

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.

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

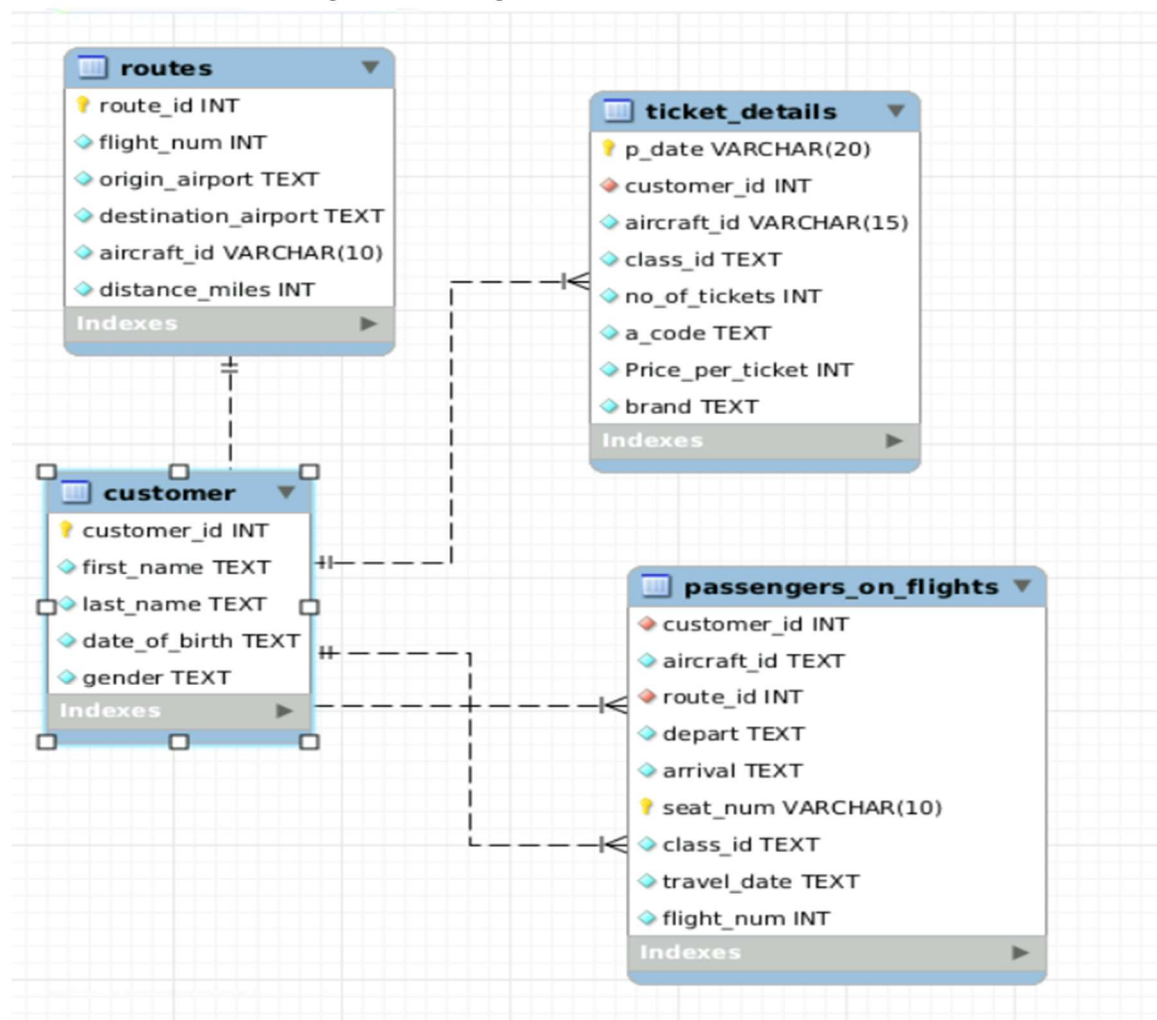
- class_id – ID of travel class
- no_of_tickets – Number of tickets purchased
- a_code – Code of each airport
- price_per_ticket – Price of a ticket
- brand – Aviation service provider for each aircraft

routes: Contains information about the route details

- Route_id – Route ID of from and to location
- Flight_num – Specific flight 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

Following operations performed:

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



- 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.

CODE:

```
CREATE table if not exists route_details (
route_id varchar(10) NOT NULL,
flight_num varchar(10) NOT NULL CHECK (flight_num>0),
origin_airport varchar(50) NOT NULL,
destination_airport varchar(50) NOT NULL,
aircraft_id varchar(10) NOT NULL,
distance_miles INT NOT NULL);
Alter table route_details add Unique(route_id);
Select*from route_details;
```

OUTPUT:

| # | route_id | flight_num | origin_airport | destination_airport | aircraft_id | distance_mile |
|---|----------|------------|----------------|---------------------|-------------|---------------|
| * | NULL | NULL | NULL | NULL | NULL | NULL |

- 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 airlines.passengers_on_flights
WHERE route_id BETWEEN 0 and 25;
```

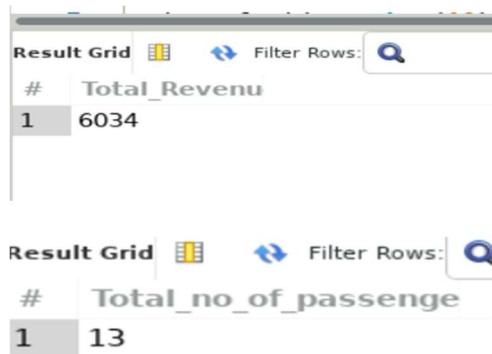
OUTPUT:

| # | customer_id | aircraft_id | route_id | depart | arrival | seat_num | class_id | travel_date | flight_num |
|---|-------------|-------------|----------|--------|---------|----------|--------------|-------------|------------|
| 1 | 2 | 767-301ER | 4 | JFK | LAX | 01E | Economy | 02-09-2018 | 1114 |
| 2 | 1 | ERJ142 | 9 | DEN | LAX | 01EP | Economy Plus | 26-12-2019 | 1119 |
| 3 | 5 | 767-301ER | 12 | ABI | ADK | 02B | Bussiness | 02-07-2018 | 1122 |
| 4 | 5 | ERJ142 | 18 | ANI | BGR | 02E | Economy | 06-05-2020 | 1128 |
| 5 | 4 | 767-301ER | 5 | LAX | JFX | 02FC | First Class | 06-04-2020 | 1115 |
| 6 | 7 | 767-301ER | 20 | AVL | BOI | 03B | Bussiness | 08-07-2020 | 1130 |
| 7 | 5 | ERJ142 | 22 | BGR | BJI | 03E | Economy | 31-05-2020 | 1132 |
| 8 | 4 | 767-301ER | 4 | JFK | LAX | 03FC | First Class | 30-04-2020 | 1114 |

- 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 "Total_no_of_passengers"  
FROM airlines.ticket_details WHERE class_id='Bussiness';  
SELECT SUM(price_per_ticket) AS "Total_Revenue"  
FROM airlines.ticket_details WHERE class_id='Bussiness';
```

OUTPUT:

| # | Total_Revenue |
|---|---------------|
| 1 | 6034 |

| # | Total_no_of_passenge |
|---|----------------------|
| 1 | 13 |

5. 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 airlines.customer;
```

OUTPUT:

| # | Full_name |
|---|------------------|
| 1 | Julie Sam |
| 2 | Steve Ryan |
| 3 | Morris Lois |
| 4 | Cathenna Emily |
| 5 | Aaron Kim |
| 6 | Alexander Scot |
| 7 | Anderson Stewart |
| 8 | Floyd Ted |

6. 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 t.customer_id, c.first_name, c.last_name  
from ticket_details t  
left join customer c on t.customer_id=c.customer_id;
```

OUTPUT:

| Result Grid | | | |
|--------------|-------------|------------|-----------|
| Filter Rows: | | | |
| # | customer_id | first_name | last_name |
| 21 | 14 | Carol | Vernon |
| 22 | 15 | Linda | William |
| 23 | 16 | Chirstine | Willis |
| 24 | 17 | Catherine | Shad |
| 25 | 18 | Gloria | Richie |
| 26 | 18 | Gloria | Richie |
| 27 | 19 | Joyce | Paul |
| 28 | 19 | Joyce | Paul |

- 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.customer_id, c.first_name, c.last_name, t.brand
FROM customer c INNER JOIN ticket_details t
ON t.customer_id=c.customer_id AND t.brand='Emirates';
```

OUTPUT:

| Result Grid | | | | |
|--------------|-------------|------------|-----------|----------|
| Filter Rows: | | | | |
| # | customer_id | first_name | last_name | brand |
| 1 | 19 | Joyce | Paul | Emirates |
| 2 | 18 | Gloria | Richie | Emirates |
| 3 | 5 | Aaron | Kim | Emirates |
| 4 | 2 | Steve | Ryan | Emirates |
| 5 | 25 | Moss | Morris | Emirates |
| 6 | 4 | Cathenna | Emily | Emirates |
| 7 | 44 | Bily | Brian | Emirates |
| 8 | 18 | Gloria | Richie | Emirates |

- 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, class_id FROM airlines.passengers_on_flights GROUP BY
customer_id, class_id HAVING class_id='Economy Plus';
```

OUTPUT:

| # | customer_id | class_id |
|---|-------------|--------------|
| 1 | 1 | Economy Plus |
| 2 | 8 | Economy Plus |
| 3 | 11 | Economy Plus |
| 4 | 17 | Economy Plus |
| 5 | 19 | Economy Plus |
| 6 | 22 | Economy Plus |
| 7 | 32 | Economy Plus |
| 8 | 47 | Economy Plus |

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

CODE:

```
SELECT @revenue := sum(ticket_details.Price_per_ticket),
IF (sum(ticket_details.Price_per_ticket)>10000, 'Revenue crossed 10000', 'Revenue not
crossed 10000')
FROM ticket_details;
```

OUTPUT:

| # | @revenue := sum(ticket_details.Price_per_ticket) | IF (sum(ticket_details.Price_per_ticket)>10000, 'Revenue crossed 10000', 'Revenue not crossed 10000') |
|---|--|---|
| 1 | 15369 | Revenue crossed 10000 |

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

CODE:

```
CREATE USER IF NOT EXISTS newuser@localhost
IDENTIFIED BY 'Asmita@123';
GRANT ALL PRIVILEGES
ON *.*
TO newuser@localhost;
SHOW GRANTS FOR newuser@localhost;
```

OUTPUT:

Result Grid  Filter Rows:  Export:  Wrap C

| # | Grants for newuser@localhost |
|---|---|
| 1 | GRANT USAGE ON *.* TO `newuser`@`localhost` |

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

CODE:

```
SELECT t.Price_per_ticket, t.class_id, MAX(t.Price_per_ticket) OVER (PARTITION
BY t.class_id) AS Max_ticket_price FROM ticket_details t;
```

OUTPUT:

| Result Grid | | | | Filter Rows: | Export |
|-------------|------------------|-----------|------------------|--------------|--------|
| # | Price_per_ticket | class_id | Max_ticket_price | | |
| 1 | 505 | Bussiness | 510 | | |
| 2 | 480 | Bussiness | 510 | | |
| 3 | 510 | Bussiness | 510 | | |
| 4 | 410 | Bussiness | 510 | | |
| 5 | 430 | Bussiness | 510 | | |
| 6 | 430 | Bussiness | 510 | | |
| 7 | 430 | Bussiness | 510 | | |

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.

CODE:

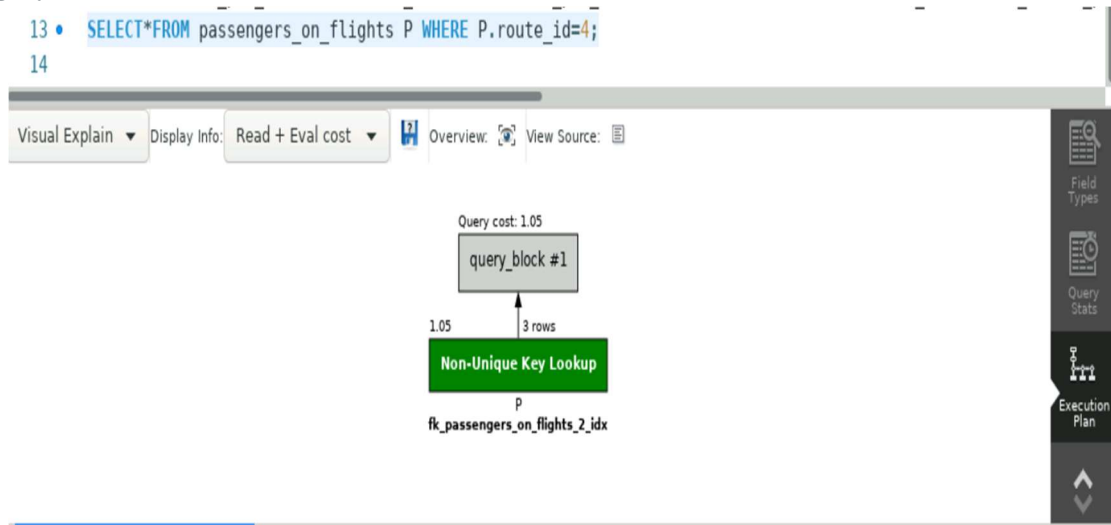
```
SELECT*FROM passengers on flights P WHERE P.route id=4;
```

OUTPUT:

[illegible]

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

OUTPUT:



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

CODE:

```
SELECT t.aircraft_id, sum(t.Price_per_ticket) AS total_price FROM ticket_details t
GROUP BY t.aircraft_id WITH ROLLUP;
```

OUTPUT:

| # | aircraft_id | total_price |
|---|-------------|-------------|
| 1 | 767-301ER | 5634 |
| 2 | A321 | 4270 |
| 3 | CRJ900 | 3440 |
| 4 | ERJ142 | 2025 |
| 5 | HULL | 15369 |

15. 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 t.customer_id, t.class_id, t.brand
FROM ticket_details t WHERE t.class_id='Bussiness';
SHOW FULL TABLES WHERE table_type='VIEW';
SELECT * FROM business_class_customers;
```


OUTPUT:

| Result Grid | | | Filter Rows: | Export: |
|-------------|--------------------------|------------|--------------|---------|
| # | Tables_in_airlines | Table_type | | |
| 1 | Emirates_customers | VIEW | | |
| 2 | business_class_customers | VIEW | | |

| Views | | | | |
|-------|-------------|-----------|------------------|--|
| # | customer_id | class_id | brand | |
| 1 | 29 | Bussiness | Jet Airways | |
| 2 | 5 | Bussiness | Emirates | |
| 3 | 15 | Bussiness | Qatar Airways | |
| 4 | 25 | Bussiness | Emirates | |
| 5 | 21 | Bussiness | Bristish Airways | |
| 6 | 24 | Bussiness | Qatar Airways | |
| 7 | 7 | Bussiness | Emirates | |
| 8 | 11 | Bussiness | Emirates | |
| 9 | 11 | Bussiness | Emirates | |

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.

CODE:

```
SELECT * from passengers_on_flights;
DROP PROCEDURE IF EXISTS passenger_details;
delimiter &&
CREATE PROCEDURE passenger_details(p_dept VARCHAR(10), p_arrival
VARCHAR(10))
BEGIN
    Select * from passengers_on_flights
    WHERE depart=p_dept AND arrival =p_arrival;
END &&
CALL passenger_details("CRW","COD");
```

OUTPUT:

| Result Grid | | | | | | | | | | |
|-------------|-------------|-------------|--------------|---------|--------------------|----------|-------------|-------------|------------|--|
| | | | Filter Rows: | Export: | Wrap Cell Content: | | | | | |
| # | customer_id | aircraft_id | route_id | depart | arrival | seat_num | class_id | travel_date | flight_num | |
| 1 | 2 | A321 | 34 | CRW | COD | 01B | Bussiness | 26-01-2019 | 1117 | |
| 2 | 41 | A321 | 34 | CRW | COD | 10FC | First Class | 15-02-2019 | 1144 | |

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.

CODE:

```
use airlines;
SELECT * from routes;
DROP PROCEDURE IF EXISTS travel_distance;
delimiter &&
CREATE PROCEDURE travel_distance()
BEGIN
SELECT route_id, flight_num, origin_airport, destination_airport, aircraft_id,
distance_miles
FROM routes WHERE distance_miles>=2000;
END&&
CALL travel_distance();
```

OUTPUT:

| Result Grid | | | | | | | |
|-------------|----------|--------------|----------------|---------------------|-------------|--------------------|--|
| | | Filter Rows: | | Export: | | Wrap Cell Content: | |
| # | route_id | flight_num | origin_airport | destination_airport | aircraft_id | distance_mile | |
| 1 | 1 | 1111 | EWB | HNL | 767-301ER | 4962 | |
| 2 | 2 | 1112 | HNL | EWB | 767-301ER | 4962 | |
| 3 | 3 | 1113 | EWB | LHR | A321 | 3466 | |
| 4 | 4 | 1114 | JFK | LAX | 767-301ER | 2475 | |
| 5 | 5 | 1115 | LAX | JFK | 767-301ER | 2475 | |
| 6 | 6 | 1116 | HNL | LAX | 767-301ER | 2556 | |
| 7 | 10 | 1120 | HNL | DEN | A321 | 3365 | |
| 8 | 12 | 1122 | ABI | ADK | 767-301ER | 4300 | |
| 9 | 13 | 1123 | ADK | BQN | A321 | 2232 | |
| 10 | 14 | 1124 | BON | CAK | A321 | 2445 | |

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 .

CODE:

```
SELECT * from routes;
DROP PROCEDURE IF EXISTS distance_categories;
delimiter &&
CREATE PROCEDURE distance_categories (IN routeid VARCHAR(10), OUT category
VARCHAR(50))
BEGIN
DECLARE dist_1 INT DEFAULT 0;
SELECT distance_miles INTO dist_1 FROM routes WHERE route_id = routeid;
IF dist_1  $\geq 0$  AND dist_1  $\leq 2000$  THEN
SET category = "SHORT DISTANCE TRAVEL";
ELSEIF dist_1  $> 2000$  AND dist_1  $\leq 6500$  THEN
SET category = "Intermediate distance travel";
ELSEIF dist_1  $> 6500$  THEN
SET category = "Long Distance Travel";
```

```

ELSE
    SET category = "Invalid Distance";
END IF ;
END &&
CALL distance_categories("1", @category);
SELECT @category;

```

OUTPUT:



| # | @category |
|---|------------------------------|
| 1 | Intermediate distance travel |

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.

Condition:

- If the class is *Business* and *Economy Plus*, then complimentary services are given as *Yes*, else it is *No*

CODE:

```

ALTER TABLE ticket_details
ADD complimentary_services VARCHAR(10);

DROP PROCEDURE IF EXISTS comp_serv;
DELIMITER &&
CREATE PROCEDURE comp_serv()
BEGIN
    Select*from ticket_details;
    UPDATE ticket_details
    SET complimentary_services = CASE
        WHEN class_id= 'Business' OR class_id = 'Economy Plus' THEN
        'Yes'
    ELSE
        'No'
    END;
END &&
CALL comp_serv();

```

OUTPUT:

| p_date | customer_id | aircraft_id | class_id | no_of_tickets | a_code | Price_per_ticket | brand | complimentary_servi... |
|------------|-------------|-------------|--------------|---------------|--------|------------------|-----------------|------------------------|
| 2018-12-26 | 27 | 767-301ER | Economy | 1 | DAL | 130 | Emirates | No |
| 2020-02-02 | 22 | ERJ142 | Economy Plus | 1 | AGB | 220 | Jet Airways | Yes |
| 2020-03-03 | 21 | CRJ900 | Bussiness | 1 | BOH | 490 | British Airways | No |
| 2020-04-04 | 4 | 767-301ER | First Class | 1 | AGB | 390 | Emirates | No |
| 2020-05-05 | 5 | ERJ142 | Economy | 1 | CTM | 120 | Jet Airways | No |
| 2020-07-07 | 7 | 767-301ER | Bussiness | 1 | BFS | 430 | Emirates | No |
| 2020-08-08 | 8 | A321 | Economy Plus | 1 | DAL | 275 | Qatar Airways | Yes |
| 2020-09-09 | 9 | 767-301ER | First Class | 1 | BOH | 380 | Emirates | No |
| 2020-10-10 | 10 | A321 | Economy | 1 | MCO | 135 | Qatar Airways | No |
| 2020-11-11 | 11 | 767-301ER | Bussiness | 1 | AGB | 465 | Emirates | No |
| 2020-12-12 | 19 | CRJ900 | Economy Plus | 1 | DEN | 225 | British Airways | Yes |

20. 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:

```
DROP PROCEDURE IF EXISTS customer_record;
DELIMITER &&
CREATE PROCEDURE customer_record()
BEGIN
DECLARE cursor_1 CURSOR
FOR SELECT * FROM customer WHERE last_name LIKE "%scott";
END&&
CALL customer_record();
```

OUTPUT:

| # | customer_id | first_name | last_name | date_of_birth | gender |
|---|-------------|------------|-----------|---------------|--------|
| 1 | 37 | Samuel | Scott | 28-01-2000 | M |
| 2 | 38 | Alexis | Scott | 31-10-2001 | M |
| * | NULL | NULL | NULL | NULL | NULL |