AD007	mgr_neha	mgr999	Airport_Manager
AD008	emp_dev	emp555	Employee
AD009	ceo_vikram	ceo777	CEO
AD010	emp_isha	emp888	Employee

🔐8️⃣2FA (Weak Entity)

Admin_ID	Contact_No	OTP
AD001	9876500011	4512
AD002	9876500012	7634
AD003	9876500013	8899
AD004	9876500014	9988
AD005	9876500015	3321
AD006	9876500016	1122
AD007	9876500017	7755
AD008	9876500018	6644
AD009	9876500019	5500
AD010	9876500020	2211

✅ **All relationships hold true:**

- Every `Flight` references valid `Airport` and `Aircraft`.
- Every `Ticket` references valid `Flight` and `Passenger`.
- Every `Finance` record references valid `Passenger` and `Ticket`.
- Every `2FA` record references valid `Admin`.

Would you like me to give you this same data as ready-to-run `INSERT INTO` SQL statements (so you can populate your DB directly)?
That would let you build the full dataset with one click inside MySQL Workbench.