Air Cargo Data Analysis

SQL Course-end Project2 - PC BA DEC 2022 Cohort 1

4/26/2023

Manohari Wijesooriya

Contents

1. Introduction

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.

2. Objective

It is necessary 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.

3. Dataset Description

Customer : Contains the information of customers

Variable	Description
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

Variable	Description
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

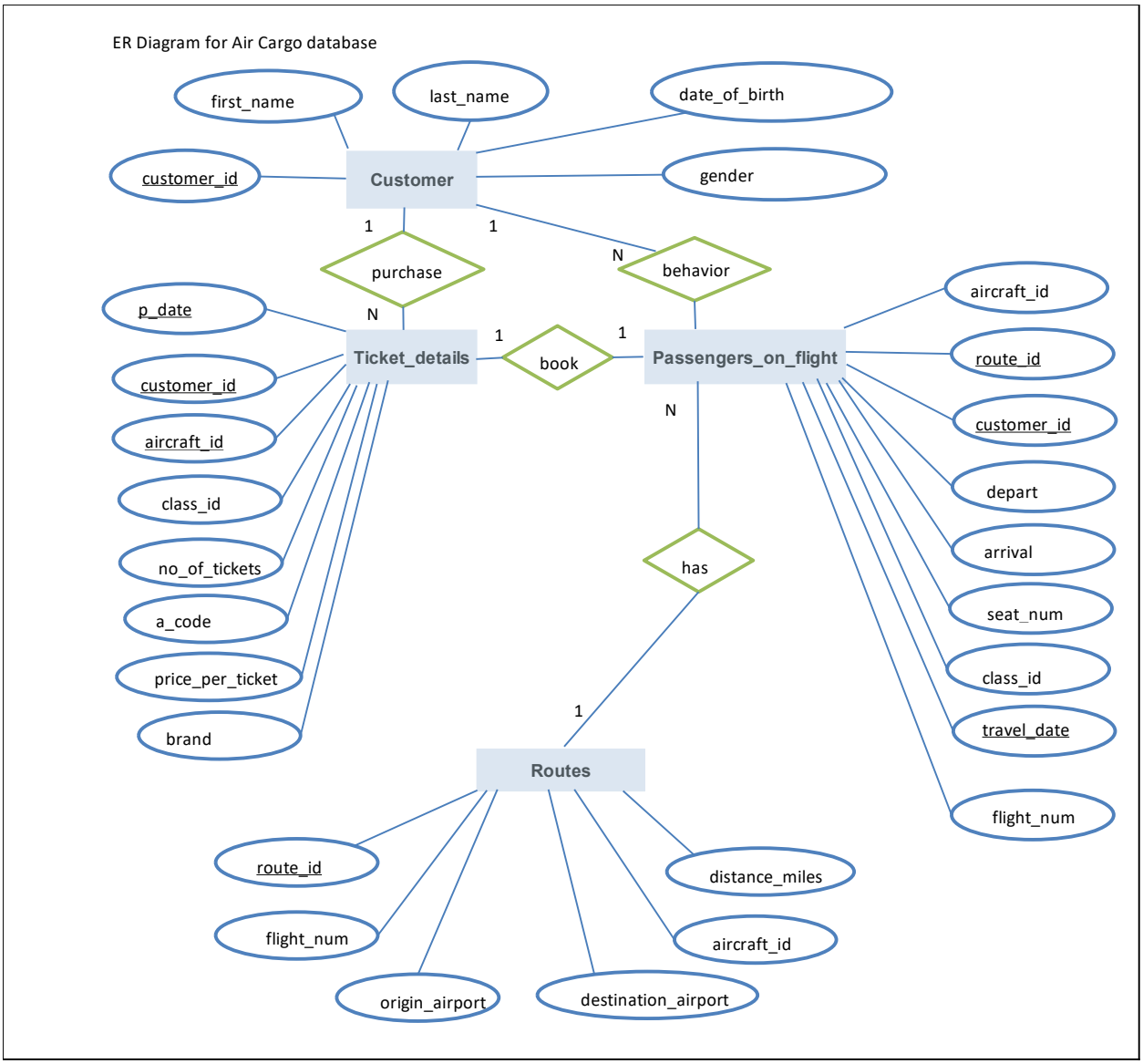
Variable	Description
p_date	Ticket purchase date
customer_id	ID of the customer
aircraft_id	ID of each aircraft in a brand
class_id	ID of travel class
no_of_tickets	Number of tickets purchased
a_code	Code of each airport
price_per_ticket	Price of a ticket
brand	Aviation service provider for each aircraft

Routes : contains information about the route details

Variable	Description
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

4. Sequence of Tasks

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



3.2. Create database and insert values.

SQL code
create databse CREATE DATABASE Air_Cargo_DB; USE Air_Cargo_DB; # create tables CREATE TABLE customer(customer_id INT primary key NOT NULL,

```
   first_name varchar(15) NOT NULL,
   last_name varchar(15) NOT NULL,
   date_of_birth date,
   gender CHAR(1),
   index(customer_id)
);

CREATE TABLE passengers_on_flight(
            customer_id INT  NOT NULL,
            aircraft_id VARCHAR(15) ,
   route_id INT  NOT NULL,
   depart CHAR(3),
            arrival CHAR(3),
   seat_num VARCHAR(4),
   class_id CHAR(12),
   travel_date date NOT NULL,
   flight_num INT,
   CONSTRAINT trdate_passenger_route PRIMARY KEY (travel_date, customer_id, route_id) # unique id to
identify passenger flights
);


CREATE TABLE routes(
            route_id INT primary key NOT NULL,
   flight_num INT,
   origin_airport CHAR(3),
   destination_airport CHAR(3),
   aircraft_id VARCHAR(15),
   distance_miles INT
);

CREATE TABLE ticket_details(
            p_date date NOT NULL,
   customer_id INT NOT NULL,
   aircraft_id VARCHAR(15) NOT NULL,
   class_id CHAR(12),
   no_of_tickets INT,
   a_code CHAR(3),
   price_per_ticket INT,
   brand VARCHAR(20),
   CONSTRAINT ticdate_cusid_aircid PRIMARY KEY (p_date, customer_id, aircraft_id)
);



# insert values to tables

INSERT INTO
Air_Cargo_DB.customer(customer_id, first_name, last_name, date_of_birth, gender)
VALUES
("1","Julie","Sam",STR_TO_DATE("12-01-1989","%d-%m-%Y"),"F" ),
("2","Steve","Ryan",STR_TO_DATE("03-04-1983","%d-%m-%Y"),"M" ),
("3","Morris","Lois",STR_TO_DATE("09-12-1993","%d-%m-%Y"),"M" ),
("4","Cathenna","Emily",STR_TO_DATE("14-09-1977","%d-%m-%Y"),"F" ),
("5","Aaron","Kim",STR_TO_DATE("18-02-1991","%d-%m-%Y"),"M" ),

INSERT INTO
Air_Cargo_DB.passengers_on_flight (customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id ,
```

```
travel_date, flight_num)
VALUES
( "2","A321","34","CRW","COD","01B","Bussiness", STR_TO_DATE("26-01-2019","%d-%m-%Y"),"1117" ) ,
( "2","767-301ER","4","JFK","LAX","01E","Economy", STR_TO_DATE("02-09-2018","%d-%m-%Y"),"1114" ) ,
( "1","ERJ142","9","DEN","LAX","01EP","Economy Plus", STR_TO_DATE("26-12-2019","%d-%m-%Y"),"1119"
) ,
( "1","CRJ900","30","BUR","STT","01FC","First Class", STR_TO_DATE("04-11-2018","%d-%m-%Y"),"1140" ) ,
( "5","767-301ER","12","ABI","ADK","02B","Bussiness", STR_TO_DATE("02-07-2018","%d-%m-%Y"),"1122" )
,

INSERT INTO
Air_Cargo_DB.routes (route_id, flight_num, origin_airport, destination_airport, aircraft_id, distance_miles)
VALUES
("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" ),
```

3.3. 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

```
CREATE TABLE route_details AS
SELECT *
FROM routes
WHERE distance_miles > 0 ;
```

Output:

route_id	flight_nun	origin_air	destinatio	aircraft_id	distance_r
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
20	1130	AVL	BOI	767-301ER	3134
21	1131	BFL	BET	A321	2425
22	1132	BGR	BJI	ERJ142	1242
23	1133	BLV	BFL	767-301ER	2354
24	1134	BJI	BQN	A321	1575
25	1135	RDM	BJI	A321	2425
26	1136	BET	BTM	ERJ142	1311
27	1137	BOI	CLD	A321	578
28	1138	BOS	CDC	767-301ER	246
29	1139	BKG	CRW	767-301ER	909
30	1140	BUR	STT	CRJ900	780
31	1141	BTM	CHA	ERJ142	660
32	1142	CLD	CHI	767-301ER	246
33	1143	CDC	CST	CRJ900	1345
34	1144	CRW	COD	A321	2452
35	1145	STT	CDB	ERJ142	2121
36	1146	CHA	COU	CRJ900	1212
37	1147	CHI	CST	767-301ER	999
38	1148	CST	DAL	A321	1111
39	1149	COD	SCC	CRJ900	1579
40	1150	CDB	DEC	A321	909
41	1151	CAE	DRT	ERJ142	898
42	1152	CSG	BOS	767-301ER	890
43	1153	CBM	BOI	A321	8989
44	1154	COU	CAK	767-301ER	7676
45	1155	CCR	EWR	CRJ900	676
46	1156	CDV	HNL	767-301ER	8668
47	1157	DAL	LAX	CRJ900	675
48	1158	SCC	DEN	A321	5645
49	1159	DEC	ABI	A321	4533
50	1160	DRT	ORD	A321	2445

3.4. 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 p.route_id as Route, CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
WHERE route_id <=25
ORDER BY route_id;
```

Output:

Route	Passenger_Name
1	Gloria Richie
4	Roger Walson
4	Steve Ryan
4	Cathenna Emily
5	Cathenna Emily
5	Roger Walson
8	Louis Douglas
9	Watson Ronald
9	Julie Sam
10	Melvin Tracy
12	Aaron Kim
13	Solomon Walter
13	Catherine Shad
14	Calvin Willis
14	Linda William
15	Russell Peter
15	Leo Travis
15	Bily Brian
18	Aaron Kim
20	James Robert
20	Anderson Stewart
21	Rose Arthur
22	Aaron Kim
22	Pheny Eri
23	Moss Morris
25	Louis Douglas

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

SQL code

```
SELECT class_id as Class, COUNT(customer_id) as Number_of_Passengers,
SUM(no_of_tickets*price_per_ticket) as Revenue
FROM Air_Cargo_DB.ticket_details
WHERE class_id = 'Bussiness'
GROUP BY class_id;
```

Output :

Class	Number_of_Passengers	Revenue
Bussiness	13	6034

3.6. 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  CONCAT(c.first_name , ' ', c.last_name) as All_Registered_Passengers
FROM Air_Cargo_DB.customer c;
```

Output:

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

3.7. 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 DISTINCT CONCAT(c.first_name , ' ', c.last_name) as Passengers_Purchased_tickets
FROM Air_Cargo_DB.customer c INNER JOIN Air_Cargo_DB.ticket_details t on
c.customer_id = t.customer_id;
```

Output:

Passengers_Purchased_tickets
Leo Travis
Joyce Paul
Gloria Richie
Watson Ronald
Floyd Ted
Sara Oliver
Aaron Kim
Roger Walson
Steve Ryan
Julie Sam
Linda William
Du plesis Chris
James Robert
Cherly Vernon
Solomon Walter
Kyle Mark
Louis Douglas
Carol Vernon
Moss Morris
Chirstine Willis
Catherine Shad
Calvin Willis
Pheny Eri
Chirstoper Sean
Chirsty Josh
Mark Ethan
Cathenna Emily
Anderson Stewart
Russell Peter
Rose Arthur
Bily Brian
Melvin Tracy
Sophia Carl

3.8. 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 CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name, t.brand
FROM Air_Cargo_DB.customer c INNER JOIN Air_Cargo_DB.ticket_details t on
c.customer_id = t.customer_id
WHERE brand = 'Emirates';
```

Output:

Passenger_Name	brand
Joyce Paul	Emirates
Gloria Richie	Emirates
Aaron Kim	Emirates
Steve Ryan	Emirates
James Robert	Emirates
Cherly Vernon	Emirates
Moss Morris	Emirates
Gloria Richie	Emirates
Moss Morris	Emirates
Carol Vernon	Emirates
Cathenna Emily	Emirates
Cathenna Emily	Emirates
Anderson Stewart	Emirates
Russell Peter	Emirates
Bily Brian	Emirates
Leo Travis	Emirates
Roger Walson	Emirates
Roger Walson	Emirates

3.9. 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 p.class_id, COUNT(p.customer_id) as No_of_Customers
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
GROUP BY p.class_id
HAVING No_of_Customers > 0 AND p.class_id = 'Economy Plus'
;
```

Output :

class_id	No_of_Customers
Economy Plus	10

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

SQL code

```
CREATE TABLE revenue as
SELECT YEAR(p_date) as Calender_year , sum(no_of_tickets * price_per_ticket) as revenue
FROM Air_Cargo_DB.ticket_details
GROUP BY Calender_year;

SELECT IF(sum(revenue) > 10000, 'Meeting target revenue', 'Not meeting target revenue')
FROM revenue;
```

Output :

Calender_year	revenue
2018	3915
2019	4969
2020	6485

IF(sum(revenue) > 10000, 'Meeting target revenue', 'Not meeting target revenue')
Meeting target revenue

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

SQL code

```
# Create a new securitydb database and a new projectdetails table.
CREATE DATABASE securitydb;

CREATE TABLE projectdetails (
            id      INT AUTO_INCREMENT,
   message VARCHAR(100) NOT NULL,
   PRIMARY KEY (id));

# create stored procedure using definer

DELIMITER $$
CREATE DEFINER = root@localhost PROCEDURE
InsertMessagess( msg VARCHAR(100))
SQL SECURITY DEFINER
BEGIN
INSERT INTO projectdetails(message)
VALUES(msg);
END$$
DELIMITER ;

# create new user junior
CREATE USER junior@localhost
IDENTIFIED BY 'junior123';

# grant access for new user
GRANT EXECUTE ON securitydb.* TO
junior@localhost;
```

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

SQL code

```
SELECT DISTINCT class_id, MAX(price_per_ticket) OVER (PARTITION BY class_id) AS Max_ticket_price
FROM Air_Cargo_DB.ticket_details;
```

Output:

class_id	Max_ticket_price
Bussiness	510
Economy	190
Economy Plus	295
First Class	395

3.13. 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

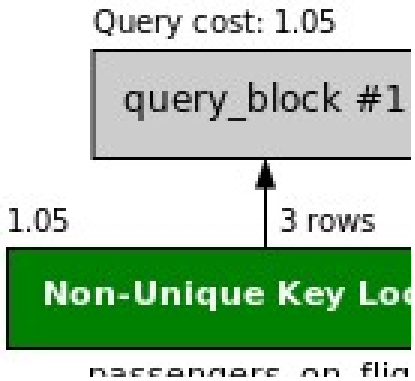
```
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'Air_Cargo_DB' AND TABLE_NAME = 'passengers_on_flight';

SELECT * FROM INFORMATION_SCHEMA.COLUMNS;

# query using database and table as it is. process time 0.025 sec
SELECT p.route_id as Route, CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
WHERE route_id =4;

# checking the table to improve performance of the query
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'Air_Cargo_DB' AND TABLE_NAME = 'passengers_on_flight';
```

```
#SELECT * FROM INFORMATION_SCHEMA.COLUMNS;

#  variable aircraft_id is defined as varchar(15)
# lets check the maximum length of the content and change the variable type.
SELECT max(length(aircraft_id)) as max_length_aircraft_id
FROM Air_Cargo_DB.passengers_on_flight;

# lets change length of variable aircraft_id to 9.
ALTER TABLE Air_Cargo_DB.passengers_on_flight MODIFY aircraft_id CHAR(9);
# run the query and note the run time.
SELECT p.route_id as Route, CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
WHERE route_id =4;
# run time is reduced to 0.00075 sec

# create an index for variable route_id to find information faster.
ALTER TABLE Air_Cargo_DB.passengers_on_flight ADD INDEX
index_col1(route_id);

SHOW INDEX From Air_Cargo_DB.passengers_on_flight;

SELECT p.route_id as Route, CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
WHERE route_id =4;

# run time is reduced to 0.00040 seconds
```

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

SQL code

```
SELECT *
FROM Air_Cargo_DB.passengers_on_flight
WHERE route_id = 4;
```



3.15. 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(no_of_tickets * price_per_ticket) as ticket_price
FROM Air_Cargo_DB.ticket_details
GROUP BY customer_id, aircraft_id WITH ROLLUP;
```

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

SQL code

```
CREATE VIEW bussiness_class_customers AS
SELECT DISTINCT CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name,
t.brand, p.class_id
FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
c.customer_id = p.customer_id
inner join Air_Cargo_DB.ticket_details t on
c.customer_id = t.customer_id
WHERE p.class_id = "Bussiness";

SELECT * FROM bussiness_class_customers;
```

Output:

Passenger_Name	brand	class_id
Watson Ronald	Jet Airways	Bussiness
Aaron Kim	Emirates	Bussiness
Roger Walson	Jet Airways	Bussiness
Steve Ryan	Emirates	Bussiness
Linda William	Qatar Airways	Bussiness
Steve Ryan	Qatar Airways	Bussiness
Moss Morris	Emirates	Bussiness
Calvin Willis	Qatar Airways	Bussiness
Watson Ronald	Qatar Airways	Bussiness
Chirsty Josh	Bristish Airways	Bussiness
Mark Ethan	Bristish Airways	Bussiness
Aaron Kim	Jet Airways	Bussiness
Anderson Stewart	Emirates	Bussiness
Russell Peter	Emirates	Bussiness
Roger Walson	Emirates	Bussiness

3.17. 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 $$
CREATE PROCEDURE routelist(rno_st, rno_end)
        BEGIN
                SELECT p.route_id as Route, CONCAT(c.first_name , ' ', c.last_name) as Passenger_Name
                FROM Air_Cargo_DB.customer c inner join Air_Cargo_DB.passengers_on_flight p on
                c.customer_id = p.customer_id
                WHERE route_id >= rno_st AND route_id <= rno_end
                ORDER BY route_id;
        END $$

CALL route_list(1,25);
```

Output:

3.18. 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

```
DROP PROCEDURE IF EXISTS get_routes;
DELIMITER $$
CREATE PROCEDURE get_routes()
BEGIN
SELECT *
FROM Air_Cargo_DB.routes
WHERE distance_miles > 2000;
END $$

CALL get_routes();
```

Output:

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
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
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

3.19. 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

```
DROP PROCEDURE IF EXISTS get_filght_category;
DELIMITER $$
CREATE PROCEDURE get_filght_category()
BEGIN
		SELECT DISTINCT flight_num, origin_airport, destination_airport, distance_miles,
		CASE WHEN distance_miles >=0    AND distance_miles <= 2000 THEN 'Short Distance Travel       (SDT)'
				WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN 'Intermediate Distance Travel (IDT)'
				WHEN distance_miles > 6500 THEN                      'Long Distance Travel        (LDT)'
		END AS flight_category
		FROM Air_Cargo_DB.routes;
END $$

CALL get_filght_category();
```

Output:

flight_num	origin_airport	destination_airport	distance_miles	flight_category
1111	EWR	HNL	4962	Intermediate Distance Travel (IDT)
1112	HNL	EWR	4962	Intermediate Distance Travel (IDT)
1113	EWR	LHR	3466	Intermediate Distance Travel (IDT)
1114	JFK	LAX	2475	Intermediate Distance Travel (IDT)
1115	LAX	JFK	2475	Intermediate Distance Travel (IDT)
1116	HNL	LAX	2556	Intermediate Distance Travel (IDT)
1117	LAX	ORD	1745	Short Distance Travel (SDT)
1118	ORD	EWR	719	Short Distance Travel (SDT)
1119	DEN	LAX	862	Short Distance Travel (SDT)
1120	HNL	DEN	3365	Intermediate Distance Travel (IDT)
1122	ABI	ADK	4300	Intermediate Distance Travel (IDT)
1123	ADK	BQN	2232	Intermediate Distance Travel (IDT)
1124	BQN	CAK	2445	Intermediate Distance Travel (IDT)
1125	CAK	ANI	2000	Short Distance Travel (SDT)
1126	ALB	APN	1700	Short Distance Travel (SDT)
1127	APN	BLV	1900	Short Distance Travel (SDT)
1128	ANI	BGR	2450	Intermediate Distance Travel (IDT)
1129	ATW	AVL	2222	Intermediate Distance Travel (IDT)
1130	AVL	BOI	3134	Intermediate Distance Travel (IDT)
1131	BFL	BET	2425	Intermediate Distance Travel (IDT)
1132	BGR	BJI	1242	Short Distance Travel (SDT)
1133	BLV	BFL	2354	Intermediate Distance Travel (IDT)
1134	BJI	BQN	1575	Short Distance Travel (SDT)
1135	RDM	BJI	2425	Intermediate Distance Travel (IDT)
1136	BET	BTM	1311	Short Distance Travel (SDT)
1137	BOI	CLD	578	Short Distance Travel (SDT)
1138	BOS	CDC	246	Short Distance Travel (SDT)
1139	BKG	CRW	909	Short Distance Travel (SDT)
1140	BUR	STT	780	Short Distance Travel (SDT)
1141	BTM	CHA	660	Short Distance Travel (SDT)
1142	CLD	CHI	246	Short Distance Travel (SDT)
1143	CDC	CST	1345	Short Distance Travel (SDT)
1144	CRW	COD	2452	Intermediate Distance Travel (IDT)
1145	STT	CDB	2121	Intermediate Distance Travel (IDT)
1146	CHA	COU	1212	Short Distance Travel (SDT)
1147	CHI	CST	999	Short Distance Travel (SDT)
1148	CST	DAL	1111	Short Distance Travel (SDT)
1149	COD	SCC	1579	Short Distance Travel (SDT)
1150	CDB	DEC	909	Short Distance Travel (SDT)
1151	CAE	DRT	898	Short Distance Travel (SDT)
1152	CSG	BOS	890	Short Distance Travel (SDT)
1153	CBM	BOI	8989	Long Distance Travel (LDT)
1154	COU	CAK	7676	Long Distance Travel (LDT)
1155	CCR	EWR	676	Short Distance Travel (SDT)
1156	CDV	HNL	8668	Long Distance Travel (LDT)
1157	DAL	LAX	675	Short Distance Travel (SDT)
1158	SCC	DEN	5645	Intermediate Distance Travel (IDT)
1159	DEC	ABI	4533	Intermediate Distance Travel (IDT)
1160	DRT	ORD	2445	Intermediate Distance Travel (IDT)

3.20. 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

SQL code

```
DROP PROCEDURE IF EXISTS get_complimentary_services;

DELIMITER $$
CREATE PROCEDURE get_complimentary_services()
BEGIN
        SELECT DISTINCT p_date, customer_id, class_id,
        CASE WHEN class_id= 'Economy Plus' OR class_id='Bussiness' THEN 'Yes' ELSE 'No '
  END AS complementary_services
        FROM Air_Cargo_DB.ticket_details;
END $$
```

```
CALL get_complimentary_services();
```

Output:

p_date	customer_id	class_id	complementary_services
2018-01-01	9	First Class	No
2018-02-01	19	Economy	No
2018-03-01	18	First Class	No
2018-04-01	29	Bussiness	Yes
2018-05-01	8	Economy	No
2018-06-01	20	First Class	No
2018-07-01	5	Bussiness	Yes
2018-08-01	11	Economy Plus	Yes
2018-09-01	2	Economy	No
2018-10-01	1	First Class	No
2018-11-01	15	Bussiness	Yes
2018-12-01	28	Economy	No
2018-12-19	31	Economy	No
2018-12-26	27	Economy	No
2019-01-01	13	First Class	No
2019-01-11	41	First Class	No
2019-01-15	46	First Class	No
2019-01-25	2	Bussiness	Yes
2019-02-02	14	Economy	No
2019-03-03	25	Bussiness	Yes
2019-04-04	16	First Class	No
2019-05-03	17	Economy Plus	Yes
2019-06-06	18	Economy	No
2019-07-07	24	Bussiness	Yes
2019-08-09	20	First Class	No
2019-09-21	25	Economy	No
2019-10-22	29	Bussiness	Yes
2019-11-23	1	Economy Plus	Yes
2019-12-24	14	Economy	No
2020-02-02	22	Economy Plus	Yes
2020-02-04	32	Economy Plus	Yes
2020-03-03	21	Bussiness	Yes
2020-03-12	33	Bussiness	Yes
2020-04-04	4	First Class	No
2020-04-29	4	First Class	No
2020-05-05	5	Economy	No
2020-05-30	5	Economy	No
2020-07-07	7	Bussiness	Yes
2020-07-17	49	Bussiness	Yes
2020-08-08	8	Economy Plus	Yes
2020-08-12	50	Economy Plus	Yes
2020-09-05	44	First Class	No
2020-09-09	9	First Class	No
2020-10-07	46	Economy	No
2020-10-10	10	Economy	No
2020-11-08	11	Bussiness	Yes
2020-11-11	11	Bussiness	Yes
2020-12-09	47	Economy Plus	Yes
2020-12-12	19	Economy Plus	Yes
2020-12-13	19	Economy Plus	Yes

3.21. 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 code

```
SELECT *
FROM Air_Cargo_DB.customer
WHERE last_name like '%Scott'
LIMIT 1;
```

Output:

customer_id	first_name	last_name	date_of_birth	gender
37	Samuel	Scott	2000-01-28	M

3.22. Write a query to output number of tickets sold and revenue by year and route.

3.23. Write a query to list top 5 most frequent customers by year.