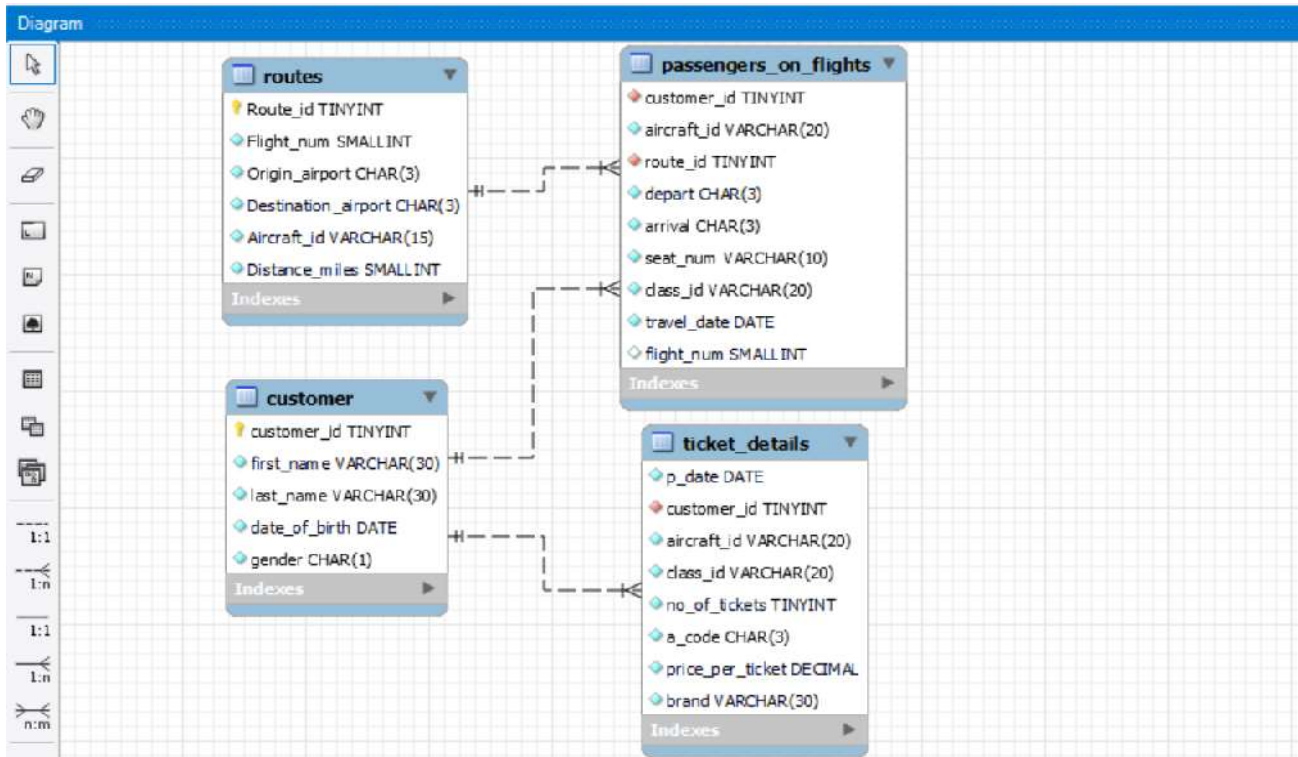
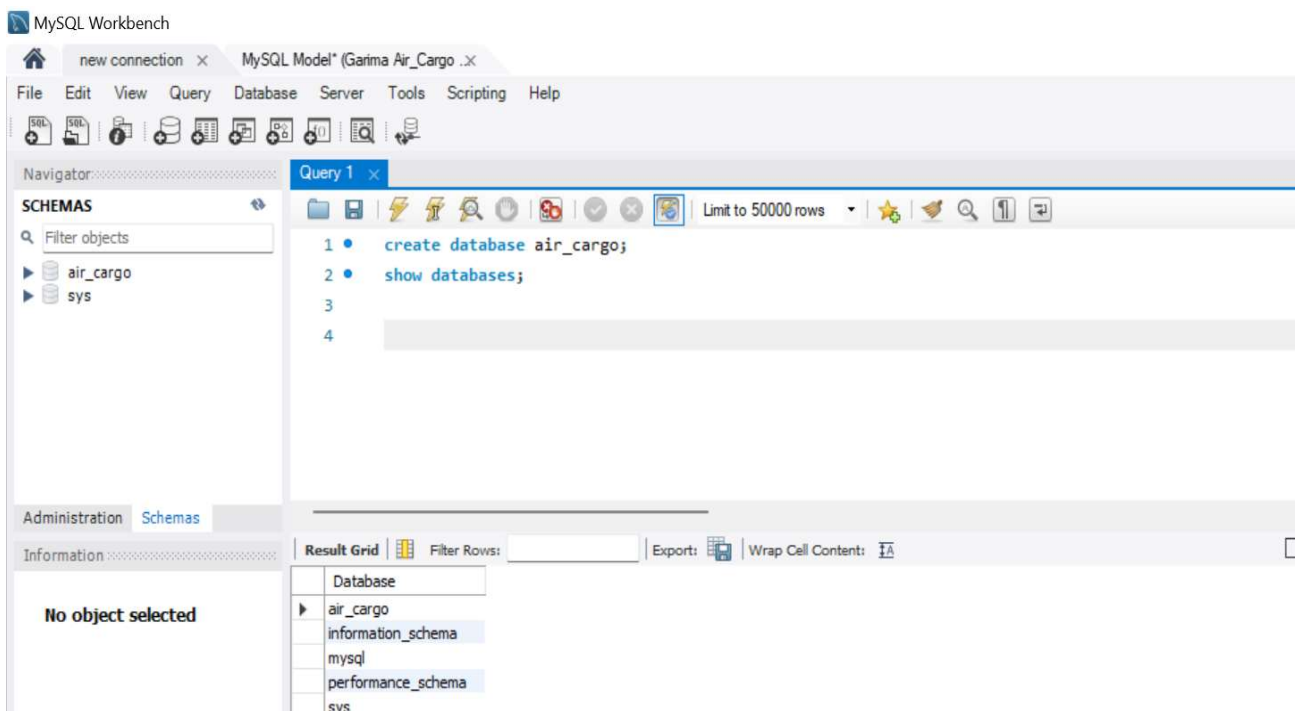


1) Create an ER diagram for the given airlines database.



2) 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.



MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

air_cargo

- Tables
- Views
- Stored Procedures
- Functions

sys

Administration Schemas

Information

No object selected

Query 1 x

Limit to 50000 rows

```
3 • use air_cargo;
4 • create table customer (
5     customer_id    tinyint not null primary key,
6     first_name      varchar(30) not null,
7     last_name       varchar(30) not null,
8     date_of_birth   date not null,
9     gender          char(1) not null
10 );
11 • describe customer;
```

Result Grid

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

MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

air_cargo

- Tables
- Views
- Stored Procedures
- Functions

sys

Administration Schemas

Information

No object selected

Query 1 x

Limit to 50000 rows

```
4 • create table customer (
5     customer_id    tinyint not null primary key,
6     first_name      varchar(30) not null,
7     last_name       varchar(30) not null,
8     date_of_birth   date not null,
9     gender          char(1) not null
10 );
11 • describe customer;
12 • select * from customer;
13
```

Result Grid

	customer_id	first_name	last_name	date_of_birth	gender
1	1	Julie	Sam	12-01-1989	F
2	2	Steve	Ryan	03-04-1983	M
3	3	Morris	Lois	09-12-1993	M
4	4	Cathenna	Emily	14-09-1977	F
5	5	Aaron	Kim	18-02-1991	M
6	6	Alexander	Scot	12-02-1985	M
7	7	Anderson	Stewart	11-01-1992	M
8	8	Floyd	Ted	21-02-1993	M
9	9	Leo	Travis	22-03-1994	M

Navigation pane showing Schemas: air_cargo (Tables: customer, passengers_on_flights, routes, ticket_details; Views; Stored Procedures; Functions; sys). Administration Schemas Information. No object selected.

Query 1:

```
10 );
11 describe customer;
12 select * from customer;
13
14 create table routes (
15     Route_id tinyint not null unique primary key,
16     Flight_num smallint constraint check_1 check (Flight_num is not null),
17     Origin_airport char(3) not null,
18     Destination_airport char(3) not null,
19     Aircraft_id varchar(15) not null,
20     Distance_miles smallint not null constraint check_2 check (distance_miles > 0)
21 );
22 describe routes;
```

Result Grid:

Field	Type	Null	Key	Default	Extra
route_id	int	YES		NULL	
flight_num	int	YES		NULL	
origin_airport	text	YES		NULL	
destination_airport	text	YES		NULL	
aircraft_id	text	YES		NULL	
distance_miles	int	YES		NULL	

MySQL Workbench interface showing the same database schema and query results.

Query 1:

```
16 Flight_num smallint constraint check_1 check (Flight_num is not null),
17 Origin_airport char(3) not null,
18 Destination_airport char(3) not null,
19 Aircraft_id varchar(15) not null,
20 Distance_miles smallint not null constraint check_2 check (distance_miles > 0)
21 );
22 describe routes;
23
24 select * from routes;
```

Result Grid:

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
1	1111	EVR	HNL	767-301ER	4962
2	1112	HNL	EVR	767-301ER	4962
3	1113	EVR	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	EVR	A321	719
9	1119	DEN	LAX	ERJ142	862
10	1120	HNL	DEN	A321	3365
12	1122	ABT	ADK	767-301ER	4300

Query Completed

System tray: 22°C Mostly clear, Search, 21:19 05-12-2022

MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1 x

Limit to 50000 rows

```
26 • create table passengers_on_flights (
27     customer_id    tinyint not null references customer(customer_id),
28     aircraft_id    varchar(20) not null,
29     route_id       tinyint not null references route(route_id),
30     depart         char(3) not null,
31     arrival        char(3) not null,
32     seat_num       varchar(10) not null,
33     class_id       varchar(20) not null,
34     travel_date    date not null,
35     flight_num     smallint not null);
36 • describe passengers_on_flights;
```

SCHEMAS

Filter objects

air_cargo

- Tables
- customer
- Views
- Stored Procedures
- Functions

sys

Administration Schemas

Information

No object selected

Field	Type	Null	Key	Default	Extra
customer_id	tinyint	NO		NULL	
aircraft_id	varchar(20)	NO		NULL	
route_id	tinyint	NO		NULL	
depart	char(3)	NO		NULL	
arrival	char(3)	NO		NULL	
seat_num	varchar(10)	NO		NULL	
class_id	varchar(20)	NO		NULL	
travel_date	date	NO		NULL	
flight_num	smallint	NO		NULL	

Navigator: Query 1 x

Limit to 50000 rows

```
32     seat_num       varchar(10) not null,
33     class_id       varchar(20) not null,
34     travel_date    date not null,
35     flight_num     smallint not null);
36 • describe passengers_on_flights;
37 • select * from passengers_on_flights;
38
39
```

SCHEMAS

Filter objects

air_cargo

- Tables
- customer
- passengers_on_flights
- routes
- Views
- Stored Procedures
- Functions

sys

Administration Schemas

Information

No object selected

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
▶	2	A321	34	CRW	COD	01B	Bussiness	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	ABI	ADK	02B	Bussiness	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
	7	767-301ER	20	AVL	BOI	03B	Bussiness	08-07-2020	1130
	5	ERJ142	22	BGR	BJI	03E	Economy	31-05-2020	1132
	11	ERJ142	31	BTM	CHA	03EP	Economy Plus	02-08-2018	1141
	4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114
	11	767-301ER	5	LAX	JFX	04B	Bussiness	12-11-2020	1115
	6	A321	42	CRW	BOI	04E	Economy	02-05-2019	1152

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

air_cargo

Tables

customer

passengers_on_flights

routes

Views

Stored Procedures

Functions

sys

Administration Schemas

Information

No object selected

Query 1

Limit to 50000 rows

```

39 • create table ticket_details (
40     p_date         date not null,
41     customer_id    tinyint not null references customer(customer_id),
42     aircraft_id     varchar(20) not null,
43     class_id        varchar(20) not null,
44     no_of_tickets   tinyint not null,
45     a_code          char(3) not null,
46     price_per_ticket decimal(5,2) not null,
47     brand           varchar(30) not null
48 );
49 • describe ticket_details;

```

Result Grid

Filter Rows:

Export: Wrap Cell Content: [IA](#)

Field	Type	Null	Key	Default	Extra
p_date	date	NO		NULL	
customer_id	tinyint	NO		NULL	
aircraft_id	varchar(20)	NO		NULL	
class_id	varchar(20)	NO		NULL	
no_of_tickets	tinyint	NO		NULL	
a_code	char(3)	NO		NULL	
price_per_ticket	decimal(5,2)	NO		NULL	
brand	varchar(30)	NO		NULL	

MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

air_cargo

Tables

customer

passengers_on_flights

routes

ticket_details

Views

Stored Procedures

Functions

sys

Administration Schemas

Information

No object selected

Query 1

Limit to 50000 rows

```

44     no_of_tickets   tinyint not null,
45     a_code          char(3) not null,
46     price_per_ticket decimal(5,2) not null,
47     brand           varchar(30) not null
48 );
49 • describe ticket_details;
50
51 • select * from ticket_details;
52

```

Result Grid

Filter Rows:

Export: Wrap Cell Content: [IA](#)

p_date	customer_id	aircraft_id	class_id	no_of_tickets	a_code	Price_per_ticket	brand
26-12-2018	27	767-301ER	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	767-301ER	First Class	1	AGB	390	Emirates
05-05-2020	5	ERJ142	Economy	1	CTM	120	Jet Airways
07-07-2020	7	767-301ER	Business	1	BFS	430	Emirates
08-08-2020	8	A321	Economy Plus	1	DAL	275	Qatar Airways
09-09-2020	9	767-301ER	First Class	1	BOH	380	Emirates
10-10-2020	10	A321	Economy	1	MCO	135	Jet Airways
11-11-2020	11	767-301ER	Business	1	AGB	465	Emirates
12-12-2020	19	CRJ900	Economy Plus	1	DEN	225	British Airways
01-01-2019	13	A321	First Class	1	YVR	395	Qatar Airways

- 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.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'air_cargo' database structure, including tables like 'customer', 'passengers_on_flights', 'routes', and 'ticket_details'. The 'Query 1' editor contains the following SQL code:

```
50
51 • select * from ticket_details;
52
53 • select * from passengers_on_flights where route_id between 1 and 25 order by route_id, customer_id;
54
```

The 'Result Grid' displays the results of the second query. The columns are: customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id, travel_date, and flight_num. The results show a list of passengers and their flight details for routes 1 through 25.

customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
18	767-301ER	1	EWB	HNL	13FC	First Class	01-04-2018	1111
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
4	767-301ER	5	LAX	JFK	02FC	First Class	06-04-2020	1115
11	767-301ER	5	LAX	JFK	04B	Business	12-11-2020	1115
46	A321	8	ORD	EWB	12FC	First Class	08-07-2011	1118
1	ERJ142	9	DEN	LAX	01EP	Economy Plus	26-12-2019	1119
29	ERJ142	9	DEN	LAX	11B	Business	03-05-2018	1119
10	A321	10	HNL	DEN	05E	Economy	11-10-2020	1120
5	767-301ER	767-301ER	ABI	ADK	02B	Business	02-07-2018	1122
13	A321	13	ADK	BQN	06FC	First Class	05-01-2019	1123
17	A321	13	ABI	ADK	04EP	Economy Plus	03-06-2019	1123
15	A321	14	BQN	CAK	06B	Business	02-11-2018	1124
24	A321	14	BQN	CAK	08B	Business	22-07-2019	1124
9	767-301ER	15	CAK	ANI	04FC	First Class	10-09-2020	1125
44	767-301ER	15	CAK	ANI	11FC	First Class	06-10-2020	1125

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

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'air_cargo' database structure. The 'Query 1' editor contains the following SQL code:

```
51 • select * from ticket_details;
52
53 • select * from passengers_on_flights where route_id between 1 and 25 order by route_id, customer_id;
54
55 • select count(*) as 'Number of Passengers', sum(no_of_tickets*price_per_ticket) as 'Total Revenue' from ticket_details where class_id = 'Business';
```

The 'Result Grid' displays the results of the third query. The columns are: Number of Passengers and Total Revenue. The results show 13 passengers and a total revenue of 6034.

Number of Passengers	Total Revenue
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.

The screenshot shows a database IDE with a Navigator pane on the left displaying the 'air_cargo' schema. The main query editor contains the following SQL code:

```
52
53 • select * from passengers_on_flights where route_id between 1 and 25 order by route_id, customer_id;
54
55 • select count(*) as 'Number of Passengers', sum(no_of_tickets*price_per_ticket) as 'Total Revenue' from ticket_details where class_id
56
57 • select concat(first_name, ' ',last_name) as "Full Name" from customer;
58
```

The Result Grid at the bottom displays the output of the third query, showing a list of full names:

Full Name
Julie Sam
Steve Ryan
Morris Lois
Catherina Emily
Aaron Kim
Alexander Scot
Anderson Stewart
Floyd Ted
Leo Travis
Melvin Tracy
Roger Watson
Shirley Wally
Solomon Walter
Carol Vernon
Linda William

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

The screenshot shows the same database IDE with a new query in the main editor:

```
52
53 • select * from passengers_on_flights where route_id between 1 and 25 order by route_id, customer_id;
54
55 • select count(*) as 'Number of Passengers', sum(no_of_tickets*price_per_ticket) as 'Total Revenue' from ticket_details where class_id
56
57 • select concat(first_name, ' ',last_name) as "Full Name" from customer;
58
59
60 • select * from customer c where exists (select 1 from ticket_details t where t.customer_id = c.customer_id) order by customer_id;
61 • select count(*) from customer c where c.customer_id in (select distinct customer_id from ticket_details);
62
63
```

The Result Grid at the bottom shows the output of the sixth query, which is a single row with the count of customers:

count(*)
33

- 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.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'air_cargo' schema with tables: customer, passengers_on_flights, routes, ticket_details, Views, Stored Procedures, and Functions. The 'sys' schema is also visible. The right pane shows a query window with the following SQL code:

```
61 select count(*) from customer c where c.customer_id in (select distinct customer_id from ticket_details);
62
63 select c.customer_id, c.first_name, c.last_name from customer c
64 where exists (select 1 from ticket_details t
65 where t.customer_id = c.customer_id and brand = 'Emirates') order by customer_id;
66
67
```

The 'Result Grid' shows the following data:

customer_id	first_name	last_name
2	Steve	Ryan
4	Cathenna	Emily
5	Aaron	Kim
7	Anderson	Stewart
9	Leo	Travis
11	Roger	Walson
14	Carol	Vernon
18	Gloria	Richie
19	Joyce	Paul
25	Moss	Morris
27	Cherly	Vernon
31	James	Robert
44	Billy	Brian
49	Russell	Peter

- 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.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'air_cargo' schema with tables: customer, passengers_on_flights, routes, ticket_details, Views, Stored Procedures, and Functions. The 'sys' schema is also visible. The right pane shows a query window with the following SQL code:

```
63 select c.customer_id, c.first_name, c.last_name from customer c
64 where exists (select 1 from ticket_details t
65 where t.customer_id = c.customer_id and brand = 'Emirates') order by customer_id;
66
67
68 select customer_id, count(customer_id) as "Number of Travels" from passengers_on_flights
69 where class_id = 'Economy Plus' group by customer_id having count(customer_id) >= 1;
70
71
```

The 'Result Grid' shows the following data:

customer_id	Number of Travels
1	1
8	1
11	1
17	1
19	2
22	1
32	1
47	1
50	1

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

QL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x EER Diagram x

Edit View Query Database Server Tools Scripting Help

ator:..... Query 1 x

Limit to 50000 rows

```
69 where class_id = 'Economy Plus' group by customer_id having count(customer_id) >= 1;
70
71 • select if(sum(no_of_tickets * price_per_ticket) > 10000, 'YES, Revenue crossed 10000', 'NO, Revenue has not crossed 10000')
72 as Result from ticket_details;
73
74
75
76
```

Result Grid

Result
YES, Revenue crossed 10000

object selected

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

Navigator:..... Schemas

Filter objects

air_cargo

- Tables
 - customer
 - passengers_on_flights
 - routes
 - ticket_details
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas

Information:.....

No object selected

Query 1 x

Limit to 50000 rows

```
62
63 • select c.customer_id, c.first_name, c.last_name from customer c
64 where exists (select 1 from ticket_details t
65 where t.customer_id = c.customer_id and brand = 'Emirates') order by customer_id;
66
67
68 • select customer_id, count(customer_id) as "Number of Travels" from passengers_on_flights
69 where class_id = 'Economy Plus' group by customer_id having count(customer_id) >= 1;
70
71 • select if(sum(no_of_tickets * price_per_ticket) > 10000, 'YES, Revenue crossed 10000', 'NO, Revenue has not crossed 10000')
72 as Result from ticket_details;
73
74
75 • create user if not exists 'air_cargouser'@'127.0.0.1' identified by 'password123';
76 • grant all privileges on air_cargo to 'air_cargouser'@'127.0.0.1';
77
78
```

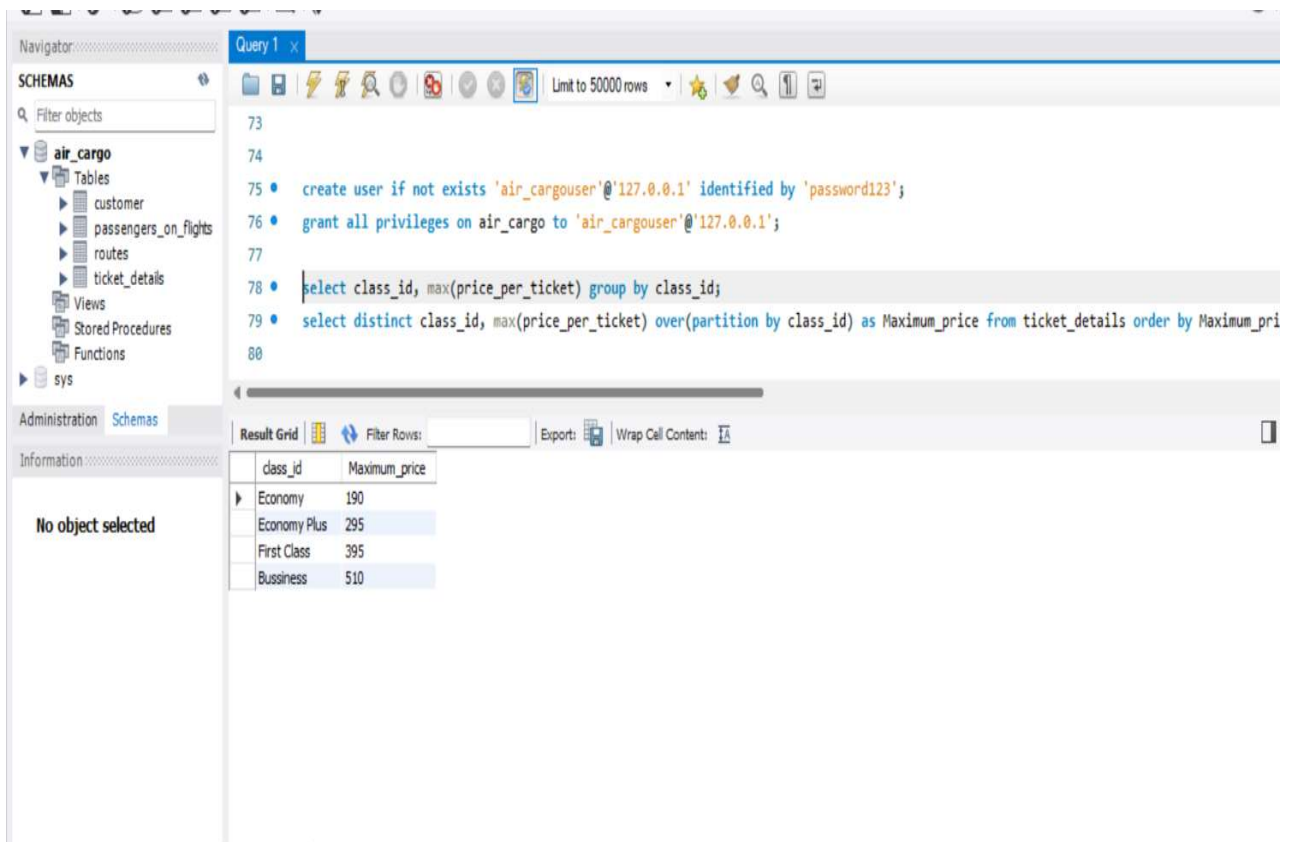
Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	15:01:33	create user if not exists 'air_cargouser'@'127.0.0.1' identified by 'password123'	0 row(s) affected	0.078 sec
3	15:01:39	grant all privileges on air_cargo to 'air_cargouser'@'127.0.0.1'	0 row(s) affected	0.031 sec

Object Info Session

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



The screenshot shows a database IDE with a SQL query editor and a results grid. The query editor contains the following SQL code:

```

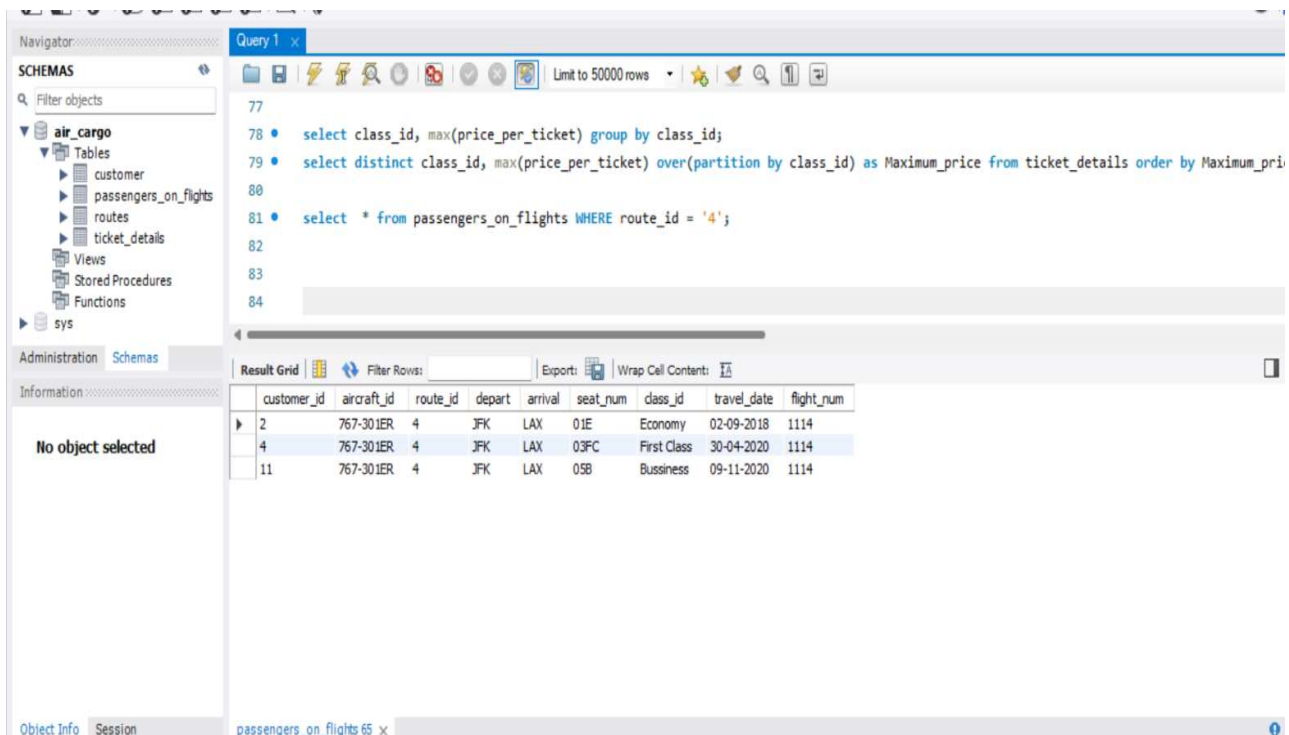
73
74
75 • create user if not exists 'air_cargouser'@'127.0.0.1' identified by 'password123';
76 • grant all privileges on air_cargo to 'air_cargouser'@'127.0.0.1';
77
78 • select class_id, max(price_per_ticket) group by class_id;
79 • select distinct class_id, max(price_per_ticket) over(partition by class_id) as Maximum_price from ticket_details order by Maximum_pri
80

```

The results grid shows the following data:

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

12) Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers_on_flights table.



The screenshot shows a database IDE with a SQL query editor and a results grid. The query editor contains the following SQL code:

```

77
78 • select class_id, max(price_per_ticket) group by class_id;
79 • select distinct class_id, max(price_per_ticket) over(partition by class_id) as Maximum_price from ticket_details order by Maximum_pri
80
81 • select * from passengers_on_flights WHERE route_id = '4';
82
83
84

```

The results grid shows the following data:

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	Bussiness	09-11-2020	1114

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'air_cargo' database, including tables like 'customer', 'passengers_on_flights', 'routes', and 'ticket_details'. The main editor shows a query with the following SQL statements:

```

77
78 • select class_id, max(price_per_ticket) group by class_id;
79 • select distinct class_id, max(price_per_ticket) over(partition by class_id) as Maximum_price from ticket_details order by Maximum_pri
80
81 • select * from passengers_on_flights WHERE route_id = '4';
82
83 • create index route_idx on passengers_on_flights (route_id);
84 • select * from passengers_on_flights where route_id = '4';
  
```

The 'Result Grid' at the bottom displays the results of the query. The columns are: customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id, travel_date, and flight_num. The results are as follows:

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

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'air_cargo' database. The main editor shows a query with the following SQL statements:

```

84 • select * from passengers_on_flights where route_id = '4';
85
86
87 • select customer_id, aircraft_id, sum(no_of_tickets * price_per_ticket) as Total_Price
88 • from ticket_details group by customer_id, aircraft_id with rollup;
89
  
```

The 'Result Grid' at the bottom displays the results of the query. The columns are: customer_id, aircraft_id, and Total_Price. The results are as follows:

customer_id	aircraft_id	Total_Price
1	CRJ900	320
1	ERJ142	250
1	NULL	570
2	767-301ER	130
2	A321	505
2	NULL	635
4	767-301ER	780
4	NULL	780
5	767-301ER	430
5	ERJ142	240
5	NULL	670
7	767-301ER	430
7	NULL	430
8	A321	465
8	NULL	465
9	767-301ER	380
9	CRJ900	390

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

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'air_cargo' database selected. The main query editor contains the following SQL code:

```

88 from ticket_details group by customer_id, aircraft_id with rollup;
89
90 • create view Business_classes as
91 select c. first_name, c. last_name , t. brand from customer c
92 inner join ticket_details t on c. customer_id = t. customer_id where class_id = "Business" order by customer_id ;
93 • SELECT * FROM Business_classes;
94
95

```

The 'Result Grid' shows the data for the 'Business_classes' view:

first_name	last_name	brand
Steve	Ryan	Qatar Airways
Aaron	Kim	Emirates
Anderson	Stewart	Emirates
Roger	Walson	Emirates
Roger	Walson	Emirates
Linda	William	Qatar Airways
Christy	Josh	British Airways
Calvin	Willis	Qatar Airways
Moss	Morris	Emirates
Watson	Ronald	Jet Airways
Watson	Ronald	Qatar Airways
Mark	Ethan	British Airways
Russell	Peter	Emirates

The status bar at the bottom indicates 'Query Completed'.

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.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'air_cargo' database selected. The main query editor contains the following SQL code:

```

95
96 delimiter $$
97 • create procedure check_routes(in rid VARCHAR (255))
98 begin
99 declare TableNotFound condition for 1146;
100 declare exit handler for TableNotFound
101 select 'Please check if table customer / route_id are created - one/both are missing' Message;
102 set @query = concat('select * from customer where customer_id in (select distinct customer_id from passengers_on_flights
103 where route_id in (' , rid, '));');
104 prepare sql_query from @query;
105 execute sql_query;
106 end$$
107 delimiter ;
108 • call check_routes("1,5");
109

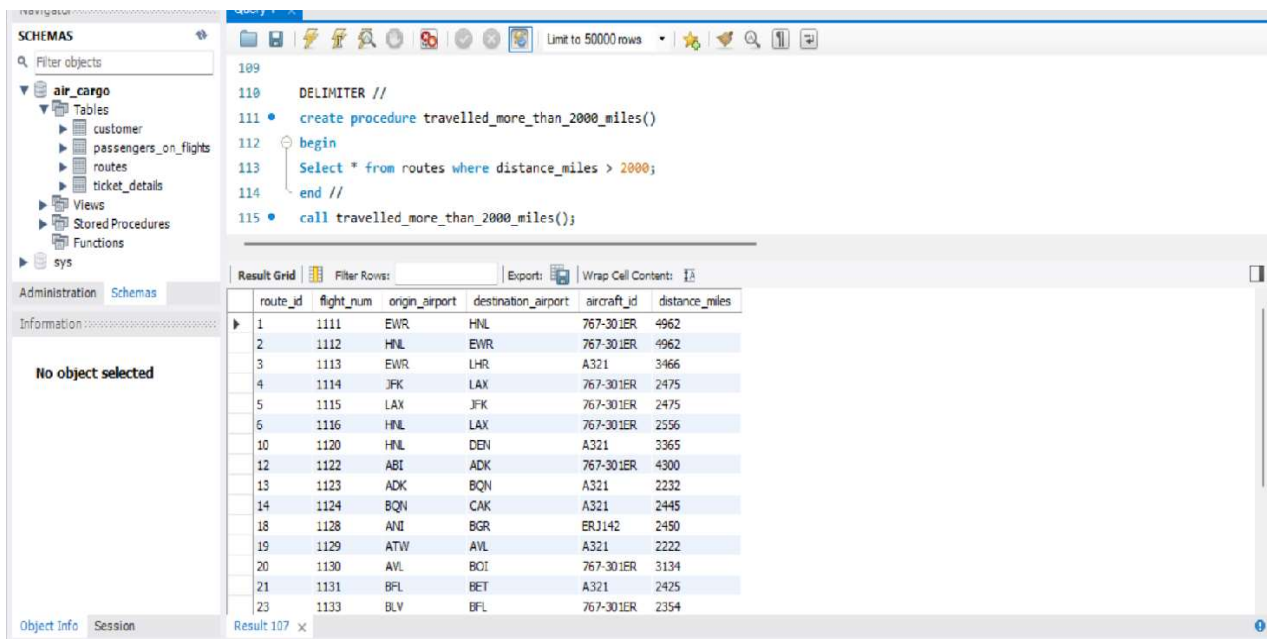
```

The 'Result Grid' shows the data for the 'check_routes' stored procedure:

customer_id	first_name	last_name	date_of_birth	gender
4	Cathenna	Emily	14-09-1977	F
11	Roger	Walson	24-05-1996	M
18	Gloria	Richie	04-12-1989	F

The status bar at the bottom indicates 'Result 106'.

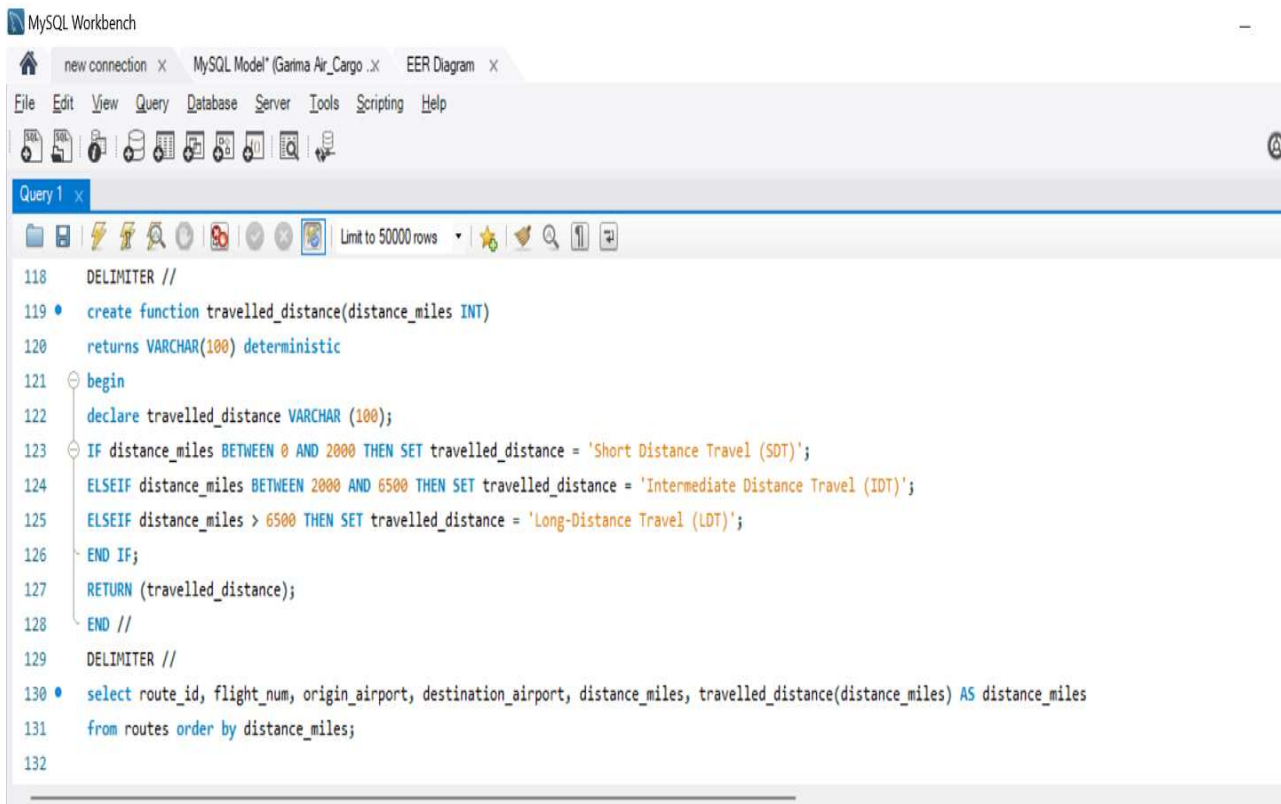
17) Write a query to create a stored procedure that extracts all the details from the routes table where the travelled distance is more than 2000 miles.



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'air_cargo' database with tables like 'customer', 'passengers_on_flights', 'routes', and 'ticket_details'. The main editor shows a SQL query to create a stored procedure named 'travelled_more_than_2000_miles()' that selects all details from the 'routes' table where 'distance_miles' is greater than 2000. Below the editor, the 'Result Grid' displays the results of the procedure call, showing a list of routes with columns: route_id, flight_num, origin_airport, destination_airport, aircraft_id, and distance_miles.

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
1	1111	EWB	HNL	767-301ER	4962
2	1112	HNL	EWB	767-301ER	4962
3	1113	EWB	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	ABT	ADK	767-301ER	4300
13	1123	ADK	BQN	A321	2232
14	1124	BQN	CAK	A321	2445
18	1128	ANI	BGR	ERJ142	2450
19	1129	ATV	AVL	A321	2222
20	1130	AVL	BOI	767-301ER	3134
21	1131	BFL	BET	A321	2425
23	1133	BLV	BFL	767-301ER	2354

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 .



The screenshot shows the MySQL Workbench interface with a query editor. The query is a stored procedure named 'travelled_distance' that takes 'distance_miles' as an input and returns a VARCHAR(100) value. It uses conditional logic to categorize the distance into 'Short Distance Travel (SDT)', 'Intermediate Distance Travel (IDT)', or 'Long-Distance Travel (LDT)'. The procedure is followed by a SELECT statement that retrieves all columns from the 'routes' table, including the calculated 'travelled_distance'.

```

118 DELIMITER //
119 create function travelled_distance(distance_miles INT)
120 returns VARCHAR(100) deterministic
121 begin
122 declare travelled_distance VARCHAR (100);
123 IF distance_miles BETWEEN 0 AND 2000 THEN SET travelled_distance = 'Short Distance Travel (SDT)';
124 ELSEIF distance_miles BETWEEN 2000 AND 6500 THEN SET travelled_distance = 'Intermediate Distance Travel (IDT)';
125 ELSEIF distance_miles > 6500 THEN SET travelled_distance = 'Long-Distance Travel (LDT)';
126 END IF;
127 RETURN (travelled_distance);
128 END //
129 DELIMITER //
130 select route_id, flight_num, origin_airport, destination_airport, distance_miles, travelled_distance(distance_miles) AS distance_miles
131 from routes order by distance_miles;
132

```

route_id	flight_num	origin_airport	destination_airport	distance_miles	distance_miles
1	1111	EWV	HNL	4962	Intermediate Distance Travel (IDT)
2	1112	HNL	EWV	4962	Intermediate Distance Travel (IDT)
3	1113	EWV	LHR	3466	Intermediate Distance Travel (IDT)
4	1114	JFK	LAX	2475	Intermediate Distance Travel (IDT)
5	1115	LAX	JFK	2475	Intermediate Distance Travel (IDT)
6	1116	HNL	LAX	2556	Intermediate Distance Travel (IDT)
14	1124	BQN	CAK	2445	Intermediate Distance Travel (IDT)
23	1133	BLV	BFL	2354	Intermediate Distance Travel (IDT)
34	1144	CRW	COO	2452	Intermediate Distance Travel (IDT)
10	1120	HNL	DEN	3365	Intermediate Distance Travel (IDT)
12	1122	ABI	ADK	4300	Intermediate Distance Travel (IDT)
13	1123	ADK	BQN	2232	Intermediate Distance Travel (IDT)
25	1135	RDM	BJI	2425	Intermediate Distance Travel (IDT)
49	1159	DEC	ABI	4533	Intermediate Distance Travel (IDT)
48	1158	SCC	DEN	5645	Intermediate Distance Travel (IDT)
50	1160	DRT	ORD	2445	Intermediate Distance Travel (IDT)
18	1128	ANI	BGR	2450	Intermediate Distance Travel (IDT)
45	1120	ATM	BJI	2222	Intermediate Distance Travel (IDT)

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*

SCHEMAS
Filter objects
air_cargo
Tables
customer
passengers_on_flights
routes
ticket_details
Views
Stored Procedures
Functions
sys
Administration
Schemas
Information
No object selected

132
133
134
135
136
137
138
139

```

select p_date, customer_id, class_id,
case when class_id in ('Business', 'Economy Plus') then 'Yes'
else "No"
end as complimentary_service from ticket_details;

```

Result Grid
Filter Rows: Export: Wrap Cell Content:

p_date	customer_id	class_id	complimentary_service
26-12-2018	27	Economy	No
02-02-2020	22	Economy Plus	Yes
03-03-2020	21	Business	No
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Business	No
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	No
12-12-2020	19	Economy Plus	Yes
01-01-2019	13	First Class	No
02-02-2019	14	Economy	No
03-03-2019	25	Business	No
04-04-2019	16	First Class	No

Query Completed
20°C
AQI 142
Search
ENG
IN
07-12-2022

MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x) EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Query 1 x

```
140 • create function chk_comp_ser(cls varchar(20))
141     returns char(3)
142     deterministic
143     begin
144     declare comp_ser CHAR(3);
145     if cls in('Bussiness', 'Economy Plus') THEN set comp_ser = 'Yes';
146     else set comp_ser = 'No';
147     end if;
148     return (comp_ser);
149     end $$
150 • create procedure chk_comp_ser_pro()
151     begin
152     select p_date, customer_id, class_id, chk_comp_ser(class_id) as complimentary_service from ticket_details;
153     end $$
154 • call chk_comp_ser_pro();
```

MySQL Workbench

new connection x MySQL Model* (Garima Air_Cargo .x) EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

air_cargo

- Tables
 - customer
 - passengers_on_flights
 - routes
 - ticket_details
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas

Information

No object selected

Query 1 x

```
140 • create function chk_comp_ser(cls varchar(20))
141     returns char(3)
142     deterministic
143     begin
144     declare comp_ser CHAR(3);
```

Result Grid

p_date	customer_id	class_id	complimentary_service
26-12-2018	27	Economy	No
02-02-2020	22	Economy Plus	Yes
03-03-2020	21	Bussiness	Yes
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Bussiness	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	Bussiness	Yes
12-12-2020	19	Economy Plus	Yes
01-01-2019	13	First Class	No
02-02-2019	14	Economy	No
03-03-2019	25	Bussiness	Yes
04-04-2019	16	First Class	No
03-05-2019	17	Economy Plus	Yes
06-06-2019	18	Economy	No
07-07-2019	24	Bussiness	Yes

Object Info Session Result 114 x

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 the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'air_cargo' database with tables like 'customer', 'passengers_on_flights', 'routes', and 'ticket_details'. The main editor shows a SQL script for a stored procedure named 'my_cursor'. The script uses a cursor to iterate through the 'customer' table, specifically filtering for last names ending in 'Scott'. The 'Result Grid' at the bottom shows the output of the procedure, which is a single row with the last name 'Scott' and the first name 'Samuel'.

```
157 DELIMITER //
158 create Procedure my_cursor ()
159 begin
160 declare a VARCHAR (100);
161 declare b VARCHAR (100);
162 declare my_cursor cursor for select last_name, first_name from customer
163 where last_name = 'Scott';
164 open my_cursor;
165 repeat fetch my_cursor into a,b;
166 until b = 0 end repeat;
167 select a as last_name, b as first_name;
168 close my_cursor;
169 end;
170 // DELIMITER ;
171 CALL my_cursor();
```

last_name	first_name
Scott	Samuel