Air Cargo Analysis

Description

Problem Statement Scenario:

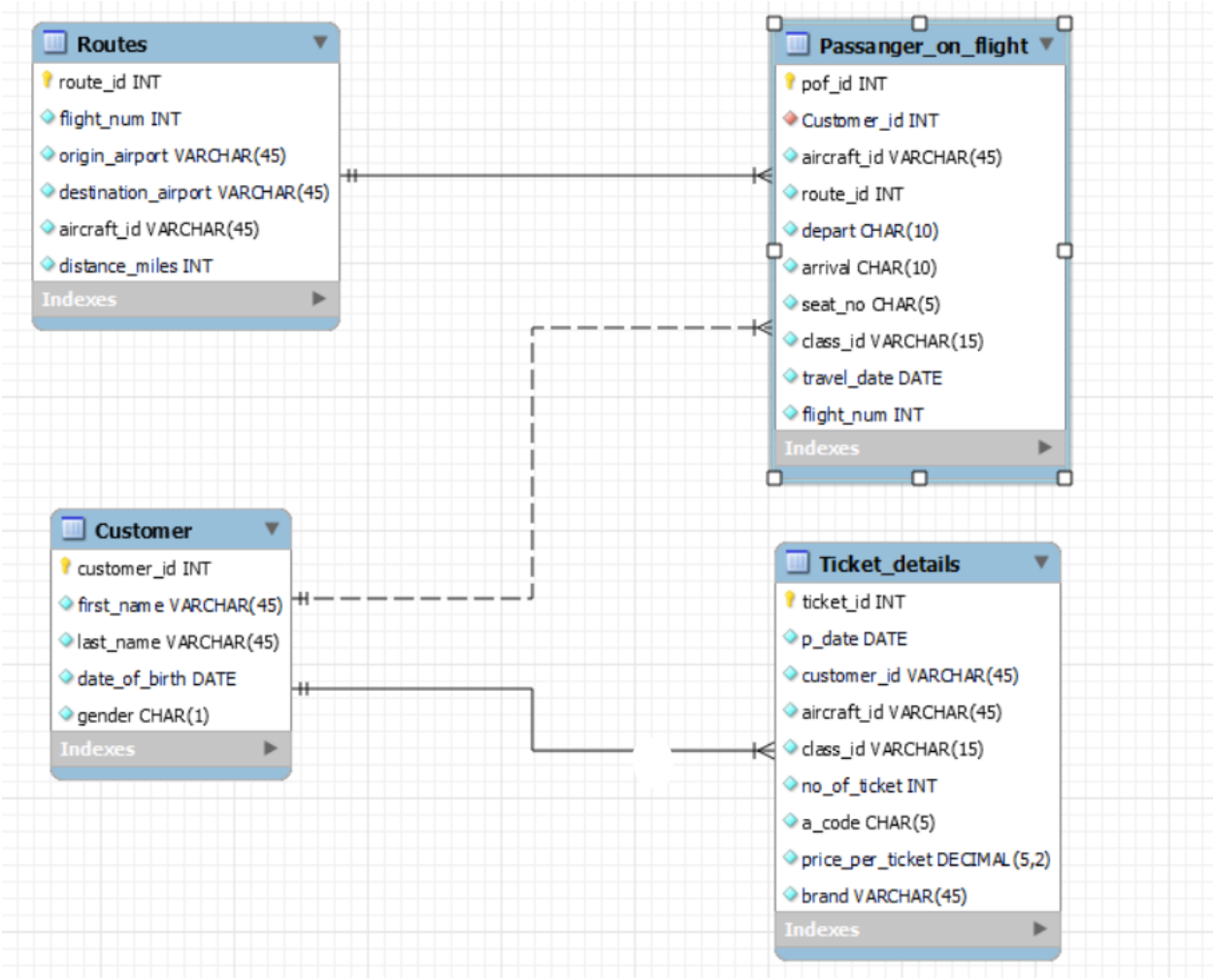
Air Cargo is an aviation company that provides air transportation services for passengers and freight. Air Cargo uses its aircraft to provide different services with the help of partnerships or alliances with other airlines. The company wants to prepare reports on regular passengers, busiest routes,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

ANS:

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



2. Create table and insert data

```
create table if not exists Customer
(
        customer_id int not null auto_increment primary key,
  first_name varchar(45) not null,
  last_name varchar(45) not null,
  date_of_birth date not null,
```

gender char(1) not null

);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(45)	NO		NULL	
last_name	varchar(45)	NO		NULL	
date_of_birth	date	NO		NULL	
gender	char(1)	NO		NULL	

create table if not exists routes

(

route_id int not null unique primary key,

flight_num int constraint chk_1 check (flight_num is not null),

origin_airport char(3) not null,

destination_airport char(3) not null,

aircraft_id varchar(10) not null,

distance_miles int not null constraint check_2 check (distance_miles >0)

);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
route_id	int	NO	PRI	NULL	
flight_num	int	YES		NULL	
origin_airport	char(3)	NO		NULL	
destination_airport	char(3)	NO		NULL	
aircraft_id	varchar(10)	NO		NULL	
distance_miles	int	NO		NULL	

create table if not exists Pof

(

```
        pof_id int auto_increment primary key,
  customer_id int not null,
  aircraft_id varchar(10) not null,
  route_id int not null,
  depart char(3) not null,
  arrival char(3) not null,
  seat_num char(4) not null,
  class_id varchar(15) not null,
  travel_date date not null,
  flight_num int not null,
  constraint fk_pof foreign key (customer_id) references Customer(customer_id)
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
pof_id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO	MUL	NULL	
aircraft_id	varchar(10)	NO		NULL	
route_id	int	NO		NULL	
depart	char(3)	NO		NULL	
arrival	char(3)	NO		NULL	
seat_num	char(4)	NO		NULL	
class_id	varchar(15)	NO		NULL	
travel_date	date	NO		NULL	
flight_num	int	NO		NULL	

```
create table if not exists Ticket_details
(
        tkt_id int auto_increment primary key,
  p_date date not null,
```

customer_id int not null,

aircraft_id varchar(10) not null,

class_id varchar(15) not null,

no_of_tkts int not null,

a_code char(3) not null,

price_per_tkts decimal (5,2) not null,

brand varchar(30) not null,

constraint fk_tkt_dts foreign key (customer_id) references Customer(customer_id)

);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
tkt_id	int	NO	PRI	NULL	auto_increment
p_date	date	NO		NULL	
customer_id	int	NO	MUL	NULL	
aircraft_id	varchar(10)	NO		NULL	
class_id	varchar(15)	NO		NULL	
no_of_tkts	int	NO		NULL	
a_code	char(3)	NO		NULL	
price_per_tkts	decimal(5,2)	NO		NULL	
brand	varchar(30)	NO		NULL	

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 Customer where customer_id in (select distinct customer_id from Pof where route_id between 1 and 25) order by customer_id;

```
select * from Customer where customer_id in (select distinct customer_id from Pof where  route_id between 1 and 25) order by customer_id;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name	date_of_birth	gender
1	Julie	Sam	1989-01-12	F
2	Steve	Ryan	1983-03-04	M
4	Cathenna	Emily	1977-09-14	F
5	Aaron	Kim	1991-02-18	M
7	Anderson	Stewart	1992-01-11	M
9	Leo	Travis	1994-03-22	M
10	Melvin	Tracy	1995-04-23	M
11	Roger	Walson	1996-05-24	M
13	Solomon	Walter	1998-07-26	M
15	Linda	William	1986-09-28	F
17	Catherine	Shad	1988-11-09	F
18	Gloria	Richie	1989-12-04	F

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

select count(distinct customer_id) as num_passangers, sum(no_of_tkts*price_per_tkts) as total_revenue from Ticket_details where

class_id="Bussiness";

```
select count(distinct customer_id) as  num_passangers, sum(no_of_tkts*price_per_tkts) as total_revenue from Ticket_details where
class_id="Bussiness";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_passangers	total_revenue
11	6034.00

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 Customer;

```
275 •  select concat(first_name, "", last_name) as Full_name from Customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Full_name
JulieSam
SteveRyan
MorrisLois
CathennaEmily
AaronKim
AlexanderScot
AndersonStewart
FloydTed
LeoTravis
MelvinTracy
RogerWalson
ShirleyWally

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

<u>select first_name, last_name from Customer where customer_id in (select distinct Customer.customer_id from Ticket_details);</u>

```
276
277 •  select first_name, last_name from Customer where customer_id in (select distinct Customer.customer_id from Ticket_details);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name
Julie	Sam
Steve	Ryan
Morris	Lois
Cathenna	Emily
Aaron	Kim
Alexander	Scot
Anderson	Stewart
Floyd	Ted
Leo	Travis
Melvin	Tracy
Roger	Walson
Shirley	Wally

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 first_name, last_name from Customer where customer_id in (select distinct customer_id from Ticket_details where brand="Emirates");

```
select first_name, last_name from Customer where customer_id in (select distinct customer_id from Ticket_details where brand="Emirates");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

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 * from Customer a inner join

(select distinct customer_id from Pof where class_id="Economy Plus") b

on a.customer_id = b.customer_id;

```
select * from Customer a inner join
(select distinct customer_id from Pof where class_id="Economy Plus") b
on a.customer_id = b.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	first_name	last_name	date_of_birth	gender	customer_id
1	Julie	Sam	1989-01-12	F	1
8	Floyd	Ted	1993-02-21	M	8
11	Roger	Walson	1996-05-24	M	11
17	Catherine	Shad	1988-11-09	F	17
19	Joyce	Paul	1990-06-02	F	19
22	Pheny	Eri	1999-01-29	M	22
32	Chirstoper	Sean	1993-06-21	M	32
47	Sophia	Carl	1999-08-11	F	47
50	Rose	Arthur	1996-05-23	F	50

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

select if((select sum(no_of_tkts*price_per_tkts) as total_revenue from Ticket_details) > 10000, 'completed 10K','not crossed 10K') as revenue_check;

```
select if((select sum(no_of_tkts*price_per_tkts) as total_revenue from Ticket_details) > 10000, 'completed 10K','not crossed 10K')
as revenue_check;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

revenue_check
completed 10K

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

Create user if not exists 'Akshay'@'127.0.0.1' identified by 'password';

grant all privileges on air_cargo to Akshay@127.0.0.1;

244	13:43:39	Create user if not exists 'Akshay'@'127.0.0.1' identified by 'password'	0 row(s) affected
245	13:43:48	grant all privileges on air_cargo to Akshay@127.0.0.1	0 row(s) affected

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

select class_id, max(price_per_tkts) from Ticket_details group by class_id;

select distinct class_id, max(price_per_tkts) over (partition by class_id) as max_price from Ticket_details order by max_price;

```
select class_id, max(price_per_tkts) from Ticket_details group by class_id;

select distinct class_id, max(price_per_tkts) over (partition by class_id) as max_price from Ticket_details order by max_price;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

class_id	max_price
Economy	190.00
Economy Plus	295.00
First Class	395.00
Bussiness	510.00

---

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

explain select * from Pof where route_id='4';

```
explain select * from Pof where route_id='4';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Pof	NULL	ALL	NULL	NULL	NULL	NULL	50	10.00	Using where

---

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.

create index idx_rid on Pof(route_id);

explain select * from Pof where route_id='4';

```
create index idx_rid on Pof(route_id);
explain select  * from  Pof where route_id='4';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Pof	NULL	ref	idx_rid	idx_rid	4	const	3	100.00	NULL

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

select customer_id, aircraft_id, sum(price_per_tkts * no_of_tkts) as total_price from Ticket_details group by customer_id, aircraft_id

order by customer_id, aircraft_id;

```
select customer_id, aircraft_id, sum(price_per_tkts * no_of_tkts) as total_price from Ticket_details group by customer_id, aircraft_id
order by customer_id, aircraft_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	aircraft_id	total_price
1	CRJ900	320.00
1	ERJ142	250.00
2	767-301ER	130.00
2	A321	505.00
4	767-301ER	780.00
5	767-301ER	430.00
5	ERJ142	240.00
7	767-301ER	430.00
8	A321	465.00
9	767-301ER	380.00
9	CRJ900	390.00
10	A321	135.00

select customer_id, aircraft_id, sum(price_per_tkts * no_of_tkts) as total_price from Ticket_details group by customer_id, aircraft_id

with rollup order by customer_id, aircraft_id;

```
select customer_id, aircraft_id, sum(price_per_tkts * no_of_tkts) as total_price from Ticket_details group by customer_id, aircraft_id
with rollup order by customer_id, aircraft_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	aircraft_id	total_price
NULL	NULL	15369.00
1	NULL	570.00
1	CRJ900	320.00
1	ERJ142	250.00
2	NULL	635.00
2	767-301ER	130.00
2	A321	505.00
4	NULL	780.00
4	767-301ER	780.00
5	NULL	670.00
5	767-301ER	430.00
5	ERJ142	240.00

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

```
create view buss_class_customer as
 select a.*,b.brand from Customer a
 inner join (select distinct customer_id, brand from Ticket_details where class_id='Bussiness' order by customer_id) b
 on a.customer_id=b.customer_id;

 select * from buss_class_customer;
```

```
create view buss_class_customer as
select a.*,b.brand from Customer a
inner join (select distinct customer_id, brand from Ticket_details where class_id='Bussiness' order by customer_id) b
on a.customer_id=b.customer_id;

select * from buss_class_customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

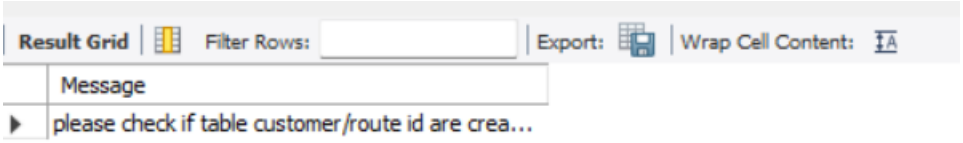
customer_id	first_name	last_name	date_of_birth	gender	brand
2	Steve	Ryan	1983-03-04	M	Qatar Airways
5	Aaron	Kim	1991-02-18	M	Emirates
7	Anderson	Stewart	1992-01-11	M	Emirates
11	Roger	Walson	1996-05-24	M	Emirates
15	Linda	William	1986-09-28	F	Qatar Airways
21	Chirsty	Josh	2004-01-10	M	Bristish Airways
24	Calvin	Willis	1994-02-05	M	Qatar Airways
25	Moss	Morris	2011-02-18	M	Emirates
29	Watson	Ronald	1991-01-11	M	Jet Airways
29	Watson	Ronald	1991-01-11	M	Qatar Airways
33	Mark	Ethan	1994-05-22	M	Bristish Airways
49	Russell	Peter	1996-06-01	M	Emirates

buss_class_customer26

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.

```
delimiter //
 create procedure check_route(in rid varchar(255))
 begin
        declare TableNotFound condition for 1146;
  declare exit handler for TableNotFound
                select  'please check if table customer/route id are created - one/both are missing' Message;
    set @query = concat( 'select * from Customer where customer_id  in (select distinct customer_id from Pod where route_id
    in (', rid,'));');
    prepare  sql_query  from @query;
    execute sql_query;
    end//
```

```
delimiter ;

call check_route("1,5");
```



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 check_dist()

begin

select * from Routes where distance_miles > 2000;

end //

delimiter ;


call check_dist;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

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.

```
select flight_num,distance_miles, case
                    when distance_miles between 0 and 2000 then "SDT"
    when distance_miles between 2001 and 6500 then "IDT"
    else "LDT"
        end distance_category from Routes;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

flight_num	distance_miles	distance_category
1111	4962	IDT
1112	4962	IDT
1113	3466	IDT
1114	2475	IDT
1115	2475	IDT
1116	2556	IDT
1117	1745	SDT
1118	719	SDT
1119	862	SDT
1120	3365	IDT
1122	4300	IDT
1123	2232	IDT

Result 30

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*

```
select p_date,customer_id, class_id, case
                                when class_id in ('Bussiness','Economy plus') then "Yes"
        else "No"
        end as complimentry_services from Ticket_details;
```

```
select p_date,customer_id, class_id, case
                when class_id in ('Bussiness','Economy plus') then "Yes"
                else "No"
                end as complimentry_services from Ticket_details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

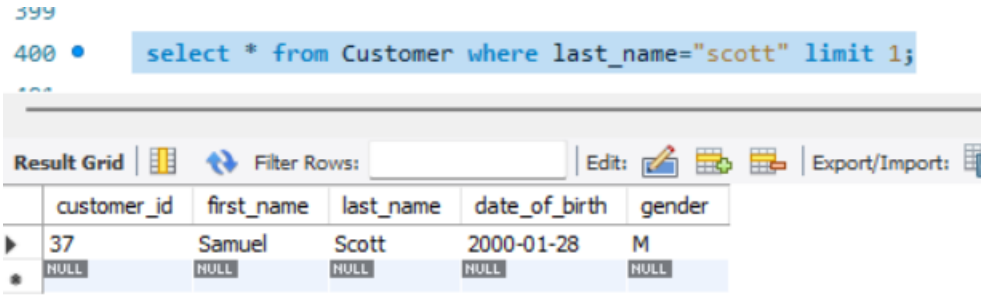
p_date	customer_id	class_id	complimentry_services
2018-12-26	27	Economy	No
2020-02-02	22	Economy Plus	Yes
2020-03-03	21	Bussiness	Yes
2020-04-04	4	First Class	No
2020-05-05	5	Economy	No
2020-07-07	7	Bussiness	Yes
2020-08-08	8	Economy Plus	Yes
2020-09-09	9	First Class	No
2020-10-10	10	Economy	No
2020-11-11	11	Bussiness	Yes
2020-12-12	19	Economy Plus	Yes
2019-01-01	13	First Class	No

```
delimiter //

create function check_comp_serv(cls varchar(15))

returns char(3)
```

```
deterministic
begin
        declare comp_ser char(3);
  if cls in ('Bussiness','Economy plus') then
                set comp_ser ='Yes';
  else
                set comp_ser='No';
  end if;
                return (copm_ser);
end //


create procedure check_comp_serv_proc()
begin
                select p_date,customer_id, class_id, check_comp_serv(class_id) as
complimentry_services from Ticket_details;
end //
delimiter ;
```

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.

```
select * from Customer where last_name="scott" limit 1;
```



```
delimiter //
 create procedure cust_lname_scott()
 begin
            declare c_id int;
   declare f_name varchar(20);
   declare l_name varchar(20);
   declare dob date;
   declare gen char(1);

   declare cust_rec cursor
   for
   select  * from  Customer where last_name ="Scott";

   create table if not exists cursor_table
   (
                        c_id int,
      f_name varchar(20),
      l_name varchar(20),
      dob date,
      gen char(1)
            );

   open cust_rec;
   fetch cust_rec  into c_id, f_name, l_name, dob, gen ;
   insert into cursor_table(c_id, f_name, l_name, dob, gen) values (c_id, f_name, l_name, dob,gen);
   close cust_rec;

   select * from cursor_table;
   end //
```

```
delimiter ;

call cust_lname_scott();
```

Project Submitted by – Mayur Nivadekar