
Course-End Project: Air Cargo Analysis

1. Write a query to create a 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

Code:

```
CREATE TABLE route_details (  
    route_id INT UNIQUE,  
    flight_num VARCHAR(10) CHECK (flight_num IS NOT NULL),  
    origin_airport VARCHAR(50),  
    destination_airport VARCHAR(50),  
    aircraft_id INT,  
    distance_miles INT CHECK (distance_miles > 0)  
);
```

Output:



2. 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.

Code:

```
SELECT * FROM passengers_on_flights  
WHERE route_id BETWEEN 1 AND 25;
```

Output:

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
	5	ERJ142	22	BGR	BJI	03E	Economy	31-05-2020	1132
	4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114
	11	767-301ER	5	LAX	JFK	04B	Bussiness	12-11-2020	1115
	17	A321	13	ABI	ADK	04EP	Economy Plus	03-06-2019	1123
	9	767-301ER	15	CAK	ANI	04FC	First Class	10-09-2020	1125
	11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114
	10	A321	10	HNL	DEN	05E	Economy	11-10-2020	1120
	15	A321	14	BQN	CAK	06B	Bussiness	02-11-2018	1124
	13	A321	13	ADK	BQN	06FC	First Class	05-01-2019	1123
	22	ERJ142	22	BGR	BJI	07EP	Economy Plus	09-02-2020	1132
	24	A321	14	BQN	CAK	08B	Bussiness	22-07-2019	1124
	25	767-301ER	23	BLV	BFL	09B	Bussiness	07-03-2019	1133
	50	A321	21	BFL	BET	10EP	Economy Plus	15-08-2020	1131
	29	ERJ142	9	DEN	LAX	11B	Bussiness	03-05-2018	1119
	44	767-301ER	15	CAK	ANI	11FC	First Class	06-10-2020	1125
	46	A321	8	ORD	EWB	12FC	First Class	08-07-2011	1118
	49	767-301ER	15	CAK	ANI	13B	Bussiness	19-08-2020	1125
	31	767-301ER	20	AVL	BOI	13E	Economy	31-12-2018	1130
	18	767-301ER	1	EBR	HNL	13FC	First Class	01-04-2018	1111
	46	A321	25	RDM	BJI	14E	Economy	25-11-2020	1135

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

Code:

```
SELECT COUNT(customer_id) AS number_of_passengers,  
SUM(no_of_tickets * price_per_ticket) AS total_revenue
```

FROM ticket_details

WHERE class_id = 'Business';

Output:

	number_of_passengers	total_revenue
▶	13	6034

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

Code:

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM customer;
```

Output:

full_name	full_name
Julie Sam	Pheny Eri
Steve Ryan	Erwin Tosh
Morris Lois	Calvin Willis
Cathenna Emily	Moss Morris
Aaron Kim	Bryan Collin
Alexander Scot	Cherly Vernon
Anderson Stewart	Du plesis Chris
Floyd Ted	Watson Ronald
Leo Travis	Donack Dukins
Melvin Tracy	James Robert
Roger Walson	Chirstoper Sean
Shirley Wally	Mark Ethan
Solomon Walter	Jacqueline Keith
Carol Vernon	Jeffrey Aaron
Linda William	Kayla Patrick
Chirstine Willis	Samuel Scott
Catherine Shad	Alexis Scott
Gloria Richie	Tyler Edward
Joyce Paul	Adam Paul
Sara Oliver	Kyle Mark
Chirsty Josh	Roger Matthew
	Joe Daniel
	Bily Brian
	Doris Walter
	Louis Douglas
	Sophia Carl
	Wayne Noah
	Russell Peter
	Rose Arthur

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

Code:

```
SELECT DISTINCT c.customer_id, c.first_name, c.last_name
FROM customer c
JOIN ticket_details t ON c.customer_id = t.customer_id;
```

Output:

customer_id	first_name	last_name
27	Cherly	Vernon
22	Pheny	Eri
21	Chirsty	Josh
4	Cathenna	Emily
5	Aaron	Kim
7	Anderson	Stewart
8	Floyd	Ted
9	Leo	Travis
10	Melvin	Tracy
11	Roger	Walson
19	Joyce	Paul
13	Solomon	Walter
14	Carol	Vernon
25	Moss	Morris
16	Chirstine	Willis
17	Catherine	Shad
18	Gloria	Richie
24	Calvin	Willis
20	Sara	Oliver
29	Watson	Ronald
1	Julie	Sam
2	Steve	Ryan
15	Linda	William
28	Du plesis	Chris
31	James	Robert
32	Chirstoper	Sean
33	Mark	Ethan
49	Russell	Peter
50	Rose	Arthur
44	Bily	Brian
46	Louis	Douglas
47	Sophia	Carl
41	Kyle	Mark

-
6. 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.

Code:

```
SELECT c.first_name, c.last_name
FROM customer c
JOIN ticket_details t ON c.customer_id = t.customer_id
WHERE t.brand = 'Emirates';
```

Output:

	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
	Cherly	Vernon
	James	Robert
	Bily	Brian
	Russell	Peter

7. 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.

Code:

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

Output:

	customer_id	travel_count
▶	1	1
	8	1
	11	1
	17	1
	19	2
	22	1
	32	1
	47	1
	50	1

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

Code:

```
SELECT
CASE
WHEN SUM(no_of_tickets * price_per_ticket) > 10000 THEN 'Yes'
ELSE 'No'
```

```
END AS revenue_crossed_10000
```

```
FROM ticket_details;
```

Output:

	revenue_crossed_10000
►	Yes

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

Code:

```
select user,host from mysql.user;
```

```
create user pooji@localhost;
```

```
show grants for pooji@localhost;
```

```
grant all on customer.* to pooji@localhost;
```

Output:

Result Grid	Filter Rows:	Export:
Grants for pooji@localhost		
►	GRANT USAGE ON *.* TO 'pooji'@'localhost'	
	GRANT ALL PRIVILEGES ON 'customer'.* TO '...	

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

Code:

```
SELECT class_id, price_per_ticket,
```

```
MAX(price_per_ticket) OVER (PARTITION BY class_id) AS max_price_per_class
```

```
FROM ticket_details;
```

Output:

class_id	price_per_ticket	max_price_per_class	class_id	price_per_ticket	max_price_per_class
Economy	100	190	Business	499	510
Economy	190	190	Business	430	510
Economy...	220	295	Business	490	510
Economy...	225	295	Business	490	510
Economy...	220	295	Business	510	510
Economy...	250	295	Business	430	510
Economy...	250	295	Business	480	510
Economy...	275	295	Business	430	510
Economy...	295	295	Business	505	510
Economy...	275	295	Business	465	510
Economy...	225	295	Business	410	510
Economy...	225	295	Business	430	510
First Class	390	395	Business	465	510
First Class	380	395	Economy	130	190
First Class	395	395	Economy	130	190
First Class	390	395	Economy	170	190
First Class	320	395	Economy	120	190
First Class	365	395	Economy	120	190
First Class	375	395	Economy	130	190
First Class	315	395	Economy	135	190
First Class	390	315	Economy	120	190
First Class	395	395	Economy	135	190
First Class	380	395	Economy	190	190
First Class	395	395	Economy	150	190
First Class	395	395	Economy	170	190

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

Code:

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

Output:

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
▶	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

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

```
Code: EXPLAIN SELECT *  
  
FROM passengers_on_flights  
  
WHERE route_id = 4;
```

Output:

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
▶	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. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.

Code:

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

Output:

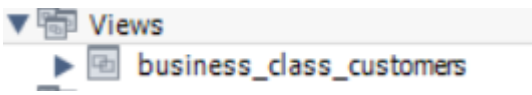
customer_id	aircraft_id	total_price	customer_id	aircraft_id	total_price	customer_id	aircraft_id	total_price
▶ 1	CRJ900	320	14	767-301ER	170	27	NULL	130
1	ERJ142	250	14	ERJ142	120	28	ERJ142	170
1	NULL	570	14	NULL	290	28	NULL	170
2	767-301ER	130	15	A321	430	29	A321	410
2	A321	505	15	NULL	430	29	ERJ142	510
2	NULL	635	16	CRJ900	395	29	NULL	920
4	767-301ER	780	16	NULL	395	31	767-301ER	130
4	NULL	780	17	A321	250	31	NULL	130
5	767-301ER	430	17	NULL	250	32	ERJ142	220
5	ERJ142	240	18	767-301ER	565	32	NULL	220
5	NULL	670	18	NULL	565	33	CRJ900	490
7	767-301ER	430	19	767-301ER	100	33	NULL	490
7	NULL	430	19	CRJ900	767-301ER	41	A321	395
8	A321	465	19	NULL	550	41	NULL	395
8	NULL	465	20	CRJ900	680	44	767-301ER	380
9	767-301ER	380	20	NULL	680	44	NULL	380
9	CRJ900	390	21	CRJ900	490	46	A321	530
9	NULL	770	21	NULL	490	46	NULL	530
10	A321	135	22	ERJ142	220	47	CRJ900	225
10	NULL	135	22	NULL	220	47	NULL	225
11	767-301ER	930	24	A321	480	49	767-301ER	430
11	ERJ142	295	24	NULL	480	49	NULL	430
11	NULL	1225	25	767-301ER	649	50	A321	275
13	A321	395	25	NULL	649	50	NULL	275
13	NULL	395	27	767-301ER	130	NULL	NULL	15369

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

Code:

```
CREATE VIEW business_class_customers AS
SELECT customer_id, brand
FROM ticket_details
WHERE class_id = 'Business';
```

Output:



customer_id	brand
21	British Airways
7	Emirates
11	Emirates
25	Emirates
24	Qatar Airways
29	Qatar Airways
2	Qatar Airways
29	Jet Airways
5	Emirates
15	Qatar Airways
33	British Airways
49	Emirates
11	Emirates

15. 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.

Code:

```
CREATE PROCEDURE GetPassengersByRouteRange(IN start_route INT, IN end_route INT)
BEGIN
    IF EXISTS (SELECT 1 FROM passengers_on_flights) THEN
        SELECT * FROM passengers_on_flights
        WHERE route_id BETWEEN start_route AND end_route;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The table does not exist';
    END IF;
END
```

Output:

customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
1	ERJ142	9	DEN	LAX	01EP	Economy Plus	26-12-2019	1119
5	767-301ER	12	ABI	ADK	02B	Bussiness	02-07-2018	1122
5	ERJ142	18	ANI	BGR	02E	Economy	06-05-2020	1128
4	767-301ER	5	LAX	JFK	02FC	First Class	06-04-2020	1115
7	767-301ER	20	AVL	BOI	03B	Bussiness	08-07-2020	1130
5	ERJ142	22	BGR	BJI	03E	Economy	31-05-2020	1132
4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114
11	767-301ER	5	LAX	JFK	04B	Bussiness	12-11-2020	1115
17	A321	13	ABI	ADK	04EP	Economy Plus	03-06-2019	1123
9	767-301ER	15	CAK	ANI	04FC	First Class	10-09-2020	1125
11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114
10	A321	10	HNL	DEN	05E	Economy	11-10-2020	1120
15	A321	14	BQN	CAK	06B	Bussiness	02-11-2018	1124
13	A321	13	ADK	BQN	06FC	First Class	05-01-2019	1123
22	ERJ142	22	BGR	BJI	07EP	Economy Plus	09-02-2020	1132
24	A321	14	BQN	CAK	08B	Bussiness	22-07-2019	1124
25	767-301ER	23	BLV	BFL	09B	Bussiness	07-03-2019	1133
50	A321	21	BFL	BET	10EP	Economy Plus	15-08-2020	1131
29	ERJ142	9	DEN	LAX	11B	Bussiness	03-05-2018	1119
44	767-301ER	15	CAK	ANI	11FC	First Class	06-10-2020	1125
46	A321	8	ORD	EWV	12FC	First Class	08-07-2011	1118
49	767-301ER	15	CAK	ANI	13B	Bussiness	19-08-2020	1125
31	767-301ER	20	AVL	BOI	13E	Economy	31-12-2018	1130
18	767-301ER	1	EWV	HNL	13FC	First Class	01-04-2018	1111
46	A321	25	RDM	BJI	14E	Economy	25-11-2020	1135

16.

16. 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.

Code:

```
CREATE PROCEDURE GetLongDistanceRoutes()
```

```
BEGIN
```

```
    SELECT *
```

```
    FROM routes
```

```
    WHERE distance_miles > 2000;
```

```
END
```

Output:

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
1	1111	EWV	HNL	767-301ER	4962
2	1112	HNL	EWV	767-301ER	4962
3	1113	EWV	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	ATV	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
43	1153	CBM	BOI	A321	8989
44	1154	COU	CAK	767-301ER	7676
46	1156	CDV	HNL	767-301ER	8668
48	1158	SCC	DEN	A321	5645
49	1159	DEC	ABI	A321	4533
50	1160	DRT	ORD	A321	2445

17. 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 .

Code:

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

Output:

flight_num	distance_category	flight_num	distance_category
1111	IDT	1135	IDT
1112	IDT	1136	SDT
1113	IDT	1137	SDT
1114	IDT	1138	SDT
1115	IDT	1139	SDT
1116	IDT	1140	SDT
1117	SDT	1141	SDT
1118	SDT	1142	SDT
1119	SDT	1143	SDT
1120	IDT	1144	IDT
1122	IDT	1145	IDT
1123	IDT	1146	SDT
1124	IDT	1147	SDT
1125	SDT	1148	SDT
1126	SDT	1149	SDT
1127	SDT	1150	SDT
1128	IDT	1151	SDT
1129	IDT	1152	SDT
1130	IDT	1153	LDT
1131	IDT	1154	LDT
1132	SDT	1155	SDT
1133	IDT	1156	LDT
1134	SDT	1157	SDT
		1158	IDT
		1159	IDT
		1160	IDT

18. 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. Condition: • If the class is Business and Economy Plus, then complimentary services are given as Yes, else it is No

Code:

```
DELIMITER //
CREATE FUNCTION get_complimentary_services(class_id Varchar(20))
RETURNS VARCHAR(3)
DETERMINISTIC
```

19. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table

Code:

```
CREATE PROCEDURE GetFirstScottCustomer()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE c_id INT;
    DECLARE c_first_name VARCHAR(50);
    DECLARE c_last_name VARCHAR(50);
    DECLARE cur CURSOR FOR
    SELECT customer_id, first_name, last_name
    FROM customer
    WHERE last_name LIKE '%Scott';
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    FETCH cur INTO c_id, c_first_name, c_last_name;

    IF NOT done THEN
        SELECT c_id AS customer_id, c_first_name AS first_name, c_last_name AS last_name;
    ELSE
        SELECT 'No customer with last name ending in Scott' AS message;
    END IF;

    CLOSE cur;
END
```

Output

	customer_id	first_name	last_name
►	37	Samuel	Scott