

# 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.

## 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

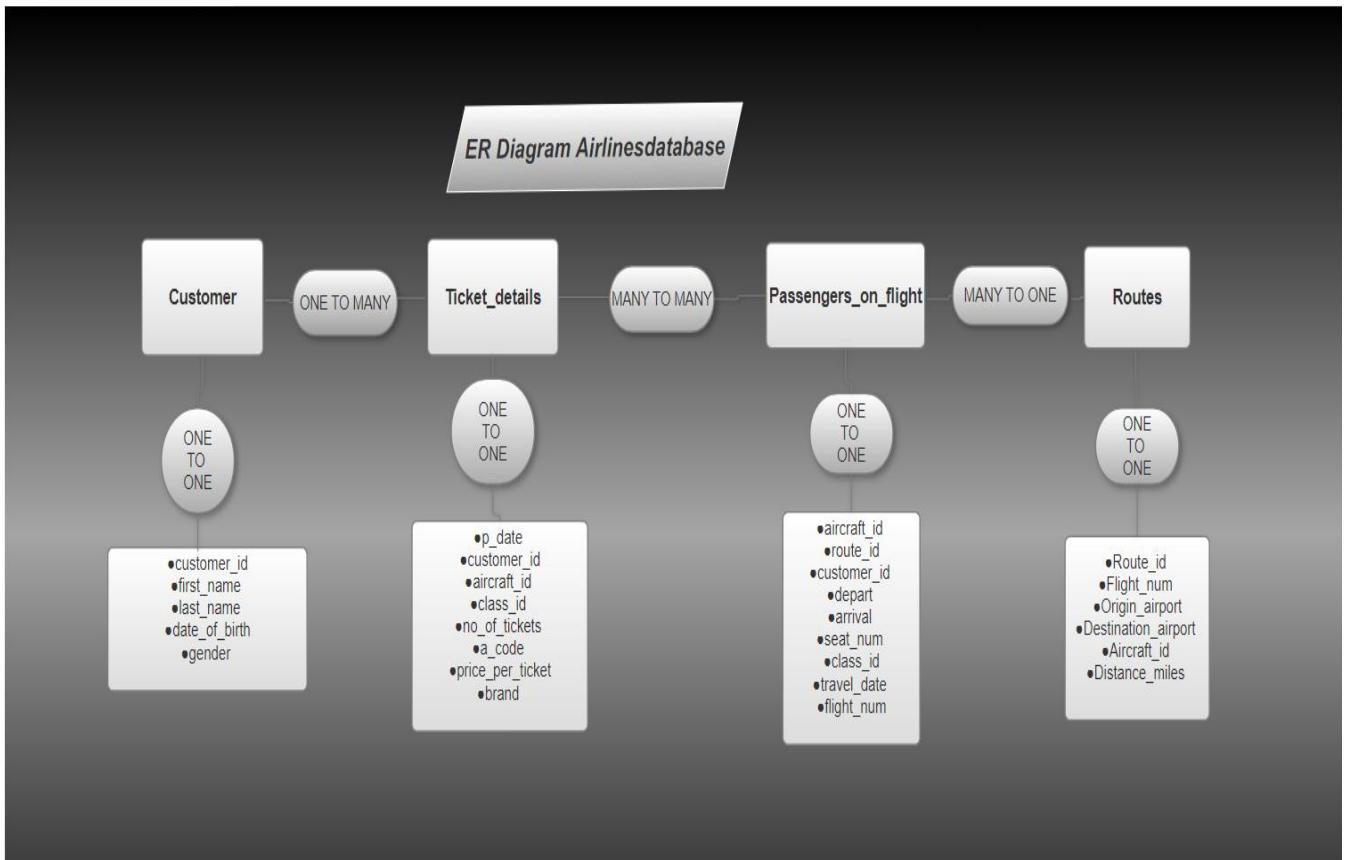
- p\_date – Ticket purchase date
- customer\_id – ID of the customer
- aircraft\_id – ID of each aircraft in a brand
- class\_id – ID of travel class
- no\_of\_tickets – Number of tickets purchased
- a\_code – Code of each airport
- price\_per\_ticket – Price of a ticket
- brand – Aviation service provider for each aircraft

**routes:** Contains information about the route details

- Route\_id – Route ID of from and to location
- Flight\_num – Specific 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

**Following operations should be performed:**

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



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 Database Aircargo;
```

```
show Databases;
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Navigator' pane on the left shows the 'SCHEMAS' tree, with 'aircargo' expanded to show 'Tables', 'Views', 'Stored Procedures', and 'Functions'. A red box highlights the 'aircargo' schema. The 'SQL File 6' tab contains the SQL commands: 'CREATE Database Aircargo;' and 'show Databases;'. The 'Result Grid' pane below shows a list of databases: 'aircargo', 'db', 'employee', 'information\_schema', 'mysql', 'performance\_schema', 'sqlbasics', and 'sys'. The 'aircargo' database is highlighted with a red box. On the right side, there are icons for 'Result Grid', 'Form Editor', and 'Field Types'.

## customer

1. use Aircargo;  
create table customer(  
customer\_id int not null primary key,  
first\_name varchar(30) not null,  
last\_name varchar(30) not null,  
date\_of\_birth date not null,  
gender char(1) not null);  
describe customer;

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Navigator' pane on the left shows the 'aircargo' schema, which contains a 'Tables' folder containing a 'customer' table. The 'SQL File 6\*' tab is active, displaying the SQL code for creating the 'customer' table:

```

1 • use Aircargo;
2 • create table customer(
3     customer_id int not null primary key,
4     first_name varchar(30) not null,
5     last_name varchar(30) not null,
6     date_of_birth date not null,
7     gender char(1) not null
8 );
9 • describe customer;

```

The 'Result Grid' tab below shows the structure of the 'customer' table:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI		MULL
first_name	varchar(30)	NO			MULL
last_name	varchar(30)	NO			MULL
date_of_birth	date	NO			MULL
gender	char(1)	NO			MULL

use aircargo;

load data infile'C:\\Program Files\\MySQL\\MySQL Server 8.0\\Uploads\\customer.csv'  
into table customer

fields terminated by ',' enclosed by "" lines terminated by '\\n' ignore 1 rows;

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'aircargo' schema, which contains a 'Tables' folder containing a 'customer' table. The 'SQL File 4\*' tab is active, displaying the SQL code for loading data from a CSV file into the 'customer' table:

```

1 use aircargo;
2 • load data infile'C:\\Program Files\\MySQL\\MySQL Server 8.0\\Uploads\\customer.csv'
3 into table customer
4 fields terminated by ',' enclosed by "" lines terminated by '\\n' ignore 1 rows;

```

The 'Result Grid' tab below shows the data inserted into the 'customer' table:

customer_id	first_name	last_name	date_of_birth	gender
1	Julie	Sam	12-01-1989	F
2	Steve	Ryan	03-04-1983	M
3	Morris	Lois	09-12-1993	M
4	Cathenna	Emily	14-09-1977	F
5	Aaron	Kim	18-02-1991	M
6	Alexander	Scot	12-02-1985	M
7	Anderson	Stewart	11-01-1992	M
8	Floyd	Ted	21-02-1993	M

## Tickets\_detail

```
use Aircargo;
create table tickets_detail(
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(30) not null,
no_of_tkts int not null,
price_per_tkt decimal(5,2)not null,
brand varchar(30) not null,
constraint fk_tkt_dts foreign key(customer_id) references customer(customer_id)
);

```

```
Describe aircargo.tickets_detail;
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard database management tools like New Table, New View, etc.
- Navigator:** Shows the schema 'aircargo' containing tables like customer, pof, routes, tickets\_detail, and views.
- SQL Editor:** SQL File 4\* tab, displaying the SQL code for creating the table and its description.
- Result Grid:** Shows the structure of the 'tickets\_detail' table with columns: tkt\_id, p\_date, customer\_id, aircraft\_id, class\_id, no\_of\_tkts, price\_per\_tkt, and brand.
- Status Bar:** Result 6 x, Read Only.

```
use aircargo;
load data infile'C:\\Program Files\\MySQL\\MySQL
8.0\\Uploads\\ticket_details.csv'into table tickets_detail
fields terminated by ',' enclosed by """
lines terminated by '\\n' ignore 1 rows
```

Server

The screenshot shows the MySQL Workbench interface. In the top tab bar, the schema 'aircargo' is selected. The SQL Editor contains the following code:

```

1 use aircargo;
2 • load data infile'C:\Program Files\MySQL\MySQL Server 8.0\Uploads\ticket_details.csv'into table tickets_detail
3 fields terminated by ',' enclosed by ""
4 lines terminated by '\n' ignore 1 rows

```

The Result Grid displays the data loaded from the CSV file:

p_date	customer_id	aircraft_id	class_id	no_of_tickets	a_code	Price_per_ticket	brand
26-12-2018	27	767-30 IER	Economy	1	DAL	130	Emirates
02-02-2020	22	ERJ142	Economy Plus	1	AGB	220	Jet Airways
03-03-2020	21	CRJ900	Business	1	BOH	490	British Airways
04-04-2020	4	767-30 IER	First Class	1	AGB	390	Emirates
05-05-2020	5	ERJ142	Economy	1	CTM	120	Jet Airways
07-07-2020	7	767-30 IER	Business	1	BFS	430	Emirates
08-08-2020	8	A321	Economy Plus	1	DAL	275	Qatar Airways
09-09-2020	9	767-30 IER	First Class	1	BOH	380	Emirates

## Routes

```

use Aircargo;
create table 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,
designation_airport char(10)not null,
aircraft_id varchar(30) not null, distance_miles int not null constraint check_2 check
(distance_miles>0) Describe routes;

```

The screenshot shows the MySQL Workbench interface. In the top tab bar, the schema 'aircargo' is selected. The SQL Editor contains the following code:

```

1 • use Aircargo;
2 • create table routes(
3     route_id int not null unique primary key,
4     flight_num int constraint chk_1 check (flight_num is not null),
5     origin_airport char(3) not null,
6     designation_airport char(10)not null,
7     aircraft_id varchar(30) not null,
8     distance_miles int not null constraint check_2 check (distance_miles>0)
9 );
10 • describe routes;

```

The Result Grid displays the structure of the 'routes' table:

Field	Type	Null	Key	Default	Extra
route_id	int	NO	PRI	NULL	
flight_num	int	YES		NULL	
origin_airport	char(3)	NO		NULL	
designation_airport	char(10)	NO		NULL	
aircraft_id	varchar(30)	NO		NULL	
distance_miles	int	NO		NULL	

```

use aircargo;
load data infile'C:\Program Files\MySQL\MySQL Server 8.0\Uploads\routes.csv'
into table routes
fields terminated by ',' enclosed by "" lines terminated by '\n' ignore 1 rows;
SELECT * FROM aircargo.routes;

```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, which includes the 'aircargo' schema containing 'Tables' like 'customer', 'routes', and 'tickets\_detail'. The main area has tabs for 'aircargo', 'SQL File 4\*', 'ticket\_details', and 'tickets\_detail'. The 'SQL File 4\*' tab contains the following SQL code:

```

1 • use aircargo;
2 • load data infile'C:\Program Files\MySQL\MySQL Server 8.0\Uploads\routes.csv'
3 into table routes
4 fields terminated by ',' enclosed by '\"' lines terminated by '\n' ignore 1 rows;
5 • SELECT * FROM aircargo.routes;

```

The 'Result Grid' tab shows the data from the 'routes' table, which lists 19 flight routes with columns: route\_id, flight\_num, origin\_airport, designation\_airport, aircraft\_id, and distance\_miles. The data is as follows:

route_id	flight_num	origin_airport	designation_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
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	ABT	ADK	767-301ER	4300
13	1123	ADM	BQN	A321	2322
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

## Passengers\_on\_flights

```

use Aircargo;
create table 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(10)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);
describe pof;

```

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: aircargo\* SQL File 4\* ticket\_details tickets\_detail passengers\_on\_flights pof

SCHEMAS Filter objects

aircargo Tables customer pof routes tickets\_detail ticket\_details Views Stored Procedures Functions db employee Tables

Administration Schemas Information

No object selected

```

1 • use Aircargo;
2 • create table pof(
3     pof_id int auto_increment primary key,
4     customer_id int not null,
5     aircraft_id varchar(10) not null,
6     route_id int not null,
7     depart char(10)not null,
8     arrival char(3) not null,
9     seat_num char(4) not null,
10    class_id varchar(15) not null,
11    travel_date date not null,
12    flight_num int not null);
13 • describe pof;
```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 5 ×

Field	Type	Null	Key	Default	Extra
pof_id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO		NULL	
aircraft_id	varchar(10)	NO		NULL	
route_id	int	NO		NULL	
depart	char(10)	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	

Result Grid Form Editor Field Types Read Only

```

use aircargo;
load data infile'C:\Program
Files\MySQL\MySQL
8.0\Uploads\passenger
s_on_flights.csv'
into table pof
fields terminated by ',' enclosed by "" lines terminated by '\n' ignore 1 rows;
```

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: aircargo\* SQL File 4\* employy routes customer passengers\_on\_flights ticket\_details

SCHEMAS Filter objects

aircargo Tables passengers\_on\_flights customer routes ticket\_details Views Stored Procedures Functions db employee sqlbascs sys

Administration Schemas Information

rs\_on\_flights 1 ×

```

1 • use aircargo;
2 • load data infile'C:\Program Files\MySQL\MySQL Server 8.0\Uploads\passenger
s_on_flights.csv'
3 • into table pof
4 • fields terminated by ',' enclosed by "" lines terminated by '\n' ignore 1 rows;
```

Result Grid | Filter Rows: Export: Wrap Cell Content: rs\_on\_flights 1 ×

customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
2	A321	34	CRW	COD	01B	Business	26-01-2019	1117
2	767-301ER	4	JFK	LAX	01E	Economy	02-09-2018	1114
1	ERJ142	9	DEN	LAX	01EP	Economy Plus	26-12-2019	1119
1	CRJ900	30	BUR	STT	01FC	First Class	04-11-2018	1140
5	767-301ER	12	ABD	ADK	02B	Business	02-07-2018	1122
5	ERJ142	18	ANI	BGR	02E	Economy	06-05-2020	1128
8	A321	38	CST	DAL	02EP	Economy Plus	09-08-2020	1148
4	767-301ER	5	LAX	JFX	02FC	First Class	06-04-2020	1115

Result Grid Form Editor Read Only

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.

```
use aircargo;
select route_id, customer_id from passengers_on_flights
where route_id between 1 and 25
order by route_id, customer_id;
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'Navigator' and 'Schemas' sections, with 'aircargo' selected. The central pane shows the SQL query and its results. The query is:

```
1 • use aircargo;
2 • select route_id, customer_id from passengers_on_flights
3 where route_id between 1 and 25
4 order by route_id, customer_id;
5
```

The results grid shows the following data:

route_id	customer_id
1	18
4	2
4	4
4	11
5	4
5	11
8	46
9	1
9	29
10	10
12	5
13	13
13	17
14	15
14	24
15	9
15	44
15	49

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

```
use aircargo;
select class_id, count(*) as passenger, sum(no_of_tickets * price_per_ticket) as revenue
from ticket_details
where class_id = 'Business'
group by class_id;
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'Navigator' and 'Schemas' sections, with 'aircargo' selected. The central pane shows the SQL query and its results. The query is:

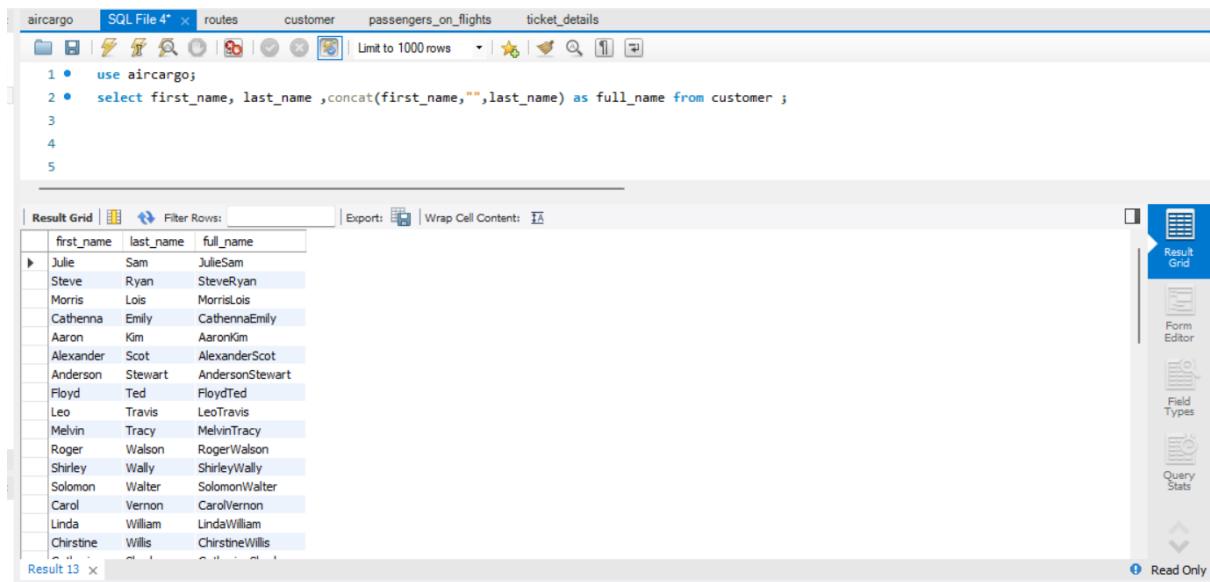
```
1 • use aircargo;
2 • select class_id, count(*) as passenger, sum(no_of_tickets * price_per_ticket) as revenue
3 where class_id = 'Business'
4 group by class_id;
5
6
```

The results grid shows the following data:

class_id	passenger	revenue
Business	13	6034

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 first_name, last_name ,concat(first_name,"",last_name) as full_name  
from customer ;
```



The screenshot shows the SQL Developer interface with a query editor and a result grid. The query is:

```
1 • use aircargo;  
2 • select first_name, last_name ,concat(first_name,"",last_name) as full_name from customer ;  
3  
4  
5
```

The result grid displays the following data:

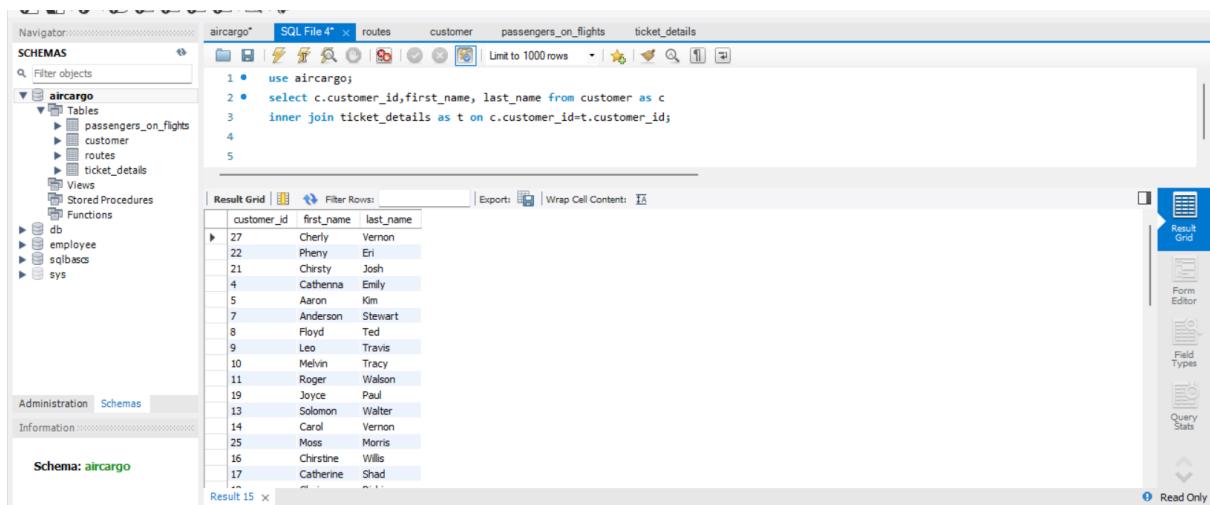
first_name	last_name	full_name
Julie	Sam	JulieSam
Steve	Ryan	SteveRyan
Morris	Lois	MorrisLois
Cathenna	Emily	CathennaEmily
Aaron	Kim	AaronKim
Alexander	Scot	AlexanderScot
Anderson	Stewart	AndersonStewart
Floyd	Ted	FloydTed
Leo	Travis	LeoTravis
Melvin	Tracy	MelvinTracy
Roger	Walson	RogerWalson
Shirley	Wally	ShirleyWally
Solomon	Walter	SolomonWalter
Carol	Vernon	CarolVernon
Linda	William	LindaWilliam
Christine	Willis	ChristineWillis
Chad	Glenda	ChadGlenda

Result 13 x Read Only

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,first_name, last_name from customer as c  
inner join ticket_details as t on c.customer_id=t.customer_id;
```



The screenshot shows the SQL Developer interface with a query editor and a result grid. The query is:

```
1 • use aircargo;  
2 • select c.customer_id,first_name, last_name from customer as c  
3 inner join ticket_details as t on c.customer_id=t.customer_id;  
4  
5
```

The result grid displays the following data:

customer_id	first_name	last_name
27	Cherly	Vernon
22	Pheny	Eri
21	Christy	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	Christine	Willis
17	Catherine	Shad

Result 15 x Read Only

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 c.customer_id,first_name, last_name, t.brand from customer as c
inner join ticket_details as t on c.customer_id=t.customer_id
where brand = 'Emirates';
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'aircargo' schema with its tables: routes, customer, passengers\_on\_flights, and ticket\_details. The right pane shows the results of the executed query:

```
customer_id | first_name | last_name | brand
2           | Steve      | Ryan     | Emirates
4           | Cathenna   | Emily    | Emirates
4           | Cathenna   | Emily    | Emirates
5           | Aaron      | Kim      | Emirates
7           | Anderson   | Stewart  | Emirates
9           | Leo        | Travis   | Emirates
11          | Roger      | Wilson   | Emirates
11          | Roger      | Wilson   | Emirates
14          | Carol      | Vernon   | Emirates
18          | Gloria     | Richie   | Emirates
18          | Gloria     | Richie   | Emirates
19          | Joyce      | Paul     | Emirates
25          | Moss       | Morris   | Emirates
25          | Moss       | Morris   | Emirates
27          | Cherly     | Vernon   | Emirates
31          | James      | Robert   | Emirates
```

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.

```
use aircargo;
select class_id, customer_id from passengers_on_flights
where class_id ='Economy plus'
GROUP BY customer_id having count(*)
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'aircargo' schema with its tables: routes, customer, ticket\_details, and passengers\_on\_flights. The right pane shows the results of the executed query:

```
class_id | customer_id
Economy Plus | 1
Economy Plus | 8
Economy Plus | 11
Economy Plus | 17
Economy Plus | 19
Economy Plus | 22
Economy Plus | 32
Economy Plus | 47
Economy Plus | 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_tickets*price_per_ticket)
as revenue from ticket_details)> 10000, 'yes','no')as crossed_10000 ;
```

	crossed_10000
	yes

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

```
use aircargo;
create user if not exists 'khushboo' identified by 'password@123';
grant all privileges on aircargo to khushboo;
```

Action	Time	Action	Message
1	19:44:40	use aircargo	0 row(s) affected
2	19:44:40	create user if not exists 'khushboo' identified by 'password@123'	0 row(s) affected, 1 warning(s): 3163 Authorization ID 'khushboo'@'%' already exists.
3	19:44:40	grant all privileges on aircargo to khushboo	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.

```
use aircargo;
select class_id ,max(price_per_ticket) from ticket_details
GROUP BY class_id;
```

The screenshot shows the SQL Server Management Studio interface. The top pane displays the SQL query:

```
aircargo" SQL File 4* routes customer passengers_on_flights ticket_details
1 • use aircargo;
2 • select class_id ,max(price_per_ticket) from ticket_details
3 GROUP BY class_id;
4
5
6
```

The bottom pane shows the result grid:

class_id	max(price_per_ticket)
Economy	190
Economy Plus	295
Business	510
First Class	395

Result 6 x

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.

```
select * from passengers_on_flights
where route_id = 4;
```

The screenshot shows the SQL Server Management Studio interface. The top pane displays the SQL query:

```
aircargo" SQL File 4* routes customer passengers_on_flights ticket_details
1 • use aircargo;
2 • select * from passengers_on_flights
3 where route_id = 4;
4
5
6
```

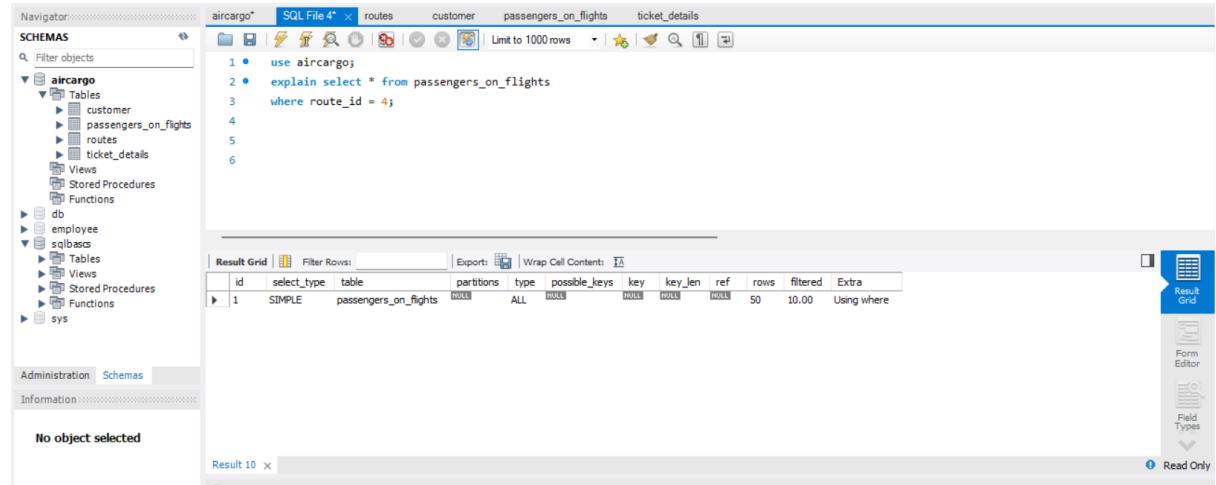
The bottom pane shows the result grid:

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	Business	09-11-2020	1114

passenger\_on\_flights 9 x

13. For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.

```
explain select * from passengers_on_flights  
where route_id = 4;
```



The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'Navigator' and 'Schemas' for the 'aircargo' database. The right pane shows the results of the executed query:

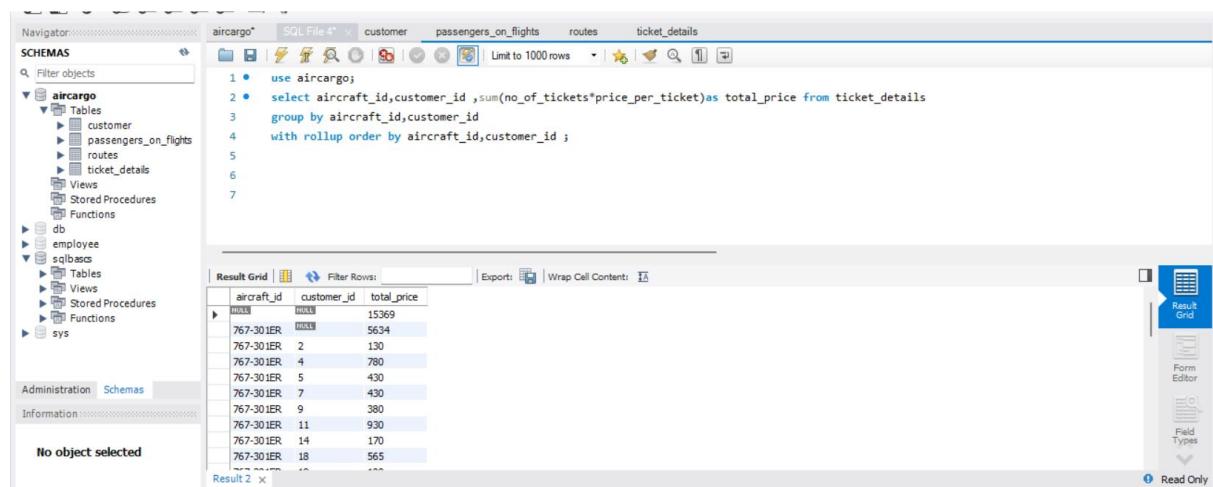
```
1 • use aircargo;  
2 • explain select * from passengers_on_flights  
3 where route_id = 4;  
4  
5  
6
```

Below the query results is a table titled 'Result Grid' with the following data:

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

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

```
use aircargo;  
select aircraft_id, customer_id ,sum(no_of_tickets*price_per_ticket)as total_price from ticket_details  
group by aircraft_id, customer_id  
with rollup order by aircraft_id, customer_id ;
```



The screenshot shows the SQL Server Management Studio interface. The left pane displays the 'Navigator' and 'Schemas' for the 'aircargo' database. The right pane shows the results of the executed query:

```
1 • use aircargo;  
2 • select aircraft_id, customer_id ,sum(no_of_tickets*price_per_ticket)as total_price from ticket_details  
3 group by aircraft_id, customer_id  
4 with rollup order by aircraft_id, customer_id ;  
5  
6  
7
```

Below the query results is a table titled 'Result Grid' with the following data:

	aircraft_id	customer_id	total_price
			15369
	767-30 IER		5634
	767-30 IER	2	130
	767-30 IER	4	780
	767-30 IER	5	430
	767-30 IER	7	430
	767-30 IER	9	380
	767-30 IER	11	930
	767-30 IER	14	170
	767-30 IER	18	565

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

```
use aircargo;
create view business_class_customers as
select c.customer_id, first_name, last_name,t.class_id,t.brand from customer as c
inner join ticket_details as t on c.customer_id =t.customer_id
where t.class_id = 'bussiness';
select*from business_class_customers ;
```

The screenshot shows a database interface with a 'Navigator' pane on the left containing 'SCHEMAS' and 'Tables' sections for 'aircargo'. The 'Tables' section lists 'customer', 'passengers\_on\_flights', 'routes', and 'ticket\_details'. The 'ticket\_details' table is selected. The main area shows a SQL editor with the following code:

```
1 • use aircargo;
2 • create view business_class_customers as
3 select c.customer_id, first_name, last_name,t.class_id,t.brand from customer as c
4 inner join ticket_details as t on c.customer_id =t.customer_id
5 where t.class_id = 'bussiness';
6 • select*from business_class_customers ;
7
```

Below the SQL editor is a 'Result Grid' table with the following data:

	customer_id	first_name	last_name	class_id	brand
▶	2	Steve	Ryan	Business	Qatar Airways
5	Aaron	Kim		Business	Emirates
7	Anderson	Stewart		Business	Emirates
11	Roger	Walson		Business	Emirates
11	Roger	Walson		Business	Emirates
15	Linda	Willan		Business	Qatar Airways
21	Christy	Josh		Business	Bristish Airways
24	Calvin	Wills		Business	Qatar Airways
25	Moss	Morris		Business	Emirates
29	Watson	Ronald		Business	Jet Airways
29	Watson	Ronald		Business	Qatar Airways
33	Mark	Ethan		Business	Bristish Airways
49	Russell	Peter		Business	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.

```
use aircargo;
delimiter &&
create procedure get_pasenger_by_route_range(
IN start_route int,
IN end_route int)
Begin declare error_msg varchar(260);
if not exists (select 1 from information_schema.tables where table_name =
'passengers_on_flights') then
set error_msg ='table flights does not exist';
signal sqlstate '45000' set message_text = error_msg;
end if;select* from passengers_on_flights where route_id between start_route and
end_route;
end &&
call get_pasenger_by_route_range(9,11);
```

```

1 • use aircargo;
2 delimiter &&
3 • create procedure get_pasenger_by_route_range(
4     IN start_route int,
5     IN end_route int
6 )
7 begin
8 declare error_msg varchar(260);
9 if not exists (select 1 from information_schema.tables where table_name = 'passengers_on_flights') then
10 set error_msg = 'Table flights does not exist';
11 signal sqlstate '45000' set message_text = error_msg;
12 end if;
13 select* from passengers_on_flights where route_id between start_route and end_route;
14 end 88
15 • call get_pasenger_by_route_range(9,11);
+-----+
| 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  |
|         10   |       A321   |     10   | HNL    | DEN    | 05E    | Economy    | 11-10-2020 |      1120  |
|         29   |      ERJ142  |      9    | DEN    | LAX    | 11B    | Business    | 03-05-2018 |      1119  |
+-----+

```

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.

```

use aircargo;
delimiter &&
create procedure route_distance()
begin select * from routes where distance_miles>2000 ;
end &&
call route_distance;

```

```

1 • use aircargo;
2 delimiter &&
3 • create procedure route_distance()
4 begin
5 select * from routes where distance_miles>2000 ;
6 end &&
7 • call route_distance;
+-----+
| route_id | flight_num | origin_airport | designation_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    |      ADK       |        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       |        BII        | A321        |      2425    |
|      34   |    1144    |      CRW       |        COD        | A321        |      2452    |
+-----+

```

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 fight_distance_category()
begin
declare distance_category varchar(20);
select flight_num,distance_miles,
case
when distance_miles between 0 and 2000 then "SDT"
when distance_miles between 20001 and 6500 then "IDT"
ELSE "LDT"
END as distance_category from routes;
end &&
call fight_distance_category()
```

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure under 'aircargo' with tables like customer, passengers\_on\_flights, routes, and ticket\_details.
- SQL File 4:** Contains the stored procedure code provided in the question.
- Result Grid:** Displays the output of the stored procedure, showing 14 rows of flight numbers, distance miles, and their corresponding distance categories.

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

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

```
use aircargo;
```

```
delimiter &&
```

```
create procedure complimentary_services ()
begin select p_date, customer_id, class_id, case
when class_id in ("bussiness", "economy plus") then "yes" else "No"
end as complimentary_services from ticket_details;
end &&
call complimentary_services ()
```

The screenshot shows the MySQL Workbench interface. On the left is the Navigator pane, which displays the schema structure of the 'aircargo' database, including tables like 'customer', 'passenger\_on\_flights', 'routes', and 'ticket\_details', as well as stored procedures and functions. The main area contains the SQL editor with the following code:

```

1 • use aircargo;
2 delimiter &&
3 • create procedure complimentary_services ()
4 begin
5     select p_date, customer_id, class_id, case
6         when class_id in ("bussiness", "economy plus") then "yes" else "No"
7     end as complimentary_services from ticket_details;
8 end &&
9 • call complimentary_services ();
10
11

```

Below the SQL editor is the Result Grid, which displays the output of the executed query. The data is as follows:

p_date	customer_id	class_id	complimentary_services
26-12-2018	27	Economy	No
02-02-2020	22	Economy Plus	yes
03-03-2020	21	Business	yes
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Business	yes
08-08-2020	8	Economy Plus	yes
09-09-2020	9	First Class	No
10-10-2020	10	Economy	No
11-11-2020	11	Business	yes
12-12-2020	19	Economy Plus	yes
01-01-2019	13	First Class	No
02-02-2019	14	Economy	No

The Result Grid has 15 rows. The right side of the interface includes various toolbars and buttons for managing the results.

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.

The screenshot shows two separate sessions in Oracle SQL Developer. Both sessions are connected to the 'aircargo' schema.

**Session 1 (Top):**

- Code:**

```

2 delimiter //
3 • create procedure cust_last_name_scott()
4 begin
5     declare customer_id int;
6     declare first_name varchar(20);
7     declare last_name varchar(20);
8     declare date_of_birth date ;
9     declare gender char(1);
10    declare cust_rec cursor
11    for
12        select * from customer where last_name = 'scot';
13    create table if not exists cursor_table(
14        customer_id int,
15        first_name varchar(20),
16        last_name varchar(20),
17        date_of_birth date ,
18        gender char(1)
19    );
20
21 end//
```
- Result Grid:**

customer_id	first_name	last_name	date_of_birth	gender
6	Alexander	Scot	12-02-1985	M

**Session 2 (Bottom):**

- Code:**

```

11 for
12 select * from customer where last_name = 'scot';
13 create table if not exists cursor_table(
14     customer_id int,
15     first_name varchar(20),
16     last_name varchar(20),
17     date_of_birth date ,
18     gender char(1)
19 );
20
21 open cust_rec;
22 fetch cust_rec into customer_id, first_name, last_name, date_of_birth, gender;
23 insert into cursor_table(customer_id, first_name, last_name, date_of_birth, gender) value(customer_id, first_name, last_name, date_of_birth, gender);
24 close cust_rec;
25 select * from cursor_table;
26 end//
```
- Result Grid:**

customer_id	first_name	last_name	date_of_birth	gender
6	Alexander	Scot	12-02-1985	M

