AIRPORT MANAGEMENT SYSTEM DATABASE

CONTENTS

1. PROJECT REVIEW

Airport facilities rely on a number of systems in order to perform some specific processes and I am creating a database as a combination of airlines and airport in which, not only our focus will be on the airline system but primarily my focus will be to create a database for the airport system as well. As Airport system is a database project that primarily deals with the management of the airport, airlines, passengers and employees working for an airport.

2. <u>PROJECT OVERVIEW :</u>

MY PROJECT HAS 4 MAIN MODULES AND IS FURTHER PROCEEDED WITH ITS ENTITIES.

AIRPORT -> CITY, AIRPORT, AIRLINES, AIRPORTCONTAINSAIRLINES , FLIGHT

EMPLOYEE -> EMPLOYEEDETAILS, EMPLOYEESALARY

PASSANGER -> PASSENGERDETAILS, PASSENGER , PASSENGERFLIGHT

TICKET -> TICKETDETAILS , TICKETBOOK , TICKETCANCEL

3. <u>ENTITIES BEFORE NORMALIZATION</u>

CITY

AirportName	STATE	COUNTRY

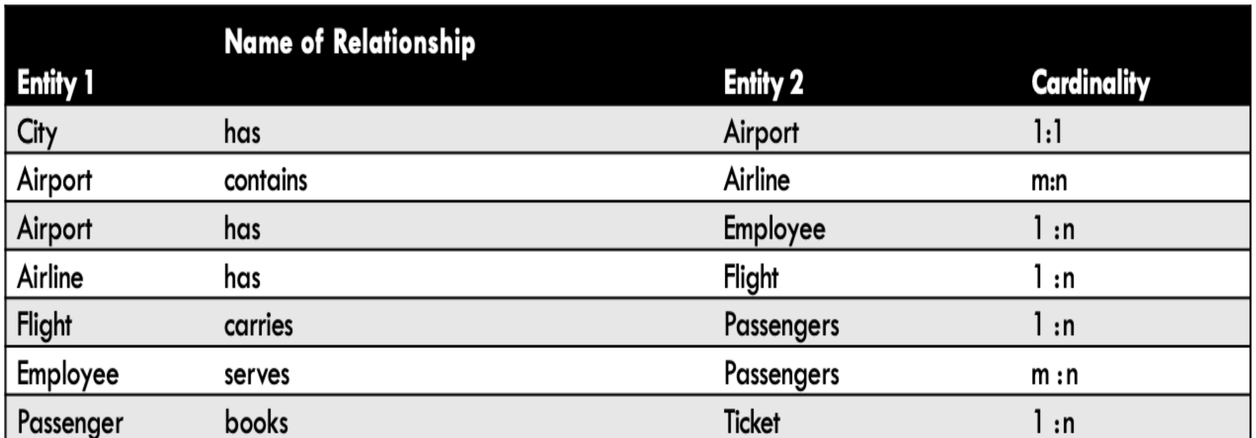
AIRPORT

CityName	STATE	COUNTRY

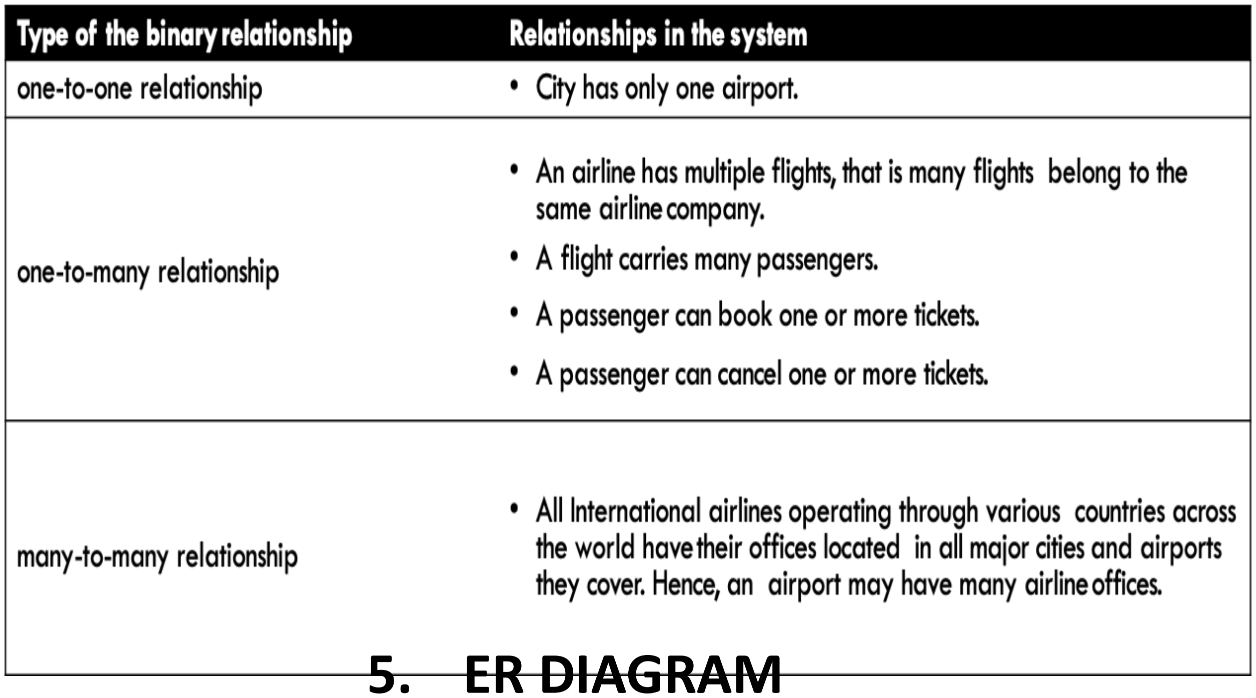
AIRLINE

4. ER/EER RELATIONSHIPS

ER diagram contains following relationships

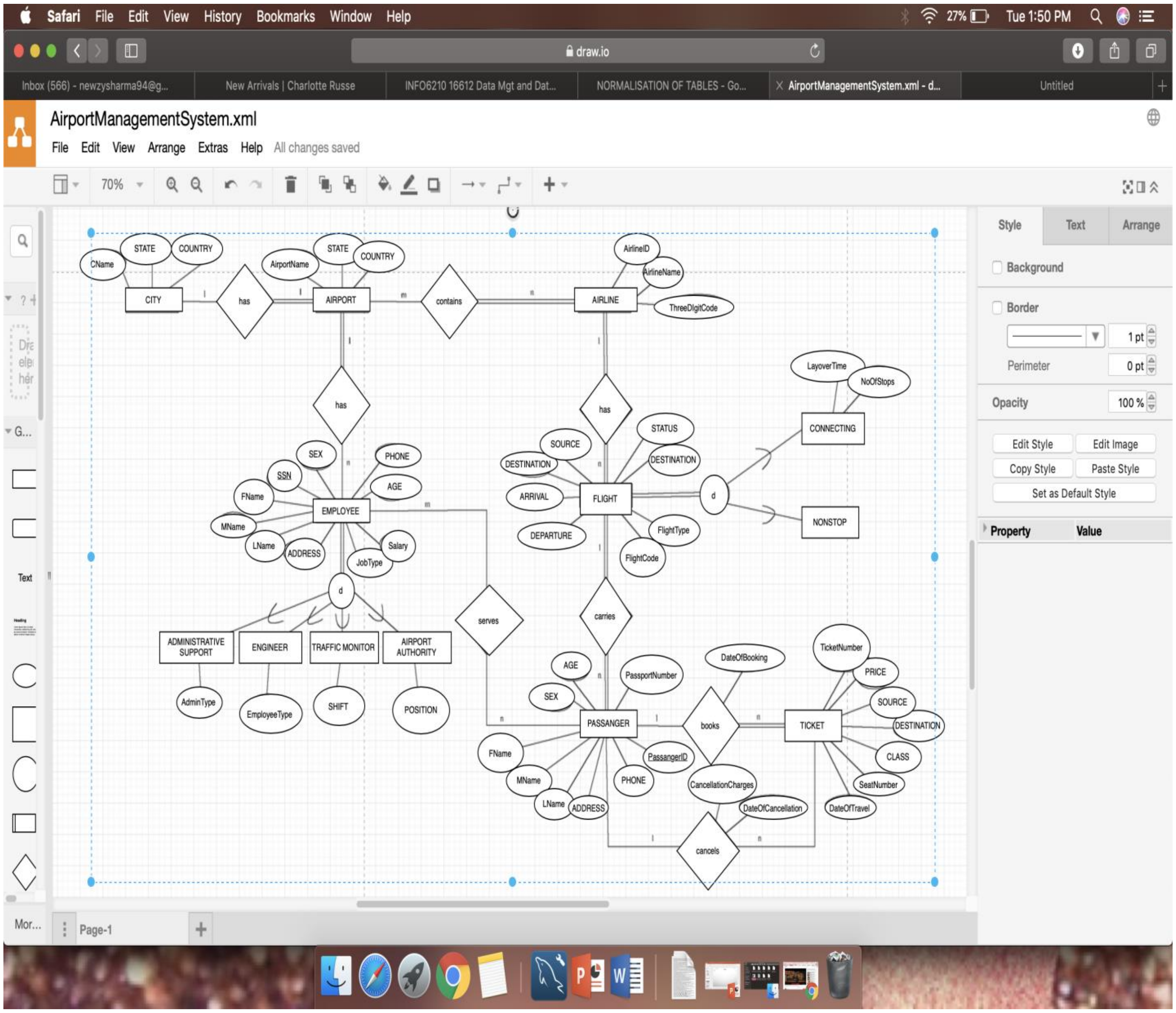


Entity 1	Name of Relationship	Entity 2	Cardinality
City	has	Airport	1:1
Airport	contains	Airline	m:n
Airport	has	Employee	1 :n
Airline	has	Flight	1 :n
Flight	carries	Passengers	1 :n
Employee	serves	Passengers	m :n
Passenger	books	Ticket	1 :n



Type of the binary relationship	Relationships in the system
one-to-one relationship	• City has only one airport.
one-to-many relationship	• An airline has multiple flights, that is many flights belong to the same airline company. • A flight carries many passengers. • A passenger can book one or more tickets. • A passenger can cancel one or more tickets.
many-to-many relationship	• All International airlines operating through various countries across the world have their offices located in all major cities and airports they cover. Hence, an airport may have many airline offices.

5. ER DIAGRAM



6. NORMALIZATION OF TABLES

<u>Definition:</u> Database normalization is the process of restructuring a relational database in accordance with a series of normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as an integral part of his relational model.

FUNCTIONAL DEPENDECIES	
PassportNumber -> FName, MName, LName, ADDRESS, PHONE, AGE, SEX	Violates 2NF
PassangerID ->FlightCode	Violates 2NF
DateOfBooking, SOURCE, DESTINATION, CLASS -> PRICE	Violates 3NF
DateOfCancellation -> CancellationCharges	Violates 3NF
JobType -> SALARY	Violates 3NF

7. TABLES AFTER NORMALIZATION

TABLES AFTER NORMALISATION

CITY (CityName, STATE, COUNTRY)

AIRPORT (AirportName, STATE, COUNTRY, CNAME) AIRLINE (AirlineID, AirlineName, ThreeDigitCode)

AIRPORTCONTAINSAIRLINES (AirlineID, AirportName)

FLIGHT (FlightCode, SOURCE, DESTINATION, ARRIVAL, DEPARTURE, STATUS, DURATION, FlightType, LayoverTime,NoOfStops, AIRLINEID)

PASSENGER (PassangerID, PassportNumber)

PASSENGERDETAILS(PASSPORTNO, FNAME, M, LNAME, ADDRESS, PHONE, AGE, SEX) PASSENGERFLIGHT (PassangerID, FLlightCode)

TICKETDETAILS (TicketNumber, SOURCE, DESTINATION,DateOfBooking,DateOfTravel,SeatNumber, CLASS,DateOfCancellation, PassangerID, PassportNumber)

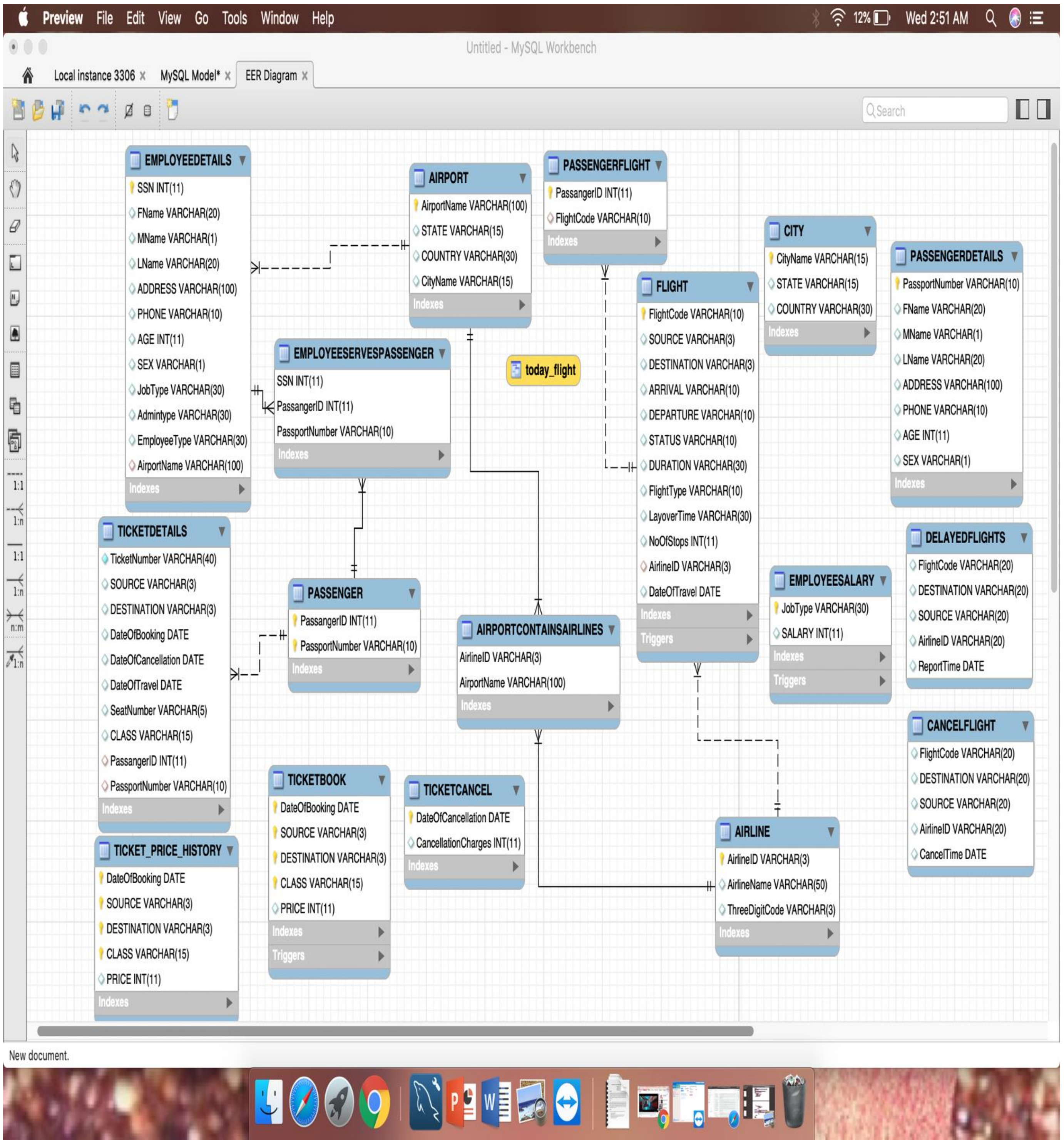
TICKETBOOK (DateOfBooking, SOURCE, DESTINATION, CLASS, PRICE)

TICKETCANCEL(DateOfCancellation, CancellationCharges)

EMPLOYEEDETAILS (SSN, FName, MName, LName, ADDRESS, PHONE, AGE, SEX, JobType, AdminType, EmployeeType, AirportName)

EMPLOYEESALARY(JobType, SALARY) EMPLOYEESERVESPASSANGER (SSN, PassangerID, PassportNumber)

8. EER AIRPORT MANAGEMNET SYSTEM



9. PRIVILEGES

DEFINITION: It is a special right, advantage, or immunity granted or available only to a particular person or group of people.

```
CREATE USER 'newzy'@'localhost'
IDENTIFIED BY 'newzy94';

CREATE USER 'bhagi'@'localhost'
IDENTIFIED BY 'bhagi18';

GRANT INSERT
ON airportmanagementsystem.EMPLOYEEDETAILS
TO 'newzy'@'localhost';

GRANT UPDATE
ON airportmanagementsystem.EMPLOYEEDETAILS
TO 'newzy'@'localhost';

GRANT DELETE
ON airportmanagementsystem.EMPLOYEEDETAILS
TO 'newzy'@'localhost';

GRANT SELECT
ON airportmanagementsystem.EMPLOYEEDETAILS
TO 'newzy'@'localhost';


GRANT SELECT
ON airportmanagementsystem.PASSENGERDETAILS
TO 'bhagi'@'localhost';
```

```
REVOKE SELECT
ON airportmanagementsystem.PASSENGERDETAILS
FROM 'bhagi'@'localhost';
```

10. INDEXES

1. CREATE FULLTEXT INDEX CityName_Index
   ON AIRPORT(CityName);

2. CREATE FULLTEXT INDEX Emplname_index
   ON EMPLOYEEDETAILS(LName);

3. CREATE FULLTEXT INDEX FnameIndex
   ON PASSENGERDETAILS(LName);

11. VIEWS

Definition: MySQL supports views, including updatable views. Views are stored queries that when invoked produce a result set. A view acts as a virtual table.

1. **View Name: today_flight**

-- **Goal:** This view is for the Airport Administrator Support (Help Desk). In this view employees can see all the relevant details of current date flight and inform the passengers.

```
CREATE VIEW today_flight
AS
SELECT
FlightCode,AirlineName,SOURCE,DESTINATION,DEPARTURE,ARRIVAL
From FLIGHT
INNER JOIN AIRLINE
ON FLIGHT.AirlineID =AIRLINE.AirlineID
WHERE Flight.DateOfTravel = curdate()
ORDER BY SOURCE,DEPARTURE;

SELECT * FROM today_flight;
```

12. PROCEDURE

Definition: Stored Procedure-> A procedure (often called a stored procedure) is a subroutine like a subprogram in a regular computing language, stored in database. A procedure has a name, a parameter list, and SQL statement(s)

1. **Stored procedure: get_passenger_details**

-- **Parameters**:
IN – passenger first name,
IN – passenger last name, I
N – airline name.,
IN – flight date

-- **Goal**: This is procedure is for the passengers to query flight details. They just need to give their first name, last name, flight code & flight date and they will get all the details about their upcoming flight.

```
DELIMITER //

CREATE PROCEDURE get_passenger_details
(
IN fname VARCHAR(50) , lname VARCHAR(50), fcode VARCHAR(50))
BEGIN
        SELECT concat_ws(' ', pd.FName, pd.LName) as 'Passenger
Name', f.FlightCode , f.AirlineID, f.SOURCE ,f.DESTINATION
,f.DEPARTURE, f.ARRIVAL
        FROM PASSENGERDETAILS pd
        INNER JOIN PASSENGER p
        ON pd.PassportNumber=  p.PassportNumber
        INNER JOIN PASSENGERFLIGHT pf
        ON pf.PassangerID = p.PassangerID
        INNER JOIN FLIGHT f
        ON pf.FlightCode = f.FlightCode
        INNER JOIN AIRLINE a
        ON f.AirlineID =a.AirlineID
        WHERE pd.FName = fname
    AND pd.LName =  lname
    AND f.FlightCode = fcode
    AND f.DateOfTravel = curdate() ;

END //

DELIMITER ;


CALL  get_passenger_details ( 'Vivek', 'Sharma' ,'QR2306');
```

2. **STORED PROCEDURE : get_flight_details**

```
DELIMITER //
CREATE PROCEDURE get_flight_details
(
        IN alname VARCHAR(50) ,
        src VARCHAR(50),
        dest VARCHAR(50)
)
BEGIN
                SELECT
al.AirlineName,f.SOURCE,f.DESTINATION,f.DEPARTURE,f.ARRIVAL
                From FLIGHT f
                INNER JOIN AIRLINE al
                ON f.AirlineID =al.AirlineID
                WHERE f.DateOfTravel = curdate()
                AND al.AirlineName = alname
                AND f.SOURCE = src
                AND f.DESTINATION = dest
                ORDER BY SOURCE,DEPARTURE;
END ;
//

CALL  get_flight_details ( 'Qatar', 'BOM' ,'DFW');
```

13. TRIGGER

Definition: trigger is a database object that is associated with a table. It will be activated when a defined action is executed for the table. The trigger can be executed when you run one of the following MySQL statements on the table: INSERT, UPDATE and DELETE and it can be invoked before or after the event.

-- Trigger, when flight is delayed it is logged into different table

1. TRIGGER : DELAYEDFLIGHTS

```
DELIMITER //
CREATE  TRIGGER DELAYEDFLIGHTS
AFTER INSERT
 ON FLIGHT
FOR EACH ROW
BEGIN
      INSERT INTO DELAYEDFLIGHTS (FlightCode, SOURCE, DESTINATION ,
AirlineID ,ReportTime)

VALUES(new.FlightCode,new.source,new.DESTINATION,new.AirlineID,n
ow());
END;
//
```

```
-- This is audit table for this trigger--
 CREATE TABLE DELAYEDFLIGHTS
   (          FlightCode VARCHAR(20),
              DESTINATION VARCHAR(20),
              SOURCE VARCHAR(20 ),
              AirlineID VARCHAR(20),
                                  ReportTime DATE
   );


INSERT INTO FLIGHT(FlightCode,
SOURCE,
DESTINATION ,
ARRIVAL ,
DEPARTURE ,
STATUS ,
DURATION ,
FlightType ,
LayoverTime ,
NoOfStops ,
AirlineID)
VALUES('AI127','BOM','DFW','02:10','03:15','Delayed','24hr','Connectin
g',3,1,'AI');

SELECT * FROM FLIGHT;
SELECT * FROM DELAYEDFLIGHTS;
```

2. TRIGGER : TICKET_PRICE_HISTORY

```
-- TRIGGER TO UPDATE 'TICKET_PRICE_HISTORY' TABLE WHEN THE
PRICE OF AN AIR TICKET IS UPDATED IN TICKETBOOK TABLE--

-- CREATING TABLE TICKET_PRICE_HISTORY--

CREATE TABLE TICKET_PRICE_HISTORY
(DateOfBooking DATE NOT NULL,
SOURCE VARCHAR(3) NOT NULL,
DESTINATION VARCHAR(3) NOT NULL,
CLASS VARCHAR(15) NOT NULL,
PRICE INT,
PRIMARY KEY(DateOfBooking, SOURCE, DESTINATION, CLASS));

-- CREATE A TRIGGER TICKET_PRICE_HISTORY --


DELIMITER //
CREATE  TRIGGER TICKET_PRICE_HISTORY
BEFORE UPDATE
ON TICKETBOOK
FOR EACH ROW
BEGIN
INSERT INTO TICKET_PRICE_HISTORY
VALUES(OLD.DateOfBooking, OLD.SOURCE, OLD.DESTINATION,
OLD.CLASS,OLD.PRICE);
END;
```

```
//

-- LET'S UPDATE A PRICE OR FARE OF AN AIR TICKET --

UPDATE TICKETBOOK
SET PRICE=150000
WHERE DateOfBooking = '2018-11-11'
AND SOURCE='BOM'
AND DESTINATION='DFW'
AND CLASS='ECONOMY'
```

3. **TRIGGER : UpdatedSalary**

```
-- TRIGGER TO UPDATE SALARY OF AN EMPLOYEE DEPENDING ON
JOBTYPE—


DELIMITER //
CREATE  TRIGGER UpdatedSalary
AFTER
INSERT
ON EMPLOYEESALARY
FOR EACH ROW
BEGIN
                CASE
                                    WHEN (JOBTYPE= 'Administrative Support')
THEN
                                     UPDATE EMPLOYEESALARY SET
SALARY=SALARY+SALARY*.10;
                                    WHEN (JOBTYPE= 'ENGINEER') THEN
                                     UPDATE EMPLOYEESALARY SET
SALARY=SALARY+SALARY*.05;
                                    WHEN (JOBTYPE= 'TRAFFIC MONITOR')
THEN
                                     UPDATE EMPLOYEESALARY SET
SALARY=SALARY+SALARY*.25;
                                    WHEN (JOBTYPE= 'AIRPORT AUTHORITY')
THEN
                                    UPDATE EMPLOYEESALARY SET
SALARY=SALARY+SALARY*.45;
                END CASE;
END ;//
```

14. STORED FUNCTION

-- Stored function: Availability
-- **Parameters**: Input is an integer i.e. availability and output return varchar availability as ‘YES’ or ‘No’
-- **Goal**: We have created this stored function to determine availability. If the availability is greater than 0, then Yes is returned else No.

```
DELIMITER //
CREATE FUNCTION Availability(a int) RETURNS VARCHAR(10)
BEGIN
                DECLARE available varchar(10);
                IF a > 0 THEN
                SET available = 'YES';
                ELSE
                SET available = 'NO';
        END IF;
                RETURN (available);
END //
DELIMITER ;
```