

# Flight Management Database

*Everything starts from an idea with general information*



## Introduction:

Databases are important for any company, organization, hospitals, hotels, etc. The concept of the database is to help “CEO, employer, employee” to reach the information they need easily. That's why we chose the flight management system .Nowadays Air flights are important in our life that should have a database that manages, simplifies and organizes data.

## Objective:

Flight management database could have a huge amount of data that is needed for the workers and the passengers. This project was made to give a simple holistic view of the data that could be in terms of flight management, take off from different airports to different destinations.

1. Eases information access
2. Airlines can increase their sales.
3. Track the traveler by knowing if he checked in and boarded for safety reasons.
4. Facilitate online payment for users
5. Facilitate the organization for several parties

## Project Description:

The project shows some strongly required information from a person that wants to take a trip somewhere and information that he should know about his trip. Starting from the airline that plane belongs to, the pilot who is driving, the general information about the flight. Moreover, for the employee, they need to know the passengers, a proof of their payment for boarding, and their boarding pass ticket.

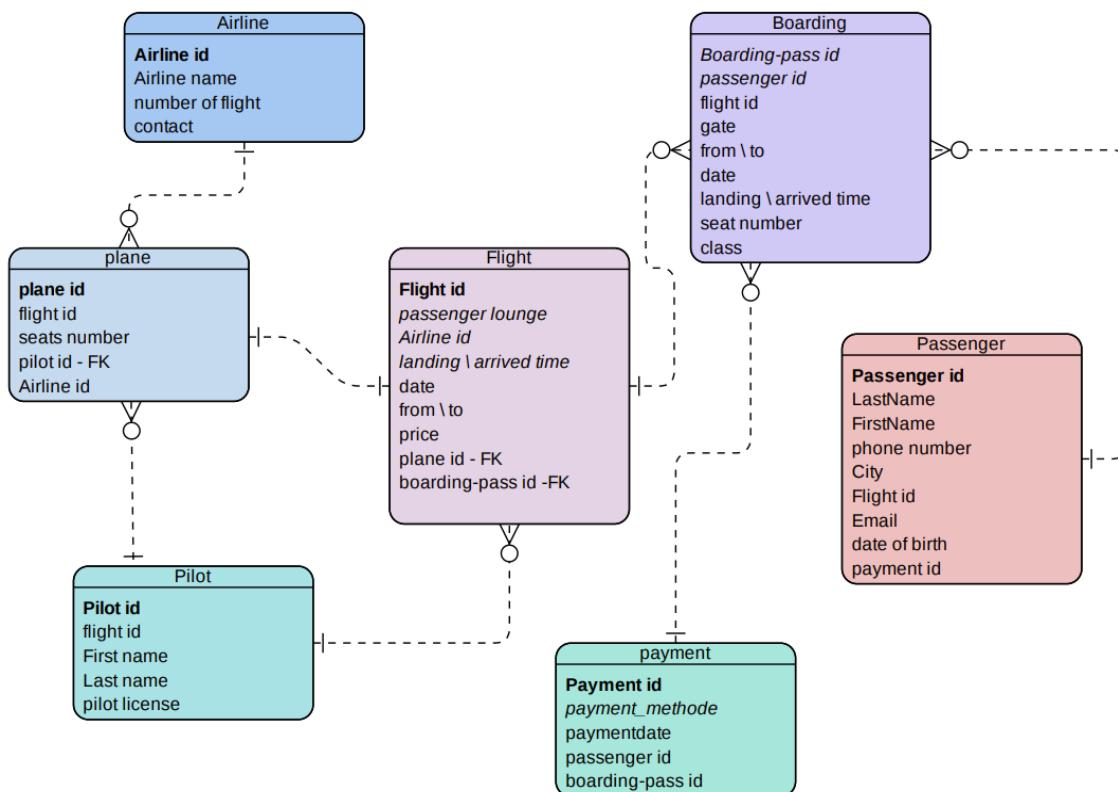
## Conceptual Design:

### ( Phase I )

Initial Business Rules:

1. Each gate has one flight at least in a day
2. Each passenger has 1 seat in a flight not more
3. Numbers of boarding depends on the number of the seats
4. Cancel the ticket before the flight time at least 24 hours

Initial ERD < Entities, Attributes >:



## ( Phase II )

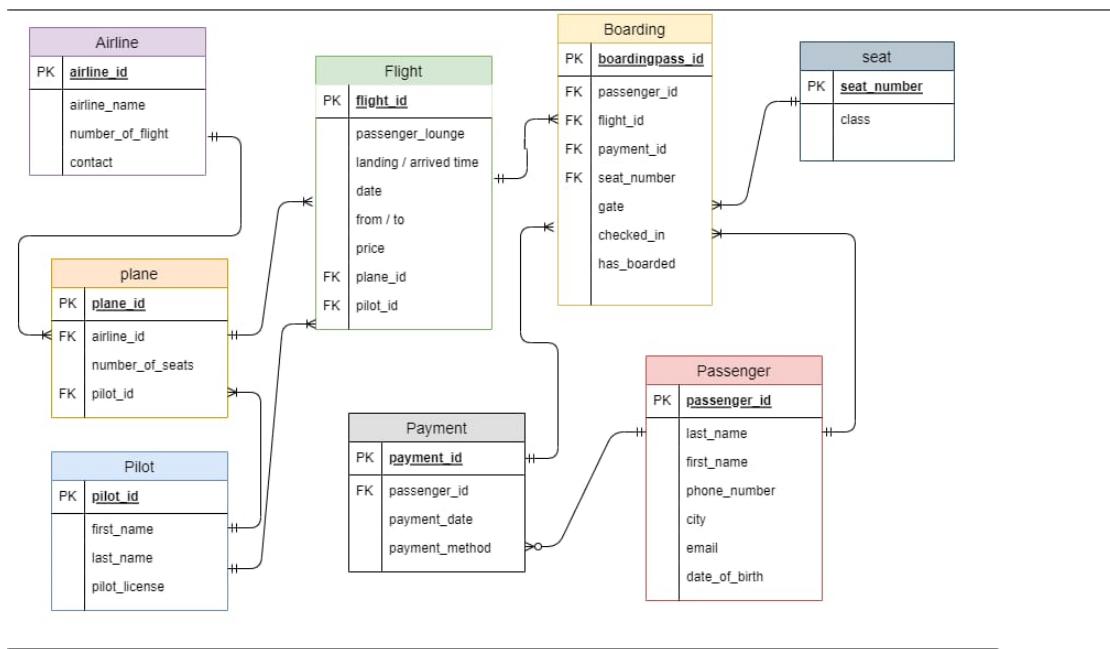
Final Business Rules:

1. Pilots can have at least one flight a day.
2. Airlines should fly at least one plane a day.
3. Planes take at least one flight a day.
4. The boarding pass will contain only one seat.
5. Passengers can have a lot of boarding in different flights.
6. Flights can have a different number of boarding.
7. Each passenger can make many payments.

Trigger:

1. Updating flight will be in “Landing/Arrived Time, Date, Passenger Lounge”
2. Boarding should track passengers if they traveled or not.

Final ERD < Entities, Attributes >:



## Normalize the Airport Database:

### 1. Table Pilot

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency.*

### 2. Table Plane

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency.*

### 3. Table Flight

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency.*

### 4. Table Passenger

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency.*

### 5. Table Payment

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency.*

### 6. Table Boarding

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: transitive dependency between seat\_number and class, so create a new table name Seat that contains seat\_number as a primary key.*

### 7. Table Seat

*1NF: Applied, because there are no multivalued attributes, composite attributes or repeating groups.*

*2NF: Applied, because there are no partial dependencies.*

*3NF: Applied, because there is no transitive dependency*

## **DDL Commands:**

DDL Commands were used many times; starting from creating tables, identifying the primary key for each table, foreign key. Using SQL Constraint “ Not Null, Unique, Default”, Moreover, data types were varied between “ Numeric - Int, Decimal -, Text - Varchar (Using different size)-, Date/Time - Date, Time - , SET ” . Furthermore, dropping tables is used many times for many reasons and also truncates tables - if we enter the wrong values it was used to delete them from the table -. At the end the command “select table” to see the last results for the database.

**In the appendix section the DDL codes.**

## **DML Commands:**

DML Command was strongly needed to complete the data. Insert command used to insert every value in the table, highly needed “Multiple rows” to save time and less repeated rows. Also, an update command was used in trigger to update some information about the flights, there was some delete command used to change the wrong price for a flight, change the payment id for the passenger, and it was used to change the wrong boarding pass id. As a summary for the DML Commands, the most used is select, to see the tables, the data in the table, and it is used to show each detail about the flight, passenger, payment, boarding.

**In the appendix section the DML codes.**

## Codes:

For each query we defined it as “Use flight\_management”, in the following order for creating:

### Airline table:

```
Use flight_management;

CREATE TABLE airline(
    airline_id INT PRIMARY KEY ,
    airline_name VARCHAR(100) NOT NULL,
    number_of_flights INT,
    contact VARCHAR(255) UNIQUE
);
```

### Pilot table:

```
CREATE TABLE pilot(
    Pilot_id INT PRIMARY KEY,
    first_name CHAR(50) NOT NULL,
    last_name CHAR(50) NOT NULL,
    pilot_license CHAR(255) UNIQUE NOT NULL
);
```

### Plane table:

```
CREATE TABLE plane(
    plane_id INT PRIMARY KEY,
    number_of_seats INT NOT NULL,
    pilot_id INT,
    airline_id INT,
    FOREIGN KEY (pilot_id) REFERENCES pilot(pilot_id),
    FOREIGN KEY (airline_id) REFERENCES airline(airline_id)
);
```

Flight table:

```
CREATE TABLE flight(
    flight_id INT PRIMARY KEY,
    passenger_lounge VARCHAR(60) DEFAULT 'Un defined',
    landing_time TIME DEFAULT '00:00:00',
    arrived_time TIME DEFAULT '00:00:00',
    from_airport VARCHAR(60) NOT NULL,
    to_airport VARCHAR(60) NOT NULL,
    flight_date DATE NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    plane_id INT,
    pilot_id INT,
    FOREIGN KEY (plane_id) REFERENCES plane(plane_id),
    FOREIGN KEY (pilot_id) REFERENCES pilot(pilot_id)
);
```

Passenger table:

```
CREATE TABLE passenger(
    passenger_id INT PRIMARY KEY,
    lastName CHAR(50) NOT NULL,
    firstName CHAR(50) NOT NULL,
    date_of_birth DATE NOT NULL,
    phone_number INT UNIQUE NOT NULL,
    city CHAR(50),
    email CHAR(100)
);
```

Payment table:

```
CREATE TABLE payment(
    payment_id INT PRIMARY KEY,
    payment_method CHAR(100) NOT NULL,
    payment_date DATE NOT NULL,
    passenger_id INT,
```

FOREIGN KEY (passenger\_id) REFERENCES passenger(passenger\_id)

);

Seat table:

CREATE TABLE seat (

seat\_number VARCHAR(30),

class VARCHAR(30),

PRIMARY KEY (seat\_number)

);

Boarding table:

CREATE TABLE boarding(

boardingpass\_id INT PRIMARY KEY,

gate INT,

passenger\_id INT,

flight\_id INT,

payment\_id INT,

checked\_in VARCHAR(20),

has\_boarded VARCHAR(20),

seat\_number VARCHAR(30) ,

FOREIGN KEY (seat\_number) REFERENCES seat(seat\_number),

FOREIGN KEY (passenger\_id) REFERENCES passenger(passenger\_id),

FOREIGN KEY (flight\_id) REFERENCES flight(flight\_id),

FOREIGN KEY (payment\_id) REFERENCES payment(payment\_id)

);

Trigger table:

CREATE TABLE changed\_flights(

flight\_id INT,

landing\_time TIME DEFAULT'00:00:00',

arrived\_time TIME DEFAULT'00:00:00',

flight\_date DATE NOT NULL,

passenger\_lounge VARCHAR(60) DEFAULT 'Un defined'

);

CREATE TRIGGER before\_update\_flights

BEFORE UPDATE

ON flight

FOR EACH ROW

INSERT INTO changed\_flights(flight\_id,landing\_time, arrived\_time,flight\_date,passenger\_lounge)

VALUES(OLD.flight\_id,OLD.landing\_time, OLD.arrived\_time, OLD.flight\_date, OLD.passenger\_lounge);

create table boarding\_safe(

boardingpass\_id INT,

message VARCHAR(90)

);

DELIMITER \$\$

CREATE TRIGGER completed\_flights

after insert

ON boarding

FOR EACH ROW

begin

if (NEW.checked\_in='Checked' AND NEW.has\_boarded='Boarded')

THEN

INSERT INTO boarding\_safe(boardingpass\_id,message)

values(NEW.boardingpass\_id,'traveled');

end if;

end \$\$

## Insertion Values:

For each query we defined it as “Use flight\_management”, in the following order for creating:

Airline:

Use flight\_management;

```
insert into airline(airline_id, airline_name, number_of_flights, contact) values  
(101, 'Saudi Arabian Airline', 5, 'Email: saudiAr@org.sa, Number: +9963224789135'),  
(223, 'FlyNas', 3, 'Email: FlyN_airline@org.sa, Number: +9661596672381');
```

Pilot:

```
insert into pilot (pilot_id, first_name, last_name, pilot_license) values  
(0217, 'Rajaa', 'Alharbi', 4110081),  
(0411, 'Layan', 'Baroum', 4110083),  
(0512, 'Ghadi', 'Nouri', 4110085),  
(0806, 'Zainab', 'Tolah', 4110100),  
(2212, 'Maryam', 'Alwedyani', 4110155),  
(0227, 'Razan', 'Thamer', 4110158),  
(0419, 'Tamim', 'Alghamdi', 4110027),  
(0922, 'Meshari', 'Alzahrani', 4110096);
```

Plane:

```
insert into plane (plane_id, number_of_seats, pilot_id, airline_id) values  
(1005, 264, 217, 101),  
(2007, 190, 512, 223),  
(3011, 220, 806, 223),  
(4015, 150, 2212, 101),  
(5017, 100, 411, 101),
```

(7356, 193, 0227, 223),

(9692, 250, 0922, 101),

(5284, 178, 0419, 101);

**Flight:**

insert into flight (flight\_id, passenger\_lounge, landing\_time, arrived\_time, from\_airport, to\_airport, flight\_date, price, plane\_id, pilot\_id) values

(1043, '1A', '102500', '193500', 'King Abdu alziz International Airport', 'Albury Airport', '2022-04-14', '3850.88', 3011, 0806 ),

(5189, '2B', '023500', '080000', 'King Abdu alziz International Airport', 'Tokyo International Airport', '2022-05-14', '5700 ', 5017, 0411 ),

(0982, '7D', '050500', '141000', 'Prince Mohammad Bin Abdulaziz International Airport ', 'Rome Fiumicino Airport', '2022-05-26', '4010.20 ', 1005, 0217),

(2117, '5C', '200500', '144000', 'King Abdu alziz International Airport', 'Domodedovo Mikhail Lomonosov International Airport', '2022-05-13', '2750 ', 2007, 0512),

(3354, '3E', '061000', '115500', 'King Abdu alziz International Airport', 'Newark International Airport', '2022-08-20', '4100.35 ', 4015, 2212),

(2268, '7B', '113000', '031500', 'King Abdu alziz International Airport', 'Vancouver International Airport', '2022-05-13', '7410.75', 7356, 0227 ),

(6507, '9A', '154500', '083000', 'King Khalid International Airport', 'Geneva Airport', '2022-06-28', '4330.65 ', 5284, 0419),

(4186, '2F', '061500', '014500', 'Bahrain International Airport', 'Athens International Airport', '2022-07-11', '2980', 9692, 0922);

**Passenger:**

insert into passenger (passenger\_id, lastName, firstName, phone\_number, city, email, date\_of\_birth) values

(1003758976, 'Aljuhani', 'Fahad', 0547166212, 'Medina', 'AljuhaniF@gmail.com', '1992-03-15' ),

(1174921729, 'Alharbi', 'Lama', 0551378284, 'Medina', 'LamaHrb@gmail.com', '1997-11-06'),

(1040432195, 'Akbar', 'Waseem', 0542807390, 'Jeddah', 'Waseem98@hotmail.com', '1998-05-22'),

(1110952865, 'Alwaleed', 'Jehan', 0531810063, 'Jeddah', 'AlwaleedJehan@hotmail.com', '2000-12-18' ),

(1000277715, 'Alsalem', 'Anas', 0561632615, 'Jeddah', 'Alsalem.Anas@gmail.com', '1988-07-19'),

(1012761134, 'Adnan', 'Altaf', 0558666579, 'Jeddah', 'AltafAD@gmail.com', '1994-06-09'),

(1509971883, 'Aldeeb', 'Saleh', 0581737821, 'Jeddah', 'Aldeebssd@gmail.com', '1989-06-11'),

(1249623525, 'Almena', 'Tarneem', 0555284517, 'Riyadh', 'trnemena@gmail.com', '1992-08-27'),

(1979826959, 'Al Qurashi', 'Rakan', 0568264959, 'Khobar', 'rakanqrshi@gmail.com', '1979-11-07');

Payment:

```
insert into payment(payment_id, payment_method, payment_date, passenger_id) values
```

(51, 'Mastercard', '2022-08-02', 1040432195 ), /\*Jed - NY, Waseem\*/

(61, 'Apple pay', '2022-05-13', 1003758976),/\*med - Italy, Fahad\*/

(84, 'Mada', '2022-04-30', 1000277715),/\*Jed - Russ, Anas\*/

(34, 'Visa', '2022-04-04', 1110952865),/\*Jed - Astra, jehan\*/

(93, 'Apple pay', '2022-05-01', 1012761134),/\*Jed- JPN, Altaf\*/

(14, 'Mada', '2022-05-24', 1174921729 ), /\*Med - Italy, lama\*/

(44, 'Mada', '2022-04-08', 1509971883),/\*Jed - CND, Saleh\*/

(18, 'Apple pay', '2022-04-22', 1249623525),/\*RDH - SWTZ, Tarneem\*/

(62, 'Visa', '2022-05-11', 1979826959);/\*KBR - GRC, Rakan\*/

Seat:

```
insert into seat(seat_number, class) values
```

('13A', 'Economy Class'),

('4B', 'First Class'),

('35C', 'Business Class'),

('7D', 'Economy Class'),

('28A', 'Business Class'),

('44B', 'First Class'),

('20C', 'Business Class'),

('49D', 'Economy Class'),

('12E', 'First Class');

Boarding:

```
insert into boarding(boardingpass_id, gate, passenger_id, flight_id, payment_id, checked_in, has_boardsed, seat_number) values
```

(8739, 1, 1110952865 , 1043, 34, 'Checked', 'Boarded','4B'), /\*Jed - ASTR, jehan\*/

(4186, 2, 1012761134 , 5189, 93, 'Checked', 'Did not Boarded','35C'), /\*Jed - JPN, Altaf\*/

(3488, 5, 1040432195, 3354, 51, 'Checked', 'Boarded','12E'),/\*Jed - NY, Waseem\*/

```
(2343, 7, 1003758976, 982, 61, 'Checked', 'Boarded','12E'), /*Med - Italy, Fahad*/
(5266, 3, 1000277715, 2117, 84, 'Checked', 'Boarded','49D'),/*Jed - Rus, Anas*/
(6450, 7, 1174921729, 982, 14, 'Did not checked', 'Did not Boarded','44B'), /*Med - Italy, Lama*/
(6382, 7, 1509971883, 2268, 44, 'Checked', 'Boarded','49D' ),/*Jed-Cnd, Saleh*/
(8512, 9, 1249623525, 6507, 18, 'Did not checked', 'Did not Boarded','44B'),/*Rdh - Swtz, Tarneem*/
(7002, 2, 1979826959, 4186, 62, 'Checked', 'Boarded','7D');/*Khb - Grc, Rakan*/
```

Trigger:

```
update flight

set landing_time = '113000',
arrived_time = '200000',
flight_date = '2022-04-14',
passenger_lounge = '3A'
where flight_id = 1043;
```

```
update flight

set landing_time = '145000',
arrived_time = '092000',
flight_date = '2022-06-28',
passenger_lounge = '9C'
where flight_id = 6507;
```

Show & Select:

```
show tables;
show triggers;
select*from airline;
select*from pilot;
select*from plane;
select*from flight;
select*from passenger;
```

```
select*from payment;
```

```
select*from boarding;
```

```
select*from seat;
```

```
select*from changed_flights;
```

```
select*from boarding_safe;
```

## Appendix - Code <DDL Commands>

Airline and Plane table:

```

CREATE TABLE airline(
    airline_id INT PRIMARY KEY ,
    airline_name VARCHAR(100) NOT NULL,
    number_of_flights INT,
    contact VARCHAR(255) UNIQUE
);

CREATE TABLE plane(
    plane_id INT PRIMARY KEY ,
    number_of_seats INT NOT NULL,
    pilot_id INT,
    airline_id INT,
    FOREIGN KEY (pilot_id) REFERENCES pilot(pilot_id),
    FOREIGN KEY (airline_id) REFERENCES airline(airline_id)
);

```

Flight table:

The screenshot shows the SQL Server Management Studio interface. The left pane displays the database schema with the 'flight\_management' schema expanded, showing tables like 'airline', 'boarding\_safe', 'flight', etc. The central pane contains the SQL code for creating the 'boarding\_safe' table:

```

19 );
20
21 • CREATE TABLE flight(
22     flight_id INT PRIMARY KEY,
23     passenger_lounge VARCHAR(60) DEFAULT 'Un defined',
24     landing_time TIME DEFAULT '00:00:00',
25     arrived_time TIME DEFAULT '00:00:00',
26     from_airport VARCHAR(60) NOT NULL,
27     to_airport VARCHAR(60) NOT NULL,
28     flight_date DATE NOT NULL,
29     price DECIMAL(10,2) NOT NULL,
30     plane_id INT,
31     pilot_id INT,
32     FOREIGN KEY (plane_id) REFERENCES plane(plane_id),
33     FOREIGN KEY (pilot_id) REFERENCES pilot(pilot_id)
34 );
35
36

```

The right pane shows the execution results with two log entries:

- Action Output: 189 20-59-28 drop table boarding
- 190 20-59-36 CREATE TABLE boardingSafe(flight\_id INT PRIMARY KEY, date INT, passenger\_id INT, flight\_id INT, n 0 rows affected

## Passenger and Seat table:

The screenshot shows the SQL Server Management Studio interface. The left pane displays the database schema with the 'flight\_management' schema expanded, showing tables like 'airline', 'boarding\_safe', 'flight', etc. The central pane contains the SQL code for creating the 'passenger' and 'seat' tables:

```

75 • CREATE TABLE passenger(
76     passenger_id INT PRIMARY KEY,
77     lastName CHAR(50)NOT NULL,
78     firstName CHAR(50)NOT NULL,
79     phone_number INT UNIQUE NOT NULL,
80     city CHAR(50),
81     email CHAR(100),
82     date_of_birth DATE NOT NULL
83 );
84
85 • CREATE TABLE seat (
86     seat_number VARCHAR(30),
87     class VARCHAR(30),
88
89     PRIMARY KEY (seat_number)
90 );
91
92

```

The right pane shows the execution results with one log entry:

- Action Output: 191 20-59-36 CREATE TABLE seat(seat\_number VARCHAR(30), class VARCHAR(30), PRIMARY KEY (seat\_number)) 0 rows affected

## Pilot and Payment table:

The screenshot shows the MySQL Workbench interface with multiple tabs open. The left sidebar displays the 'SCHEMAS' tree, which includes 'flight\_management' and its sub-tables: 'airline', 'boarding\_safe', 'flight', 'passenger', 'payment', 'pilot', 'plane', and 'seat'. The main query editor window contains the SQL code for creating the 'boarding\_safe' table:

```

57 • CREATE TABLE pilot(
58     Pilot_id INT PRIMARY KEY,
59     first_name CHAR(50) NOT NULL,
60     last_name CHAR(50) NOT NULL,
61     pilot_license CHAR(255) UNIQUE NOT NULL
62 );
63
64
65 • CREATE TABLE payment(
66     payment_id INT PRIMARY KEY,
67     payment_method CHAR(100) NOT NULL,
68     payment_date DATE NOT NULL,
69     passenger_id INT,
70     FOREIGN KEY (passenger_id) REFERENCES passenger(passenger_id)
71 );
72
73
74

```

The 'Table: boarding\_safe' section below the code lists the columns: 'boardingpass\_id int' and 'message varchar(90)'. The 'Output' pane shows the execution results.

Boarding table:

The screenshot shows the MySQL Workbench interface with multiple tabs open. The left sidebar displays the 'SCHEMAS' tree, which includes 'flight\_management' and its sub-tables: 'airline', 'boarding\_safe', 'flight', 'passenger', 'payment', 'pilot', 'plane', and 'seat'. The main query editor window contains the SQL code for creating the 'boarding' table:

```

36 • CREATE TABLE boarding(
37     boardingpass_id INT PRIMARY KEY,
38     gate INT,
39     passenger_id INT,
40     flight_id INT,
41     payment_id INT,
42     checked_in VARCHAR(20),
43     has_boarded VARCHAR(20),
44     seat_number VARCHAR(30),
45
46
47
48     FOREIGN KEY (seat_number) REFERENCES seat(seat_number),
49     FOREIGN KEY (passenger_id) REFERENCES passenger(passenger_id),
50     FOREIGN KEY (flight_id) REFERENCES flight(flight_id),
51     FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
52 );
53

```

The 'Table: boarding\_safe' section below the code lists the columns: 'boardingpass\_id int' and 'message varchar(90)'. The 'Output' pane shows the execution results, indicating 1 row affected.

### Trigger table(1):

The screenshot shows the MySQL Workbench interface with multiple tabs open. The main query editor contains the following SQL code:

```

1 * use flight_management;
2 * CREATE TABLE changed_flights(
3     flight_id INT,
4     landing_time TIME DEFAULT '00:00:00',
5     arrived_time TIME DEFAULT '00:00:00',
6     flight_date DATE NOT NULL,
7     passenger_lounge VARCHAR(60) DEFAULT 'Un defined'
8 );
9 * CREATE TRIGGER before_update_flights
10 BEFORE UPDATE
11 ON flight
12 FOR EACH ROW
13 INSERT INTO updated_flights(flight_id,landing_time,arrived_time,flight_date,passenger_lounge)
14 VALUES(OLD.flight_id,OLD.landing_time,OLD.arrived_time,OLD.flight_date,
15 OLD.passenger_lounge);
16

```

The 'Information' pane below the code lists the columns of the 'boarding\_safe' table:

**Table: boarding\_safe**

**Columns:**

- boardingpass\_id int
- message varchar(90)

### Trigger table(2):

The screenshot shows the MySQL Workbench interface with multiple tabs open. The main query editor contains the following SQL code:

```

3
4 * create table boarding_safe(
5     boardingpass_id INT,
6     message VARCHAR(90)
7 );
8 DELIMITER $$ 
9 * CREATE TRIGGER completed_flights
10 after insert
11 ON boarding
12 FOR EACH ROW
13 begin
14     if (NEW.checked_in='Checked' AND NEW.has_boarded='Boarded') 
15     THEN
16         INSERT INTO boarding_safe(boardingpass_id,message)
17         values(NEW.boardingpass_id,'traveled')
18     end if;
19 end $$ 
20

```

The 'Information' pane below the code lists the columns of the 'boarding\_safe' table:

**Table: boarding\_safe**

**Columns:**

- boardingpass\_id int
- message varchar(90)

## Appendix - Output <DML Commands>

**Insert data:**

## Fundamental OF DATABASES CS 311

local instance flight\_management × local instance flight\_management × local instance (classicmodels) × local instance flight\_management × local instance flight\_management × local instance (flight\_management) ×

Edit View Query Database Server Tools Scripting Help

Navigator

ITEMAS

Filter objects

ssimodels  
unries  
unries  
**ght\_management**

- Tables
  - airline
  - boarding
  - flight
  - passenger
  - payment
  - pilot
  - plane
  - seat
- Views
- Stored Procedures
- Functions

i4

Administration S +

Information

```

1 • use flight_management;
2 • insert into airline(airline_id, airline_name, number_of_flights, contact) values
3 (101, 'Saudi Arabian Airline', 5, 'Email: saudia@org.sa, Number: +963224789135'),
4 (223, 'FlyNas', 3, 'Email: FlyN_airline@org.sa, Number: +9661596672381');

5
6 • insert into pilot (pilot_id, first_name, last_name,pilot_license) values
7 (0217, 'Rajaa', 'Alharbi', 4110081),
8 (0411, 'Layan', 'Baroum', 4110083),
9 (0512, 'Ghadi', 'Nouri', 4110085),
10 (0806, 'Zainab', 'Tolah', 4110100),
11 (2212, 'Maryam', 'Almedyan', 4110155),
12 (0227, 'Razan', 'Thamer', 4110158),
13 (0419, 'Tamim', 'Alghamdi', 4110027),
14 (0922, 'Meshari', 'Alzahrani', 4110096);

15
16 • insert into plane (plane_id, number_of_seats, pilot_id, airline_id) values
17 (1005 , 264, 217, 101),
18 (2007 ,190, 512, 223),
19 (3011 ,220, 806, 223),
20 (4015 ,150, 2212, 101),
21 (5017 ,100, 411, 101),
22 (7356, 193, 0227, 223),
23 (9692, 250,0922, 101),
24 (5284, 178, 0419,101);

```

No object selected

Output

Action Output

#	Time	Action
231	23.19.44	SELECT * FROM updated_flights LIMIT 0,1000

Message  
1 row(s) returned

Count Help Skip

15
16 • insert into plane (plane\_id, number\_of\_seats, pilot\_id, airline\_id) values
17 (1005 , 264, 217, 101),
18 (2007 ,190, 512, 223),
19 (3011 ,220, 806, 223),
20 (4015 ,150, 2212, 101),
21 (5017 ,100, 411, 101),
22 (7356, 193, 0227, 223),
23 (9692, 250,0922, 101),
24 (5284, 178, 0419,101);

1 • use flight\_management;
2 • insert into flight (flight\_id, passenger\_lounge, landing\_time, arrived\_time,from\_airport, to\_airport, flight\_date,
3 price, plane\_id, pilot\_id) values
4 (1043, '1A', '102500', '193500', 'King Abdu alziz International Airport', 'Albury Airport', '2022-04-14', '3850.88',3011, 0806 ),
5 (5189, '2B', '023500', '080000', 'King Abdu alziz International Airport', 'Tokyo International Airport', '2022-05-14', '5700 ', 5017, 0411 ),
6 (0982, '7D', '050500', '141000', 'Prince Mohammad Bin Abdulaziz International Airport', 'Rome Fiumicino Airport', '2022-05-26', '4010.20 ', 1005, 0217),
7 (2117, '5C', '200500', '144000', 'King Abdu alziz International Airport', 'Domodedovo Mikhail Lomonosov International Airport', '2022-05-13', '2750 ',
8 (3354, '3E', '061000', '115500', 'King Abdu alziz International Airport', 'Newark International Airport', '2022-08-20', '4100.35 ', 4015, 2212),
9 (2268, '7B', '113000', '031500', 'King Abdu alziz International Airport', 'Vancouver International Airport', '2022-05-13', '7410.75 ', 7356, 0227 ),
10 (6597, '9A', '154500', '083000', 'King Khalid International Airport', 'Geneva Airport', '2022-06-28', '4330.65 ', 5284, 0410),
11 (4186, '2F', '061500', '014500', 'Bahrain International Airport', 'Athens International Airport', '2022-07-11', '2980 ', 9692, 0922);

12
13
14 • insert into passenger (passenger\_id, lastName, firstName, phone\_number, city, email, date\_of\_birth)
values
15
16 (1003758976, 'Aljuhani', 'Fahad', 0547166212, 'Medinah', 'Aljuhani@gmail.com', '1992-03-15' ),
17 (1174921729, 'Alharbi', 'Lama', 0551378284, 'Medinah', 'LamaHrb@gmail.com', '1997-11-06' ),
18 (1040432195, 'Akbar', 'Waseem', 0542807390, 'Jeddah', 'Waseem98@hotmail.com', '1998-05-22' ),
19 (1110952865, 'Alwaleed', 'Jehan', 0531810063, 'Jeddah', 'Alwaleedjehan@hotmail.com', '2000-12-18' ),
20 (1000277715, 'Alsalem', 'Anas', 0561632615, 'Jeddah', 'Alsalem.Anas@gmail.com', '1988-07-19' ),
21 (1012761134, 'Adnan', 'Altaf', 0558666579, 'Jeddah', 'AltafAD@gmail.com', '1994-06-09' ),
22 (1509971883, 'Aldeeb', 'Saleh', 0581737821, 'Jeddah', 'Aldeebssd@gmail.com', '1989-06-11' ),
23 (1249623525, 'Almena', 'Tarneem', 0555284517, 'Riyadh', 'trnemena@gmail.com', '1992-08-27' ),
24 (1979826959, 'Alqurashi', 'Rakan', 0568264959, 'Khobar', 'rakanqrshi@gmail.com', '1979-11-07' );

```

1 • use flight_management;
2 • insert into boarding(boardingpass_id, gate, passenger_id, flight_id, payment_id, checked_in,
3 has_boarded,seat_number) values
4 (8739, 1, 1110952865 , 1043, 34, 'Checked', 'Boarded','4B'), /*Jed - ASTR, jehan*/
5 (4186, 2, 1012761134 , 5189, 93, 'Checked', 'Did not Boarded','35C'), /*Jed - JPN, altaf*/
6 (3488, 5, 1040432195 , 3354, 51, 'Checked', 'Boarded','12E'),/*Jed - NY, Waseem*/
7 (2343, 7, 1003758976 , 982, 61, 'Checked', 'Boarded','12E'),/*Med - italy, Fahad*/
8 (5266, 3, 1000277715 , 2117, 84, 'Checked', 'Boarded','49D'),/*Jed - Rus, Anas*/
9 (6450, 7, 1174921729 , 982, 14, 'Did not checked', 'Did not Boarded','44B'), /*Med - italy, Lama*/
10 (6382, 7, 1509971883 , 2268, 44, 'Checked', 'Boarded','49D'),/*Jed-Cnd, Saleh*/
11 (8512, 9, 1249623525 , 6507, 18, 'Did not checked', 'Did not Boarded','44B'),/*Rdh - Swtz, Tarneem*/
12 (7002, 2, 1979826959 , 4186, 62, 'Checked', 'Boarded','7D');/*Kbh - Grc, Rakan*/
13

14 • insert into payment(payment_id, payment_method,payment_date, passenger_id) values
15 (51, 'Mastercard', '2022-08-02', 1040432195 ), /*jed - NY, waseem*/
16 (61, 'Apple pay', '2022-05-13', 1003758976),/*med - italy, fahad*/
17 (84, 'Mada', '2022-04-30', 1000277715),/*Jed - rus, anas*/
18 (34, 'Visa', '2022-04-04', 1110952865),/*Jed - astr, jehan*/
19 (93, 'Apple pay', '2022-05-01', 1012761134),/*jed - jpn, altaf*/
20 (14, 'Mada', '2022-05-24', 1174921729 ), /*Med - Italy, lama*/
21 (44, 'Mada', '2022-04-08', 1509971883),/*Jed - CND, Saleh*/
22 (18, 'Apple pay', '2022-04-22', 1249623525),/*Rdh - SWTZ, Tarneem*/
23 (62, 'Visa', '2022-05-11', 1979826959);/*Kbh - GRC, Rakan*/
24

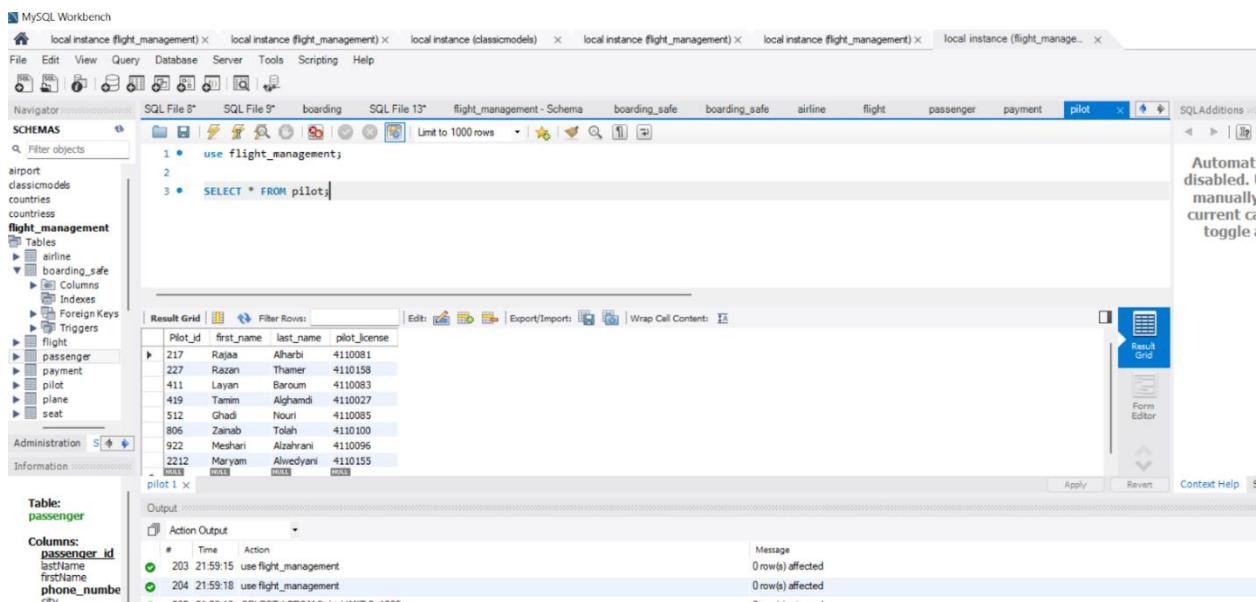
25 • insert into seat(seat_number, class) values
26 ('13A', 'Economy Class' ),
27 ('4B','First Class' ),
28 ('35C', 'Business Class'),
29 ('7D', 'Economy Class'),
30 ('28A', 'Business Class'),
31 ('44B', 'First Class'),
32 ('20C', 'Business Class'),
33 ('49D', 'Economy Class'),
34 ('12E', 'First Class');
35
36
--
```

## Retrieve all data by SELECT:

Airline table:

airline_id	airline_name	number_of_flights	contact
101	Saudi Arabian Airline	5	Email: saudAr@org.sa, Number: +9963224789...
223	FlyNas	3	Email: FlyN_Airline@org.sa, Number: +9661596...

Pilot table:



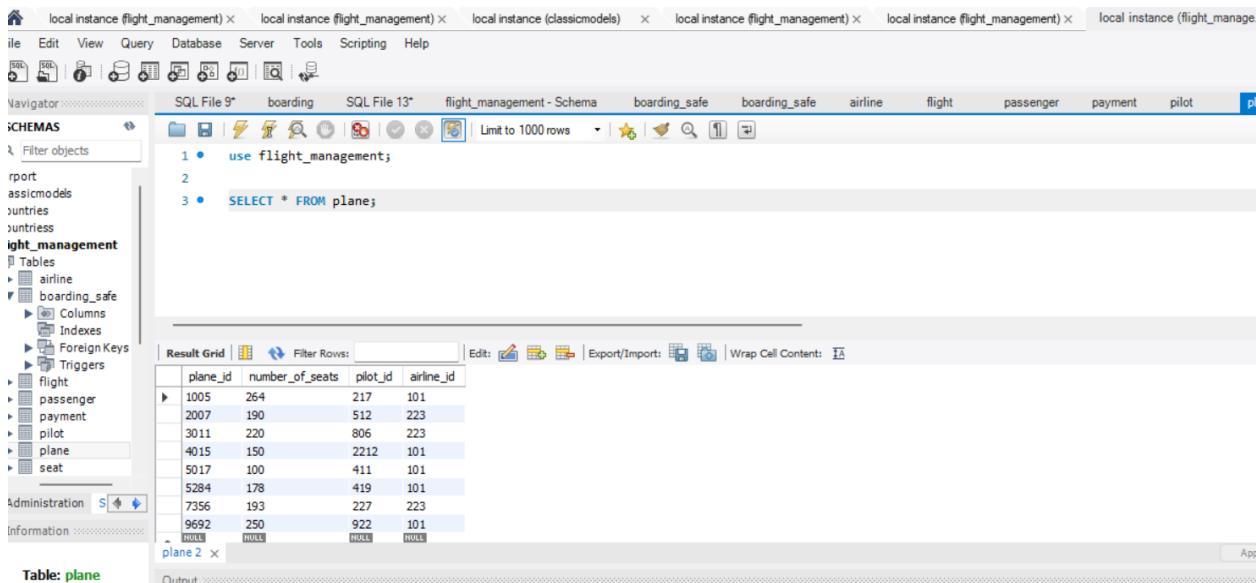
The screenshot shows the MySQL Workbench interface with the 'flight\_management' database selected. The 'pilot' table is currently being viewed. The results grid displays the following data:

pilot_id	first_name	last_name	pilot_license
217	Rajae	Alhabri	4110081
227	Razan	Thamer	4110158
411	Layan	Baroum	4110083
419	Tamim	Aghandi	4110027
512	Ghadi	Nouri	4110085
806	Zainab	Tolah	4110100
922	Meshari	Alzahrani	4110096
2212	Maryam	Alwedyani	4110155

The 'Output' pane shows the history of actions taken:

#	Time	Action	Message
203	21:59:15	use flight_management	0 row(s) affected
204	21:59:18	use flight_management	0 row(s) affected

Plane table:



The screenshot shows the MySQL Workbench interface with the 'flight\_management' database selected. The 'plane' table is currently being viewed. The results grid displays the following data:

plane_id	number_of_seats	pilot_id	airline_id
1005	264	217	101
2007	190	512	223
3011	220	806	223
4015	150	2212	101
5017	100	411	101
5284	178	419	101
7356	193	227	223
9692	250	922	101

### Flight table:

The screenshot shows the Oracle SQL Developer interface with the 'flight' schema selected. The query window contains the following SQL code:

```

1 • use flight_management;
2
3 • SELECT * FROM flight;

```

The result grid displays 19 rows of flight data:

flight_id	passenger_lounge	landing_time	arrived_time	from_airport	to_airport	flight_date	price	plane_id	pilot_id
982	7D	05:05:00	14:10:00	Prince Mohammad Bin Abdulaziz International Ai...	Rome Fiumicino Airport	2022-05-26	4010.20	1005	217
1043	1A	10:25:00	19:35:00	King Abdulla International Airport	Albury Airport	2022-04-14	3850.88	3011	806
2117	5C	20:05:00	14:40:00	King Abdulla International Airport	Domodedovo Mikhail Lomonosov International A...	2022-05-13	2750.00	2007	512
2268	7B	11:30:00	03:15:00	King Abdulla International Airport	Vancouver International Airport	2022-05-13	7410.75	7356	227
3354	3E	06:10:00	11:55:00	King Abdulla International Airport	Newark International Airport	2022-08-20	4100.35	4015	2212
4186	2F	06:15:00	01:45:00	Bahrain International Airport	Athens International Airport	2022-07-11	2980.00	9692	922
5189	2B	02:35:00	08:00:00	King Abdulla International Airport	Tokyo International Airport	2022-05-14	5700.00	5017	411
...	...	...	...	...	...	...	...	...	...

The output pane shows the executed query and its results:

```

# Time Action
199 21:56:18 SELECT * FROM flight_management.payment LIMIT 0, 1000
Message
9 row(s) returned
Duration / Fetch
0.016 sec / 0.000

```

### Passenger table:

The screenshot shows the Oracle SQL Developer interface with the 'passenger' schema selected. The query window contains the following SQL code:

```

1
2
3 • select * FROM passenger
4 ORDER BY date_of_birth;

```

The result grid displays 10 rows of passenger data:

passenger_id	lastName	firstName	phone_number	city	email	date_of_birth
1979826959	Alqurashi	Rakan	568264959	Khobar	rakanqrs@gmail.com	1979-11-07
1000277715	Alsalem	Anas	561632615	Jeddah	Alsalem.Anas@gmail.com	1988-07-19
1509971883	Alideeb	Saleh	581737821	Jeddah	Aldeebss@gmail.com	1989-06-11
1003758976	Aljuhani	Fahad	547166212	Medinah	AljuhaniF@gmail.com	1992-03-15
1249623525	Almena	Tarneem	555284517	Riyadh	trnemena@gmail.com	1992-08-27
1012761134	Adnan	Altaf	558666579	Jeddah	AltafAD@gmail.com	1994-06-09
...	...	...	...	...	...	...

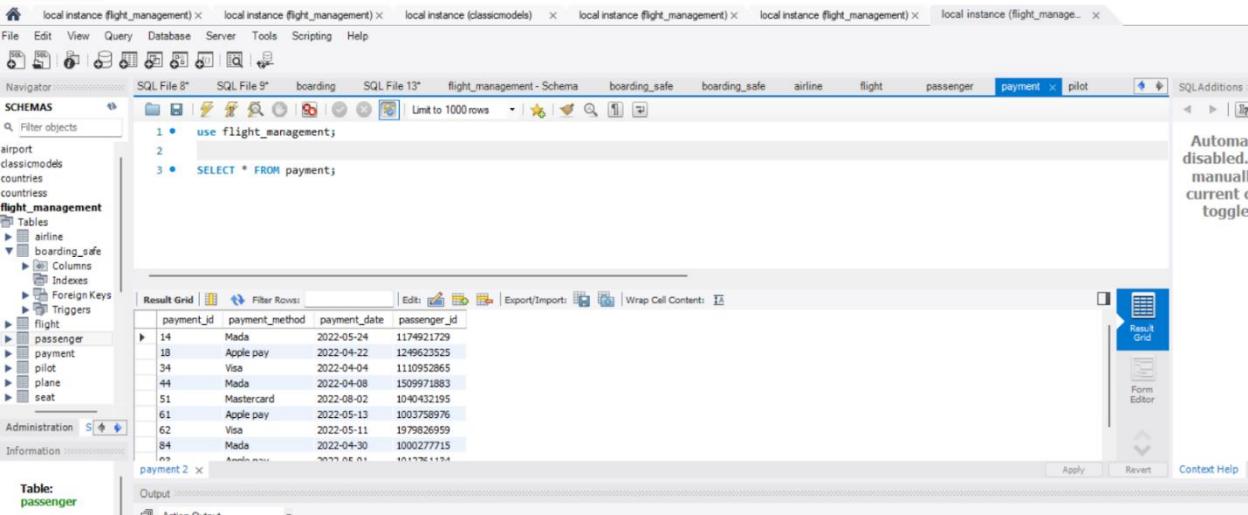
The output pane shows the executed query and its results:

```

Output:

```

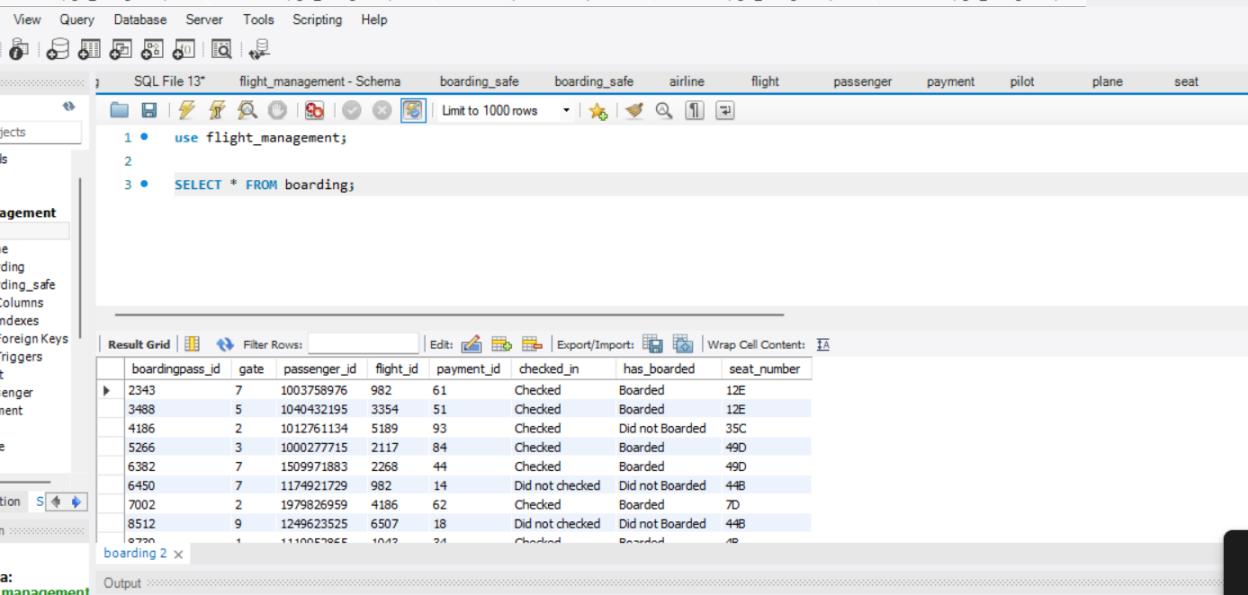
Payment table:



The screenshot shows the SQL Server Management Studio interface with multiple tabs open. The current tab is 'payment'. The left sidebar shows the schema structure under 'flight\_management' with tables like 'airline', 'boarding', 'boarding\_safe', etc. The 'Result Grid' pane displays the following data from the 'payment' table:

payment_id	payment_method	payment_date	passenger_id
14	Mada	2022-05-24	1174921729
18	Apple pay	2022-04-22	1249623525
34	Visa	2022-04-04	1109952865
44	Mada	2022-04-08	1509971883
51	Mastercard	2022-08-02	1040432195
61	Apple pay	2022-05-13	1003758976
62	Visa	2022-05-11	1979826959
84	Mada	2022-04-30	1000277715
93	Amex	2022-06-01	1012761194

Boarding table:



The screenshot shows the SQL Server Management Studio interface with multiple tabs open. The current tab is 'boarding'. The left sidebar shows the schema structure under 'flight\_management' with tables like 'airline', 'flight', 'passenger', etc. The 'Result Grid' pane displays the following data from the 'boarding' table:

boardingpass_id	gate	passenger_id	flight_id	payment_id	checked_in	has_boarded	seat_number
2343	7	1003758976	982	61	Checked	Boarded	12E
3488	5	1040432195	3354	51	Checked	Boarded	12E
4186	2	1012761134	5189	93	Checked	Did not Boarded	35C
5266	3	1000277715	2117	84	Checked	Boarded	49D
6382	7	1509971883	2268	44	Checked	Boarded	49D
6450	7	1174921729	982	14	Did not checked	Did not Boarded	44B
7002	2	1979826959	4186	62	Checked	Boarded	70
8512	9	1249623525	6507	18	Did not checked	Did not Boarded	44B
9720	1	1109952865	1012	21	Checked	Boarded	49

Seat table:

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, under the 'HEMAS' schema, the 'seat' table is selected. In the 'Scripting' pane, a T-SQL script is displayed:

```

1 • use flight_management;
2
3 • SELECT * FROM seat;

```

The results grid shows the following data:

seat_number	class
12E	First Class
13A	Economy Class
20C	Business Class
28A	Business Class
35C	Business Class
41B	First Class
49D	Economy Class
48	First Class
51	Economy Class

Below the results grid, the 'Output' pane shows the execution message:

```

# Time Action
1 22:01:15 SELECT * FROM payment LIMIT 0, 1000
Message
9 row(s) returned

```

## Aggregate Functions:

1. Retrieve the lowest price for a flight from each airport according to the starting place of the flight using MIN() .

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, under the 'flight\_management' schema, the 'flight' table is selected. In the 'Scripting' pane, a T-SQL query is displayed:

```

1 • use flight_management;
2
3 • SELECT from_airport,
4     MIN(price) as lowest_price
5     FROM flight
6     GROUP BY from_airport;

```

The results grid shows the following data:

from_airport	lowest_price
Prince Mohammad Bin Abdulaziz International Ai...	4010.20
King Abdu alaziz International Airport	2750.00
Bahrain International Airport	2980.00
King Khalid International Airport	4330.65

Below the results grid, the 'Output' pane shows the execution message:

```

# Time Action
1 22:04:00 use flight_management
Message
0 rows affected

```

## 2. AND & OR

```

use flight_management;
SELECT * FROM boarding
WHERE gate=7 AND checked_in='checked' OR payment_id>30
  
```

The screenshot shows the SQL Server Management Studio interface. The left sidebar displays the database schema with tables like airline, boarding, flight, passenger, payment, pilot, plane, and seat. The main window shows a query window with the following SQL code:

```

use flight_management;
SELECT * FROM boarding
WHERE gate=7 AND checked_in='checked' OR payment_id>30
  
```

The results grid shows the following data:

boardingpass_id	gate	passenger_id	flight_id	payment_id	checked_in	has_boarded	seat_number
2343	7	1003758976	982	61	Checked	Boarded	12E
3488	5	1040432195	3354	51	Checked	Boarded	12E
4186	2	1012761134	5189	93	Checked	Did not Boarded	35C
5266	3	1000277715	2117	84	Checked	Boarded	49D
6382	7	1509971883	2268	44	Checked	Boarded	49D
7002	2	1979826959	4186	62	Checked	Boarded	7D

## 3. Retrieve “Select” the average number of seats for a specific airline using avg function.

The screenshot shows the SQL Server Management Studio interface. The left sidebar displays the database schema with tables like airline, boarding, flight, passenger, payment, pilot, plane, and seat. The main window shows a query window with the following SQL code:

```

Select avg(number_of_seats)
as ' AVG Number of seat in Saudi Airline '
from plane
where airline_id = 101;
  
```

The results grid shows the following data:

Avg Number of seat in Saudi Airline
188.4000

4. Retrieve “Select” the total price for flights according to the starting place of the flight using sum function.

```

1
2 • Select sum(price)
3 As "King Abdu alziz Intarnational Airport flights price"
4 From flight
5 Where from_airport = 'King Abdu alziz Intarnational Airport';

```

The screenshot shows a SQL Server Management Studio interface. In the center, there is a query window with the following SQL code:

```

1
2 • Select sum(price)
3 As "King Abdu alziz Intarnational Airport flights price"
4 From flight
5 Where from_airport = 'King Abdu alziz Intarnational Airport';

```

Below the code, the results grid displays the output:

King Abdu alziz Intarnational Airport flights price
23811.98

## Join:

Sometimes we need to get some data from many tables which are not directly related, so we used inner join.

```

1
2 • SELECT
3     airline_id , plane_id, flight_id, landing_time, arrived_time, passenger_id, payment_id
4 FROM airline
5
6     Inner join plane using(airline_id)
7     inner join flight using(plane_id)
8     inner join boarding using(flight_id)

```

The screenshot shows a SQL Server Management Studio interface. In the center, there is a query window with the following SQL code:

```

1
2 • SELECT
3     airline_id , plane_id, flight_id, landing_time, arrived_time, passenger_id, payment_id
4 FROM airline
5
6     Inner join plane using(airline_id)
7     inner join flight using(plane_id)
8     inner join boarding using(flight_id)

```

Below the code, the results grid displays the output:

airline_id	plane_id	flight_id	landing_time	arrived_time	passenger_id	payment_id
101	1005	982	05:05:00	14:10:00	1003758976	61
101	1005	982	05:05:00	14:10:00	1174921729	14
101	4015	3354	06:10:00	11:55:00	1040432195	51
101	5017	5189	02:35:00	08:00:00	1012761134	93
101	5284	6507	15:45:00	08:30:00	1249623525	18
101	9692	4186	06:15:00	01:45:00	1979826959	62
223	2007	2117	20:05:00	14:40:00	1000277715	84
223	3011	1043	10:25:00	19:35:00	1110952865	34

At the bottom, the message pane shows:

- Action Output
 

#	Time	Action
216	22:04:45	SELECT * FROM flight_management.boarding LIMIT 0, 1000
217	22:05:05	use flight_management
- Message
  - 9 row(s) returned
  - 0 row(s) affected

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQL File 14\*' containing the following SQL code:

```

1 • use airport;
2
3 • SELECT airline_id, plane_id, flight_id
4   FROM airline
5
6   Inner join plane using(airline_id)
7   inner join flight using(plane_id)
8   ORDER BY airline_id

```

Below the code, the 'Result Grid' pane displays the following data:

airline_id	plane_id	flight_id
101	1005	982
101	4015	3354
101	5017	5189
101	5284	6507
101	9692	4186
223	2007	2117
223	3011	1043
223	7356	2268

The 'Output' pane below shows the following log entries:

- 1 22:43:53 SELECT airline\_id , plane\_id , flight\_id,landing\_time,arrived\_time,passenger\_id,payment\_id FROM airline Inner join plane using(airline\_id) inner join flight using(plane\_id)... 9 row(s) returned
- 2 22:50:06 SELECT airline\_id , plane\_id , flight\_id FROM airline Inner join plane using(airline\_id) inner join flight using(plane\_id)... 8 row(s) returned
- 3 22:51:04 SELECT airline\_id , plane\_id , flight\_id FROM airline Inner join plane using(airline\_id) inner join flight using(plane\_id)... 8 row(s) returned

Trigger table:

Trigger(1):

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQL File 12\*' containing the following SQL code:

```

1 • use flight_management;
2
3 • update flight
4   set landing_time = '113000',
5     arrived_time = '200000',
6     flight_date = '2022-04-14',
7     passenger_lounge = '3A'
8   where flight_id = 1043;
9
10 • update flight
11   set landing_time = '145000',
12     arrived_time = '092000',
13     flight_date = '2022-06-28',
14     passenger_lounge = '9C'
15   where flight_id = 6507;
16
17

```

The 'Output' pane below shows the following log entries:

- Action Output

```

13     flight_date = '2022-06-28',
14     passenger_lounge = '9C'
15     where flight_id = 6507;
16
17 •   SELECT * FROM updated_flights;

```

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

flight_id	landing_time	arrived_time	flight_date	passenger_lounge
1043	10:25:00	19:35:00	2022-04-14	1A
6507	15:45:00	08:30:00	2022-06-28	9A

updated\_flights3 x

Trigger(2):

SQL File 13\* flight\_management - Schema boarding\_safe boarding\_safe X airline flight

```

1
2 •   SELECT * FROM boarding_safe;

```

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

boardingpass_id	message
8739	traveled
3488	traveled
2343	traveled
5266	traveled
6382	traveled

boarding\_safe1 x

Output: