SQL Training

Project 2

By Nora Alsaeed

Air Cargo Analysis

Description

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

Project Objective:

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

Note: You must download the dataset from the course resource section in the LMS and create the tables to perform the above objective.

Dataset description:

Customer: Contains the information of customers

- customer_id – ID of the customer
- first_name – First name of the customer
- last_name – Last name of the customer
- date_of_birth – Date of birth of the customer
- gender – Gender of the customer

passengers_on_flights: Contains information about the travel details

- aircraft_id – ID of each aircraft in a brand
- route_id – Route ID of from and to location
- customer_id – ID of the customer
- depart – Departure place from the airport
- arrival – Arrival place in the airport
- seat_num – Unique seat number for each passenger
- class_id – ID of travel class
- travel_date – Travel date of each passenger
- flight_num – Specific flight number for each route

ticket_details: Contains information about the ticket details

- p_date – Ticket purchase date
- customer_id – ID of the customer
- aircraft_id – ID of each aircraft in a brand
- class_id – ID of travel class
- no_of_tickets – Number of tickets purchased
- a_code – Code of each airport
- price_per_ticket – Price of a ticket
- brand – Aviation service provider for each aircraft

routes: Contains information about the route details

- Route_id – Route ID of from and to location
- Flight_num – Specific fight number for each route
- Origin_airport – Departure location
- Destination_airport – Arrival location
- Aircraft_id – ID of each aircraft in a brand
- Distance_miles – Distance between departure and arrival location

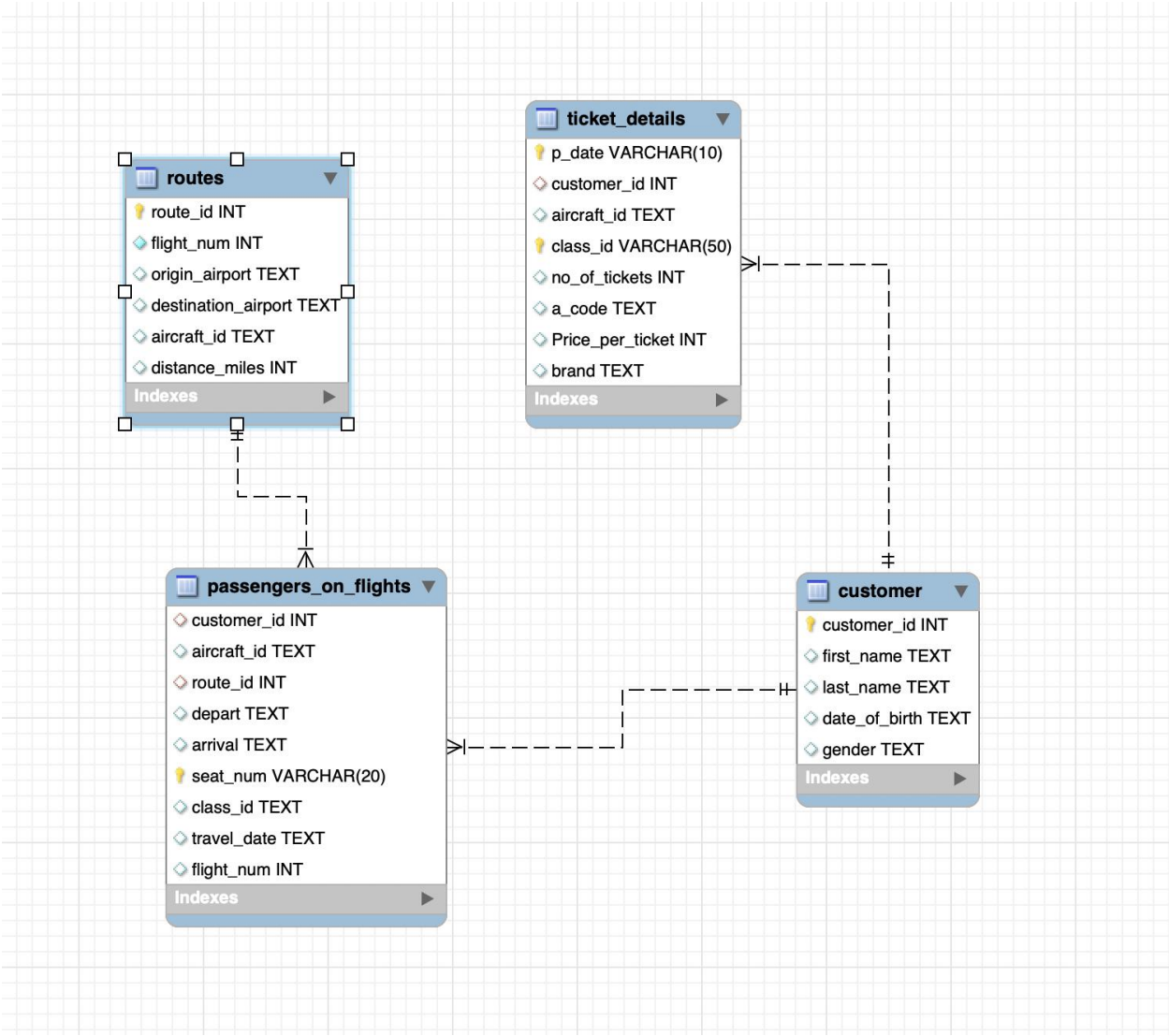
The task to be performed:

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

SQL code:

```
DESCRIBE customer;
DESCRIBE routes;
DESCRIBE passengers_on_flights;
DESCRIBE ticket_details;

ALTER TABLE customer
ADD primary key (customer_id);

ALTER TABLE passengers_on_flights
ADD primary key (seat_num);

ALTER TABLE passengers_on_flights
MODIFY  seat_num VARCHAR(20);

ALTER TABLE ticket_details
ADD primary key (p_date, class_id);

ALTER TABLE ticket_details
MODIFY  p_date VARCHAR(10),
MODIFY class_id VARCHAR(50);

ALTER TABLE passengers_on_flights
MODIFY  travel_date VARCHAR(10),
MODIFY class_id VARCHAR(50);

ALTER TABLE passengers_on_flights
ADD FOREIGN KEY(customer_id)
REFERENCES customer(customer_id);

ALTER TABLE passengers_on_flights
ADD FOREIGN KEY (route_id)
REFERENCES routes(route_id);

ALTER TABLE ticket_details
ADD FOREIGN KEY(customer_id)
REFERENCES customer(customer_id);

ALTER TABLE passengers_on_flights
ADD FOREIGN KEY(travel_date, class_id)
REFERENCES ticket_details(p_date, class_id);
```

Output:


routes
route_id INT
flight_num INT
origin_airport TEXT
destination_airport TEXT
aircraft_id TEXT
distance_miles INT
Indexes
ticket_details
p_date VARCHAR(10)
customer_id INT
aircraft_id TEXT
class_id VARCHAR(50)
no_of_tickets INT
a_code TEXT
Price_per_ticket INT
brand TEXT
Indexes
passengers_on_flights
customer_id INT
aircraft_id TEXT
route_id INT
depart TEXT
arrival TEXT
seat_num VARCHAR(20)
class_id TEXT
travel_date TEXT
flight_num INT
Indexes
customer
customer_id INT
first_name TEXT
last_name TEXT
date_of_birth TEXT
gender TEXT
Indexes

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

SQL code:

```
DESCRIBE routes;
-- route_id should be a primary key in order to be Unique and cannot contain Null values
ALTER TABLE routes
MODIFY flight_num int NOT NULL;
ALTER TABLE routes
ADD primary key (route_id);
ALTER TABLE routes
ADD CHECK (flight_num > 1),
ADD CHECk (distance_miles > 0);

SELECT *
FROM routes
WHERE distance_miles > 0;
```

Output:

100% 6:10

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
route_id	int	NO	PRI	NULL	
flight_num	int	NO		NULL	
origin_airport	text	YES		NULL	
destination_airport	text	YES		NULL	
aircraft_id	text	YES		NULL	
distance_miles	int	YES		NULL	

100% 28:26

Result Grid Filter Rows: Search Edit: Export/Import:

route_id	flight_num	origin_airp...	destination_airp...	aircraft_id	distance_mil...
1	1111	EWR	HNL	767-301ER	4962
2	1112	HNL	EWR	767-301ER	4962
3	1113	EWR	LHR	A321	3466
4	1114	JFK	LAX	767-301ER	2475
5	1115	LAX	JFK	767-301ER	2475
6	1116	HNL	LAX	767-301ER	2556
7	1117	LAX	ORD	A321	1745
8	1118	ORD	EWR	A321	719
9	1119	DEN	LAX	ERJ142	862
10	1120	HNL	DEN	A321	3365
12	1122	ABI	ADK	767-301ER	4300
13	1123	ADK	BQN	A321	2232
14	1124	BQN	CAK	A321	2445
15	1125	CAK	ANI	767-301ER	2000
16	1126	ALB	APN	A321	1700
17	1127	APN	BLV	767-301ER	1900
18	1128	ANI	BGR	ERJ142	2450
19	1129	ATW	AVL	A321	2222

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

SQL code:

```
SELECT c.customer_id, c.first_name, c.last_name,
p.route_id, p.travel_date
FROM customer as c
INNER JOIN passengers_on_flights as p
ON c.customer_id=p.customer_id
WHERE p.route_id BETWEEN 01 AND 25;
```

Output:

100% 6:32

Result Grid Filter Rows: Search Export:

customer_id	first_name	last_name	route_id	travel_date
1	Julie	Sam	9	26-12-2019
2	Steve	Ryan	4	02-09-2018
4	Cathenna	Emily	4	30-04-2020
4	Cathenna	Emily	5	06-04-2020
5	Aaron	Kim	22	31-05-2020
5	Aaron	Kim	18	06-05-2020
5	Aaron	Kim	12	02-07-2018
7	Anderson	Stewart	20	08-07-2020
9	Leo	Travis	15	10-09-2020
10	Melvin	Tracy	10	11-10-2020
11	Roger	Walson	4	09-11-2020
11	Roger	Walson	5	12-11-2020
13	Solomon	Walter	13	05-01-2019
15	Linda	William	14	02-11-2018
17	Catherine	Shad	13	03-06-2019
18	Gloria	Richie	1	01-04-2018

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

SQL code:

```
SELECT SUM(no_of_tickets) AS num_passenger_in_business_class,
SUM(Price_per_ticket * No_of_tickets) AS total_revenue_of_business_class
FROM ticket_details
WHERE class_id = 'Bussiness';
```

Output:

100% 7:46

Result Grid Filter Rows: Search Export:

num_passenger_in_business_cl...	total_revenue_of_business_cl...
13	6034

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

SQL code:

```
SELECT customer_id,
concat_ws('',first_name,last_name) AS customer_full_name
FROM customer;
```

Output:

100% 25:57

Result Grid Filter Rows: Search Export:

customer_id	customer_full_name
1	JulieSam
2	SteveRyan
3	MorrisLois
4	CathennaEmily
5	AaronKim
6	AlexanderScot
7	AndersonStewart
8	FloydTed
9	LeoTravis
10	MelvinTracy
11	RogerWalson
12	ShirleyWally
13	SolomonWalter
14	CarolVernon
15	LindaWilliam
16	ChirstineWillis

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

SQL code:

```
SELECT c.*
FROM customer c
JOIN ticket_details td ON c.customer_id = td.customer_id;
```

Output:

100% 8:66

Result Grid Filter Rows: Search Export:

customer_id	first_name	last_name	date_of_bir...	gender
27	Cherly	Vernon	19-03-1992	F
22	Pheny	Eri	29-01-1999	M
21	Chirsty	Josh	10-01-2004	M
4	Cathenna	Emily	14-09-1977	F
5	Aaron	Kim	18-02-1991	M
7	Anderson	Stewart	11-01-1992	M
8	Floyd	Ted	21-02-1993	M
9	Leo	Travis	22-03-1994	M
10	Melvin	Tracy	23-04-1995	M
11	Roger	Walson	24-05-1996	M
19	Joyce	Paul	02-06-1990	F
13	Solomon	Walter	26-07-1998	M
14	Carol	Vernon	27-08-1999	F
25	Moss	Morris	18-02-2011	M
16	Chirstine	Willis	06-10-1987	F

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

SQL code:

```
SELECT c.first_name, c.last_name
FROM customer c
JOIN ticket_details td ON c.customer_id = td.customer_id
WHERE td.brand = 'Emirates';
```

Output:

100% 11:76

Result Grid Filter Rows: Search Export:

first_name	last_name
Steve	Ryan
Cathenna	Emily
Cathenna	Emily
Aaron	Kim
Anderson	Stewart
Leo	Travis
Roger	Walson
Roger	Walson
Carol	Vernon
Gloria	Richie
Gloria	Richie
Joyce	Paul
Moss	Morris
Moss	Morris

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

SQL code:

```
SELECT customer_id
FROM passengers_on_flights
WHERE class_id = 'Economy Plus'
GROUP BY customer_id
HAVING COUNT(*) > 0;
```

Output:

100% 12:87

Result Grid Filter Rows: Search Export:

customer_id
1
8
11
17
19
22
32
47
50

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

SQL code:

```
SELECT IF(SUM(price_per_ticket * no_of_tickets) > 10000, 'Revenue has crossed 10000',
'Revenue has not crossed 10000') AS revenue_status
FROM ticket_details;
```

Output:

100% 21:98

Result Grid Filter Rows: Search Export:

revenue_status
Revenue has crossed 10000

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

SQL code:

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY '123';

GRANT ALL PRIVILEGES ON airlines.* TO 'new_user'@'localhost';
```

Output:

35	21:47:47	CREATE USER 'new_user'@'localhost' IDENTIFIED BY '123'	0 row(s) affected	0.023 sec
36	21:47:52	GRANT ALL PRIVILEGES ON airlines.* TO 'new_user'@'localhost'	0 row(s) affected	0.0036 sec

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

SQL code:

```
SELECT class_id, MAX(price_per_ticket) OVER (PARTITION BY class_id) AS max_ticket_price
FROM ticket_details;
```

Output:

100% 13:121

Result Grid Filter Rows: Search Export:

class_id	max_ticket_pri...
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Bussiness	510
Economy	190
Economy	190
Economy	190
Economy	190
Economy	190

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

SQL code:

```
SELECT *
FROM passengers_on_flights
WHERE route_id = 4;
```

Output:

100% 20:131

Result Grid Filter Rows: Search Export:

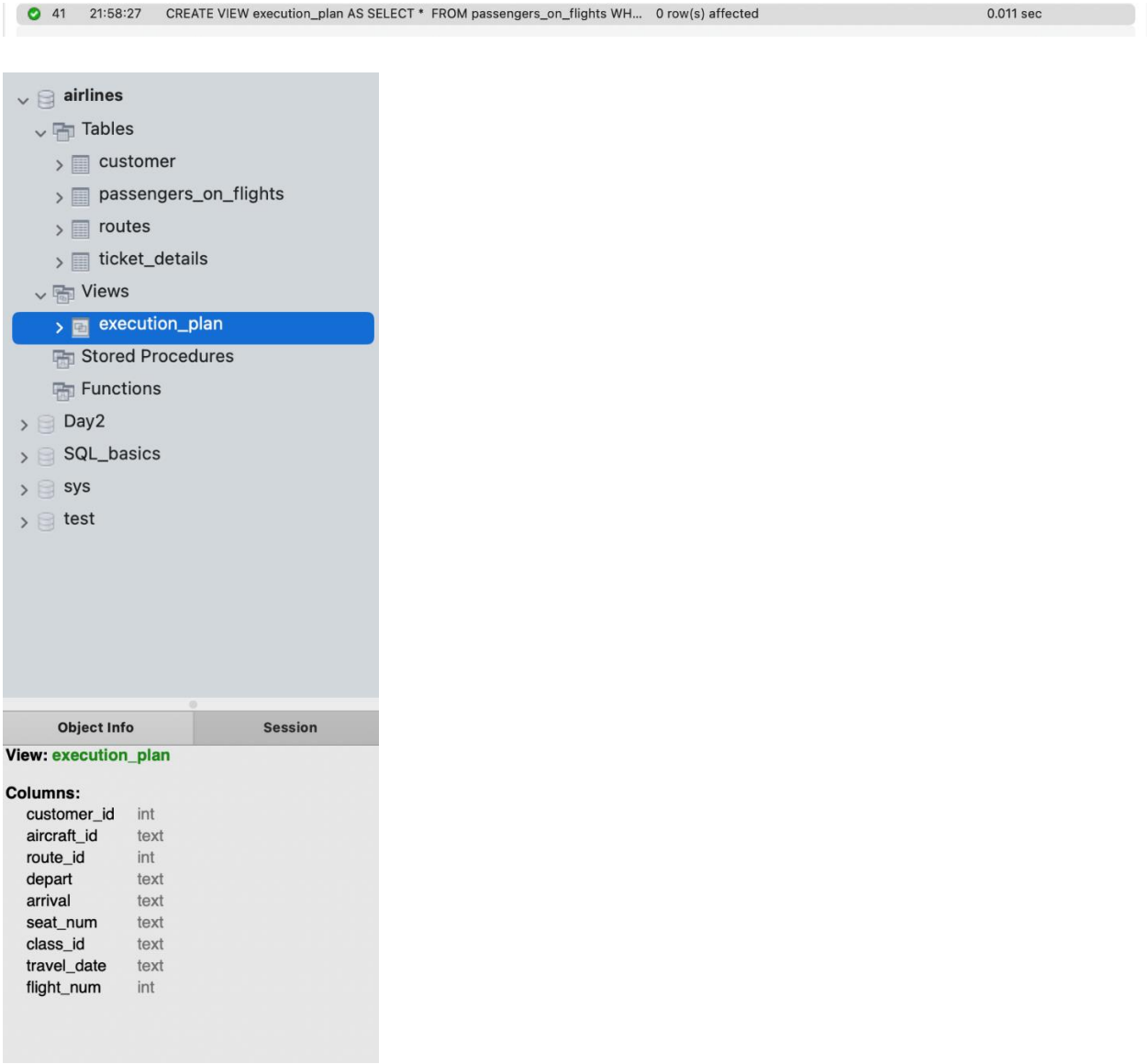
customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_nu...
2	767-301ER	4	JFK	LAX	01E	Economy	02-09-2018	1114
4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114
11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114

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

SQL code:

```
CREATE VIEW execution_plan AS
SELECT *
FROM passengers_on_flights
WHERE route_id = 4;
```

Output:



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

SQL code:

```
SELECT customer_id, aircraft_id, SUM(Price_per_ticket) AS total_price
FROM ticket_details
GROUP BY customer_id, aircraft_id WITH ROLLUP;
```

Output:

100% 1:150

Result Grid Filter Rows: Search Export:

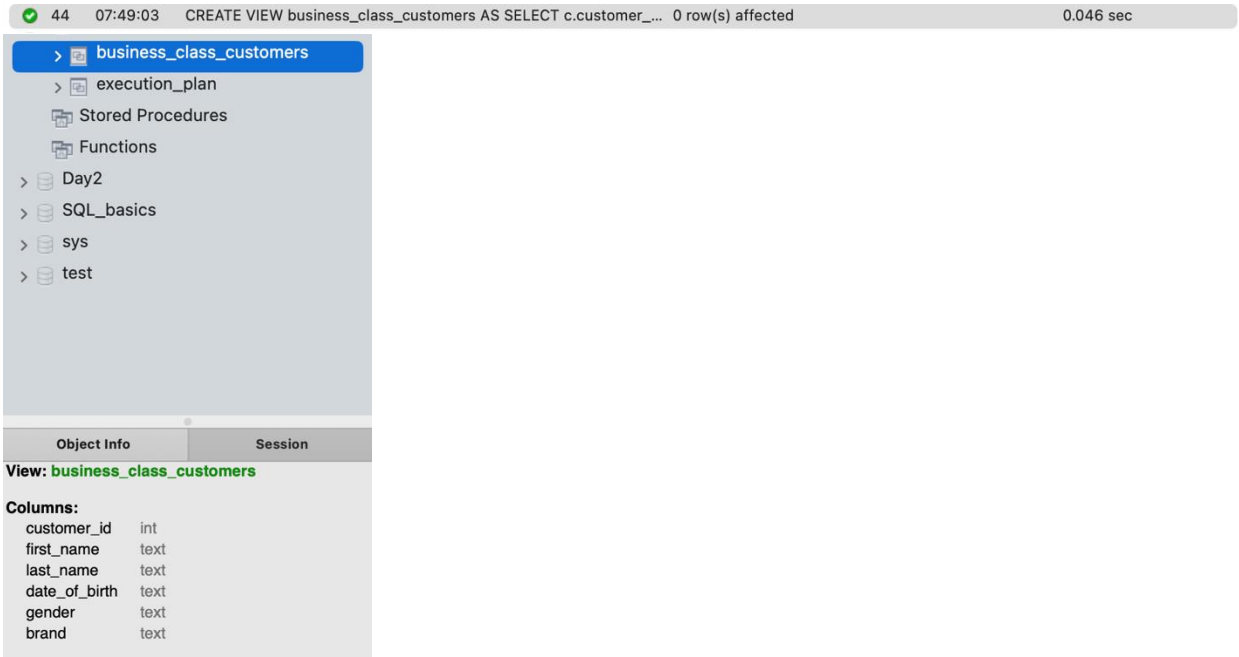
customer_id	aircraft_id	total_price
1	CRJ900	320
1	ERJ142	250
1	NULL	570
2	767-301ER	130
2	A321	505
2	NULL	635
4	767-301ER	780
4	NULL	780
5	767-301ER	430
5	ERJ142	240
5	NULL	670
7	767-301ER	430
7	NULL	430
8	A321	465
8	NULL	465
9	767-301ER	380
9	CRJ900	390
9	NULL	770

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

SQL code:

```
CREATE VIEW business_class_customers AS
SELECT c.customer_id, c.first_name, c.last_name, c.date_of_birth, c.gender, td.brand
FROM Customer c
JOIN ticket_details td ON c.customer_id = td.customer_id
WHERE td.class_id = 'Business';
```

Output:



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

SQL code:

```
DELIMITER $$
CREATE PROCEDURE get_flight_route_range (IN flight_route_id1 INT, IN flight_route_id2 INT)
BEGIN

DECLARE passengers_table_exists INT;
DECLARE customer_table_exists INT;

SELECT COUNT(*) INTO passengers_table_exists
FROM information_schema.tables
WHERE table_schema = DATABASE() AND table_name = 'passengers_on_flights';
SELECT COUNT(*) INTO customer_table_exists
FROM information_schema.tables
WHERE table_schema = DATABASE() AND table_name = 'customer';

-- Return an error message if either of the tables does not exist
IF passengers_table_exists = 0 OR customer_table_exists = 0 THEN
SELECT 'Error: One or more of the required tables are not exist. ' AS Message;
ELSE
--  Check the number of rows that would be returned by the query
SET @num_rows = (
SELECT COUNT(*)
FROM passengers_on_flights AS p
WHERE p.route_id BETWEEN flight_route_id1 AND flight_route_id2
);
-- Return an error message if there is no matching rows
IF @num_rows = 0 THEN
SELECT 'Error: No data found for the specified flight route range. Table Doesnt Exist' AS
Message;
ELSE
-- Fetching passenger and customer details between the specified routes
SELECT p.route_id,
p.depart,
p.arrival,
p.seat_num,
c.*
FROM passengers_on_flights AS p
INNER JOIN customer AS c ON p.customer_id = c.customer_id
WHERE p.route_id BETWEEN flight_route_id1 AND flight_route_id2
```

```
ORDER BY p.route_id;
END IF;
END IF;
END $$
DELIMITER ;
CALL get_flight_route_range('1','30');
-- CALL get_flight_route_range('2','3');
```

Output:

78 09:23:55 CREATE PROCEDURE get_flight_route_range (IN flight_route_id1... 0 row(s) affected 0.0040 sec

Result Grid Filter Rows: Search Export:

route_id	depart	arrival	seat_num	customer_id	first_name	last_name	date_of_bir...	gender
1	EWR	HNL	13FC	18	Gloria	Richie	04-12-1989	F
4	JFK	LAX	01E	2	Steve	Ryan	03-04-1983	M
4	JFK	LAX	03FC	4	Cathenna	Emily	14-09-1977	F
4	JFK	LAX	05B	11	Roger	Walson	24-05-1996	M
5	LAX	JFX	02FC	4	Cathenna	Emily	14-09-1977	F
5	LAX	JFX	04B	11	Roger	Walson	24-05-1996	M
8	ORD	EWR	12FC	46	Louis	Douglas	22-09-1997	M
9	DEN	LAX	01EP	1	Julie	Sam	12-01-1989	F
9	DEN	LAX	11B	29	Watson	Ronald	11-01-1991	M
10	HNL	DEN	05E	10	Melvin	Tracy	23-04-1995	M
12	ABI	ADK	02B	5	Aaron	Kim	18-02-1991	M
13	ABI	ADK	04EP	17	Catherine	Shad	09-11-1988	F
13	ADK	BQN	06FC	13	Solomon	Walter	26-07-1998	M
14	BQN	CAK	08B	24	Calvin	Willis	15-02-1994	M
14	BQN	CAK	06B	15	Linda	William	28-09-1986	F
15	CAK	ANI	13B	49	Russell	Peter	01-06-1996	M
15	CAK	ANI	11FC	44	Bily	Brian	26-10-2002	M
15	CAK	ANI	04FC	9	Leo	Travis	22-03-1994	M

```
215 CALL get_flight_route_range('2','3');
```

00% 35:215

Result Grid Filter Rows: Search Export:

Message
Error: No data found for the specified flight route…

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

SQL code:

```
CREATE PROCEDURE get_routes_with_distance()
BEGIN
 SELECT *
 FROM routes
 WHERE distance_miles > 2000;
END &&

DELIMITER ;

CALL get_routes_with_distance();
```

Output:

54 08:08:12 CREATE PROCEDURE get_routes_with_distance() BEGIN SELEC... 0 row(s) affected 0.0027 sec

Result Grid Filter Rows: Search Export:

route_id	flight_nu...	origin_airp...	destination_airp...	aircraft_id	distance_mil...
1	1111	EWR	HNL	767-301ER	4962
2	1112	HNL	EWR	767-301ER	4962
3	1113	EWR	LHR	A321	3466
4	1114	JFK	LAX	767-301ER	2475
5	1115	LAX	JFK	767-301ER	2475
6	1116	HNL	LAX	767-301ER	2556
10	1120	HNL	DEN	A321	3365
12	1122	ABI	ADK	767-301ER	4300
13	1123	ADK	BQN	A321	2232
14	1124	BQN	CAK	A321	2445
18	1128	ANI	BGR	ERJ142	2450
19	1129	ATW	AVL	A321	2222
20	1130	AVL	BOI	767-301ER	3134
21	1131	BFL	BET	A321	2425
23	1133	BLV	BFL	767-301ER	2354
25	1135	RDM	BJI	A321	2425
34	1144	CRW	COD	A321	2452
35	1145	STT	CDB	ERJ142	2121

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

SQL code:

```
DELIMITER &&

CREATE PROCEDURE group_distance_travel()
BEGIN
 SELECT
  flight_num,
  CASE
   WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN 'SDT'
   WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN 'IDT'
   WHEN distance_miles > 6500 THEN 'LDT'
  END AS distance_category
 FROM routes;
END &&

DELIMITER ;

CALL group_distance_travel();
```

Output:

56 08:11:18 CREATE PROCEDURE group_distance_travel() BEGIN SELECT... 0 row(s) affected 0.0045 sec

Result Grid Filter Rows: Search Export:

flight_num	distance_category
1111	IDT
1112	IDT
1113	IDT
1114	IDT
1115	IDT
1116	IDT
1117	SDT
1118	SDT
1119	SDT
1120	IDT
1122	IDT
1123	IDT
1124	IDT
1125	SDT
1126	SDT
1127	SDT
1128	IDT
1129	IDT

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

SQL code:

```
DELIMITER &&

CREATE PROCEDURE get_ticket_details()
BEGIN
 SELECT
  p_date,
  customer_id,
  class_id,
  CASE
   WHEN class_id = 1 THEN 'Yes'
   ELSE 'No'
  END AS complimentary_service
 FROM
  ticket_details
 WHERE
  class_id IS NOT NULL;
END &&

DELIMITER ;

CALL get_ticket_details();
```

Output:

58 08:13:59 CREATE PROCEDURE get_ticket_details() BEGIN SELECT p_d... 0 row(s) affected 0.0036 sec

Result Grid Filter Rows: Search Export:

p_date	customer_id	class_id	complimentary_servi...
26-12-2018	27	Economy	No
02-02-2020	22	Economy Plus	No
03-03-2020	21	Bussiness	No
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Bussiness	No
08-08-2020	8	Economy Plus	No
09-09-2020	9	First Class	No
10-10-2020	10	Economy	No
11-11-2020	11	Bussiness	No
12-12-2020	19	Economy Plus	No
01-01-2019	13	First Class	No
02-02-2019	14	Economy	No
03-03-2019	25	Bussiness	No
04-04-2019	16	First Class	No
03-05-2019	17	Economy Plus	No
06-06-2019	18	Economy	No
07-07-2019	24	Bussiness	No