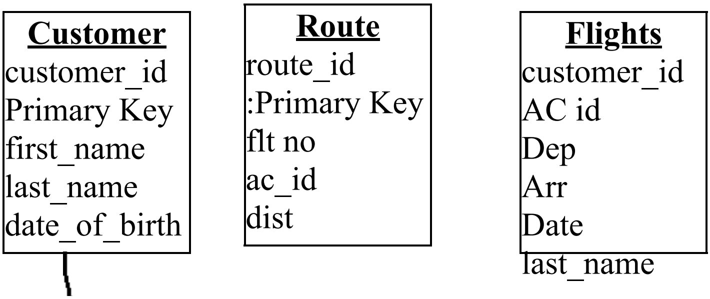
Date 04 Feb23

WRITE UP FOR SQL PROJ 2
AIRLINE PASSENGER SERVICE

The problem statement was read in detail.
The data set was examined in detail and the data was transformed with excel function by carrying out data transformation, changing date fields to yyyy-mm-dd and also trimming the data.
The structure and schema was made in sql with the SQL commands and the csv file was imported.
The import was checked.
There after point wise, every task was undertaken and sql commands were written for the same.
The last task for cursors was the most difficult. Lots of online research was done to get the code to function.
All errors were eliminated by trouble shooting and using select statements within procedures/function.
The stored procedures and functions main stay on this project.
Their practical use in application could be understood.

This was a challenging project for a non-IT person. I was fully exercised. Good learning value.

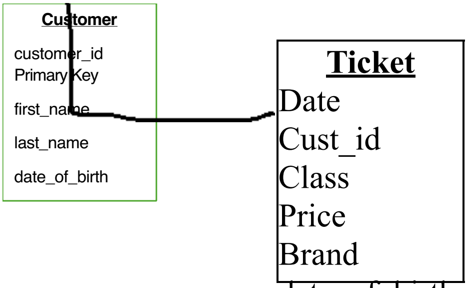
The ER diagram



Customer
customer_id
Primary Key
first_name
last_name
date_of_birth

Route
route_id
:Primary Key
flt no
ac_id
dist

Flights
customer_id
AC id
Dep
Arr
Date
last_name



Customer

customer_id
Primary Key

first_name

last_name

date_of_birth

Ticket

Date
Cust_id
Class
Price
Brand

date_of_birth

Customer

customer_id
Primary Key

first_name

last_name

date_of_birth

date_of_birth

Customer

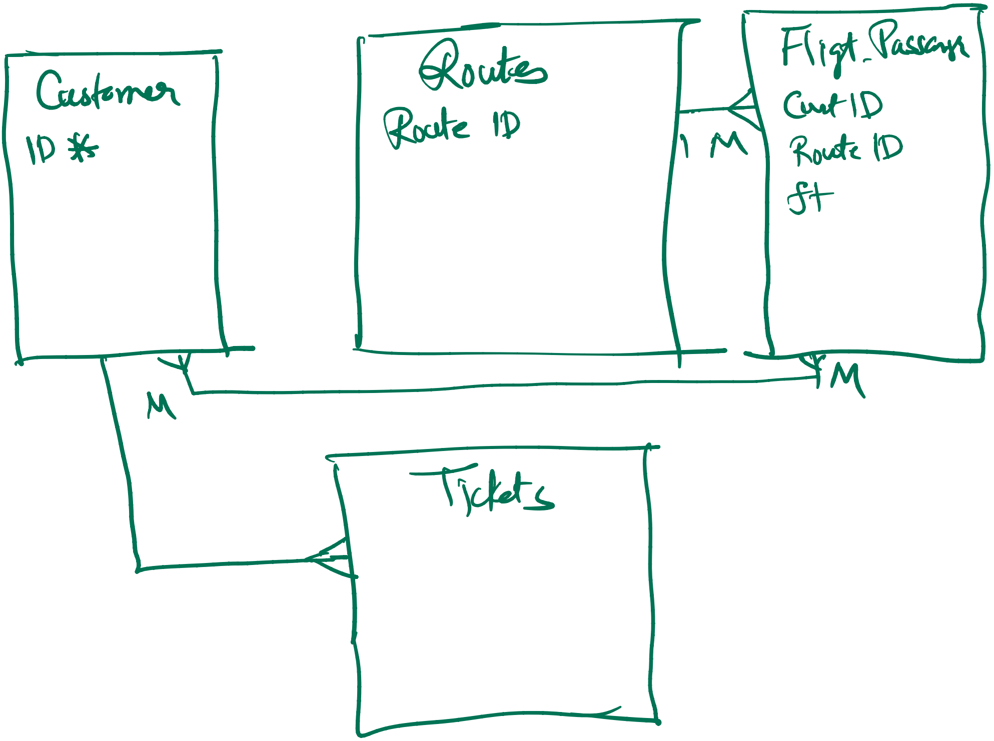
customer_id
Primary Key

first_name

last_name

date_of_birth

ER Diagram



SQL Training

Course-End Project Problem Statement

simplilearn

Get Certified. Get Ahead.

□

Course-End Project: Air Cargo Analysis

Problem Statement Scenario:

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.

Only in yyyy-mm-dd.

Dataset description:

Process & transform
Import CSV in blank exel
→ Click transform
→ Load
→ save as csv

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

Tail Nos of a/c

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

Typical format
1-2 Alphanumeric
with + numeric

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

Define a min data structure
Avoid 'txt' as it inflates data.

Difference between Varchar & char
Where to use what !

Following operations should be performed:

1. Create an ER diagram for the given airlines database. → Begining
2. 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.
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.
4. Write a query to identify the number of passengers and total revenue in business class from the ticket_details table.
5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.
6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket_details tables.
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.

?

This is just a blank table →

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.

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

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

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

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.

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

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

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

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.

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.

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.

Function in a procedure

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*

Cursors in SQL →

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.

SQL SCRIPT

```
CREATE TABLE customers (
 customer_id INT PRIMARY KEY,
 first_name VARCHAR(50),
 last_name VARCHAR(50),
 date_of_birth DATE,
 gender CHAR(1)
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/customer.csv'
INTO TABLE customers
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
-- create table for passenger flight
CREATE TABLE flights (
  customer_id INT,
  aircraft_id VARCHAR(10),
  route_id INT,
  depart VARCHAR(3),
  arrival VARCHAR(3),
  seat_num VARCHAR(4),
  class_id VARCHAR(10),
  travel_date DATE,
  flight_num INT
);
ALTER TABLE flights
MODIFY COLUMN class_id varchar(20);
```

Modify is specific to MySQL

} My class_id var inadequate min 15

Snippet

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.3/Uploads/passengers_on_flights.csv' INTO TABLE flights
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Hiddendir in PC

→ Ignores the header of the CSV.

```
-- create table for routes
CREATE TABLE routes (
  route_id INT,
  flight_num INT,
  origin_airport VARCHAR(3),
  destination_airport VARCHAR(3),
  aircraft_id VARCHAR(10),
  distance_miles INT
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/routes.csv' INTO
TABLE routes
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
-- import csv ticket details
CREATE TABLE tickets (
  p_date DATE,
  customer_id INT,
  aircraft_id VARCHAR(15),
  class_id VARCHAR(15),
  no_of_tickets INT,
  a_code VARCHAR(3),
  Price_per_ticket DECIMAL(10,2),
  brand VARCHAR(20)
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/ticket_details.csv'
INTO TABLE tickets
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Check your imports by Select *

```
/*

2.          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.
*/
CREATE TABLE route_details (
 route_id INT PRIMARY KEY, -- ensures route_id is unique
 flight_num VARCHAR(10) CHECK (flight_num REGEXP '^[A-Z0-9]{2,10}$'),  -- reg
expression
 origin_airport VARCHAR(50) NOT NULL,
 destination_airport VARCHAR(50) NOT NULL,
 aircraft_id INT REFERENCES flights(aircraft_id),
 distance_miles INT CHECK (distance_miles > 0)
);
/* flight number has a check constraint that ensures that the value matches a regular expression
pattern.
The pattern '^[A-Z0-9]{2,10}$' means that the value must start and end with an alphanumeric
character (A-Z or 0-9)
and have a length between 2 and 10 characters.  google regular expression for details*/
```

Reg expression also used in python.

explanation: Cheetsheet for syntax its complex

```
/*

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. */
SELECT * FROM flights
WHERE route_id BETWEEN 1 AND 25
ORDER by route_id, customer_id;
/*

4.          Write a query to identify the number of passengers and total revenue in
business class from the ticket_details table.
*/
SELECT class_id, COUNT(*) AS passengers, SUM(Price_per_ticket * no_of_tickets) AS
revenue
FROM tickets
WHERE class_id = 'Bussiness'
GROUP BY class_id;
/*
```

The spelling is wrong in data. Rectify during cleaning.

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

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM customers;
/*
```

Concatenate

alias

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

```
SELECT c.customer_id, c.first_name, c.last_name
FROM customers c
INNER JOIN tickets t ON c.customer_id = t.customer_id;
/*
```

Whenever you use joins use alias for tbl names

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. */

```
select * from tickets;
SELECT c.customer_id, c.first_name, c.last_name
FROM customers c
INNER JOIN tickets t ON c.customer_id = t.customer_id
WHERE  t.brand ='Emirates';
```

~~--not working~~

```
/*
```

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. */

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

These are not recognised as ids

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

```
SELECT p_date, customer_id, class_id, Price_per_ticket * no_of_tickets AS revenue,
IF(Price_per_ticket * no_of_tickets > 10000, 'YES', 'NO') AS crossed_10000
FROM tickets;
```

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

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'abc1234';
FLUSH PRIVILEGES;

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

This is syntax @ localhost

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

```
SELECT customer_id, price_per_ticket, MAX(price_per_ticket) OVER (PARTITION BY
class_id) AS max_price
FROM tickets
GROUP BY class_id, price_per_ticket,customer_id;
/*

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. */
CREATE INDEX idx_route_id ON flights (route_id);

SELECT * FROM flights
WHERE route_id = 4;


/*
14. Total price per customer across aircraft IDs with rollup: */
SELECT customer_id, SUM(price_per_ticket * no_of_tickets) AS total_price
FROM tickets
GROUP BY customer_id WITH ROLLUP;
/* WITH ROLLUP modifier to the GROUP BY clause. This will create an extra row at the
end of the result set, with a NULL value for the customer_id and the
sum of all total_price values for all customers. This is the grand total of all tickets. */
```

Rollup adds a row below with sum of coln

```
-- Task 15. View with business class customers and brands:
CREATE VIEW business_class_view AS
SELECT c.customer_id, c.first_name, c.last_name, t.brand
FROM customers c
INNER JOIN tickets t ON c.customer_id = t.customer_id
WHERE t.class_id = 'Bussiness';
-- Show view
SELECT * FROM business_class_view;
```

Call view

```
/*

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. */
```

MySQL delimiters mandatory.

```
DELIMITER &&
CREATE PROCEDURE get_passengers_by_route_range(
 IN start_route INT,
 IN end_route INT
)
BEGIN
 DECLARE error_msg VARCHAR(255);

 IF NOT EXISTS (SELECT 1 FROM information_schema.tables WHERE table_name =
'flights') THEN
   SET error_msg = 'Table flights does not exist.';
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = error_msg;
 END IF;
 SELECT *
 FROM flights
 WHERE route_id BETWEEN start_route AND end_route;
END &&
Call get_passengers_by_route_range(2,4); -- Call the procedure
```

Call by sending two int values relative to 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. */
DELIMITER &&
CREATE PROCEDURE get_long_distance_routes()
BEGIN
 SELECT *
 FROM routes
 WHERE distance_miles > 2000;
END &&
Call get_long_distance_routes();

/*
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. */
DELIMITER &&
CREATE PROCEDURE categorize_flight_distances()
BEGIN
 DECLARE distance_category VARCHAR(10);

 SELECT
   flight_num,
   CASE
     WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN 'SDT'
     WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN 'IDT'
     ELSE 'LDT'
   END AS distance_category
 FROM routes;
END &&
call categorize_flight_distances()
```

Returns two colns

```
/*
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

```
 */
DELIMITER &&
CREATE FUNCTION is_complimentary_service(class_id varchar(15))
RETURNS VARCHAR(3)
BEGIN
 DECLARE service VARCHAR(3);
 CASE class_id
   WHEN 'Bussiness' THEN SET service = 'YES';
   WHEN "First Class" THEN SET service = 'YES';
   ELSE SET service = 'NO';
 END CASE;
 RETURN service;
END &&
```

calling this function

```
DELIMITER &&
CREATE PROCEDURE get_ticket_details_with_services()
BEGIN
 SELECT p_date, customer_id, class_id, price_per_ticket * no_of_tickets AS total_price,
        is_complimentary_service(class_id) AS complimentary_services
 FROM tickets;
END &&
call get_ticket_details_with_services();
/*
```

Procedure calling a function

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.

```
*/

DELIMITER &&
CREATE PROCEDURE get_first_scott_customer()
BEGIN
 DECLARE done INT DEFAULT FALSE;
 DECLARE c_id INT;
 DECLARE c_name VARCHAR(255);
 DECLARE cur CURSOR FOR SELECT customer_id, CONCAT(first_name, ' ', last_name)
                     FROM customers
                     WHERE last_name LIKE '%Scott';

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

 OPEN cur;

 FETCH cur INTO c_id, c_name;
```

Sequence of declaring var is very important.

Open the cursor.

```
 WHILE NOT done DO
   SELECT c_id, c_name;
   FETCH cur INTO c_id, c_name;
 END WHILE;

 CLOSE cur;

 IF c_id IS NULL THEN
   SELECT 'No customer found with last name ending in Scott.';
 ELSE
   SELECT c_id, c_name;
 END IF;
END &&

CALL get_first_scott_customer();
```

We can also give input parameters to procedure

```
--Alternate method feeding parameters to the procedure
DELIMITER //

CREATE PROCEDURE get_customers_by_last_name(IN pattern VARCHAR(50))
BEGIN
 DECLARE customer_id INT;
 DECLARE first_name VARCHAR(50);
 DECLARE last_name VARCHAR(50);
 DECLARE date_of_birth DATE;
 DECLARE gender CHAR(1);
 DECLARE done INT DEFAULT FALSE;

 DECLARE customer_cursor CURSOR FOR
   SELECT customer_id, first_name, last_name, date_of_birth, gender
   FROM customers
   WHERE last_name LIKE pattern;

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

 OPEN customer_cursor;
FETCH customer_cursor INTO customer_id, first_name, last_name, date_of_birth, gender;
 customer_loop: LOOP
 select customer_id, first_name, last_name, date_of_birth, gender;
   FETCH customer_cursor INTO customer_id, first_name, last_name, date_of_birth, gender;
   IF done THEN
     LEAVE customer_loop;
   END IF;

   -- Print customer details
   SELECT CONCAT('Customer ID: ', customer_id) AS customer_id,
          CONCAT('First Name: ', first_name) AS first_name,
          CONCAT('Last Name: ', last_name) AS last_name,
          CONCAT('Date of Birth: ', DATE_FORMAT(date_of_birth, '%d-%m-%Y')) AS
date_of_birth,
          CONCAT('Gender: ', gender) AS gender;
 END LOOP customer_loop;

 CLOSE customer_cursor;
END //

DELIMITER ;

CALL get_customers_by_last_name('%Stewart');

SELECT * FROM customers WHERE last_name LIKE '%Stewart%';
```

This portion is buggy

Rectify