Course-End Project: Air Cargo Analysis

Problem Statement Scenario:

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

Project Objective:

You, as a DBA expert, need to focus on identifying the regular customers to provide offers, analyze the busiest route which helps to increase the number of aircraft required and prepare an analysis to determine the ticket sales details. This will ensure that the company improves its operability and becomes more customer-centric and a favorable choice for air travel.

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

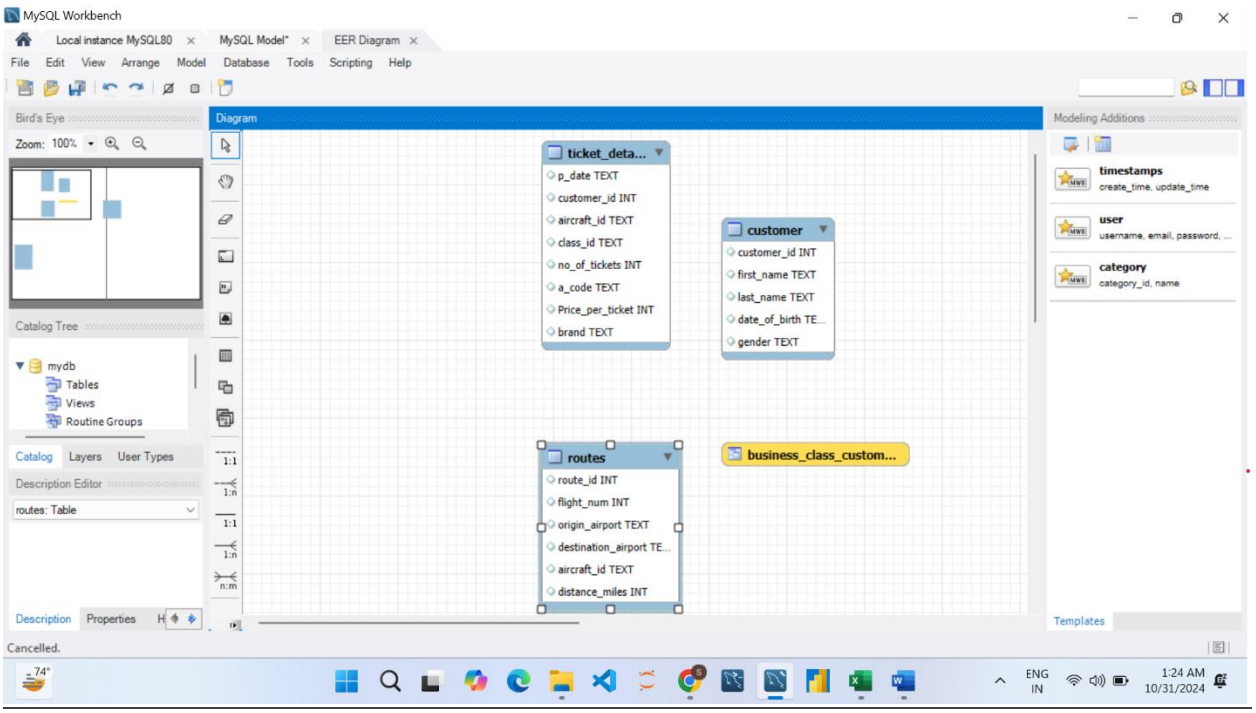
- Route_id – Route ID of from and to location
- Flight_num – Specific fight number for each route
- Origin_airport – Departure location
- Destination_airport – Arrival location
- Aircraft_id – ID of each aircraft in a brand
- Distance_miles – Distance between departure and arrival location

Following operations should be performed:

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

   - **Sql code**
     reverse engineer of air_cargo_db database

output

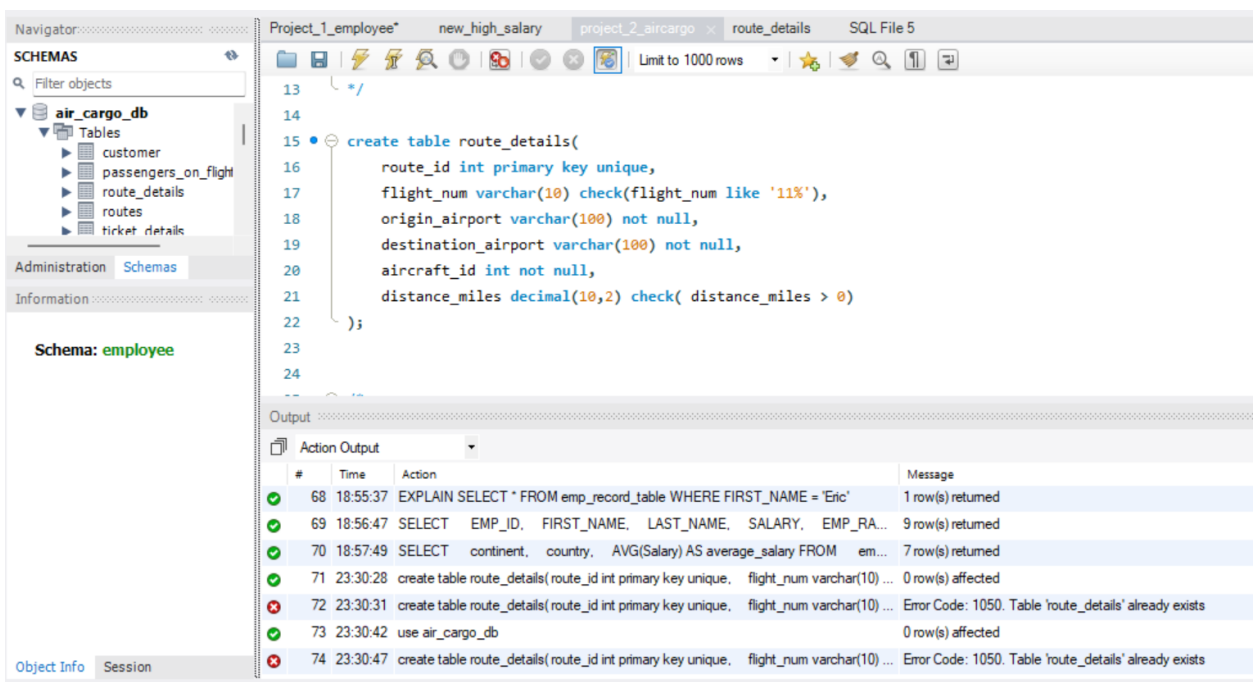


2. Write a query to create a route_details table using suitable data types for the fields, such as route_id, flight_num, origin_airport, destination_airport, aircraft_id, and distance_miles. Implement the check constraint for the flight number and unique constraint for the route_id fields. Also, make sure that the distance miles field is greater than 0.

- **Sql code**

```
create table route_details(
        route_id int primary key unique,
   flight_num varchar(10) check(flight_num like '11%'),
   origin_airport varchar(100) not null,
   destination_airport varchar(100) not null,
   aircraft_id int not null,
   distance_miles decimal(10,2) check( distance_miles > 0)
);
```

Output



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.

   - **Sql code**

```
select *
from passengers_on_flights
where route_id between 1 and 25;
```

output

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

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
1	ERJ142	9	DEN	LAX	01EP	Economy Plus	26-12-2019	1119
5	767-301ER	12	ABI	ADK	02B	Bussiness	02-07-2018	1122
5	ERJ142	18	ANI	BGR	02E	Economy	06-05-2020	1128
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
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
17	A321	13	ABI	ADK	04EP	Economy Plus	03-06-2019	1123
9	767-301ER	15	CAK	ANI	04FC	First Class	10-09-2020	1125
11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114
10	A321	10	HNL	DEN	05E	Economy	11-10-2020	1120
15	A321	14	BQN	CAK	06B	Bussiness	02-11-2018	1124
13	A321	13	ADK	BQN	06FC	First Class	05-01-2019	1123
22	ERJ142	22	BGR	BJI	07EP	Economy Plus	09-02-2020	1132

rs_on_flights 1

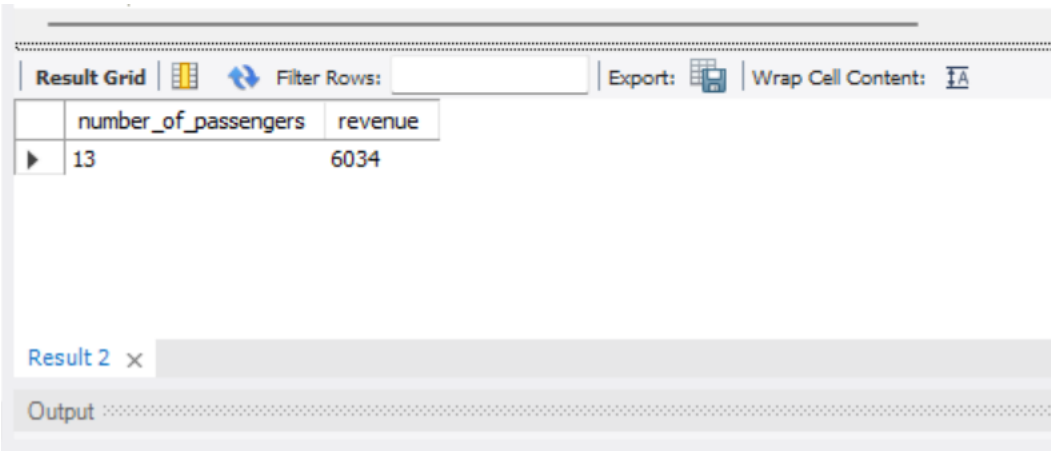
Output

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

   - **Sql code**

```
select count(*) as number_of_passengers,
sum(Price_per_ticket) as revenue
from ticket_details
where class_id = "Bussiness";
```

output

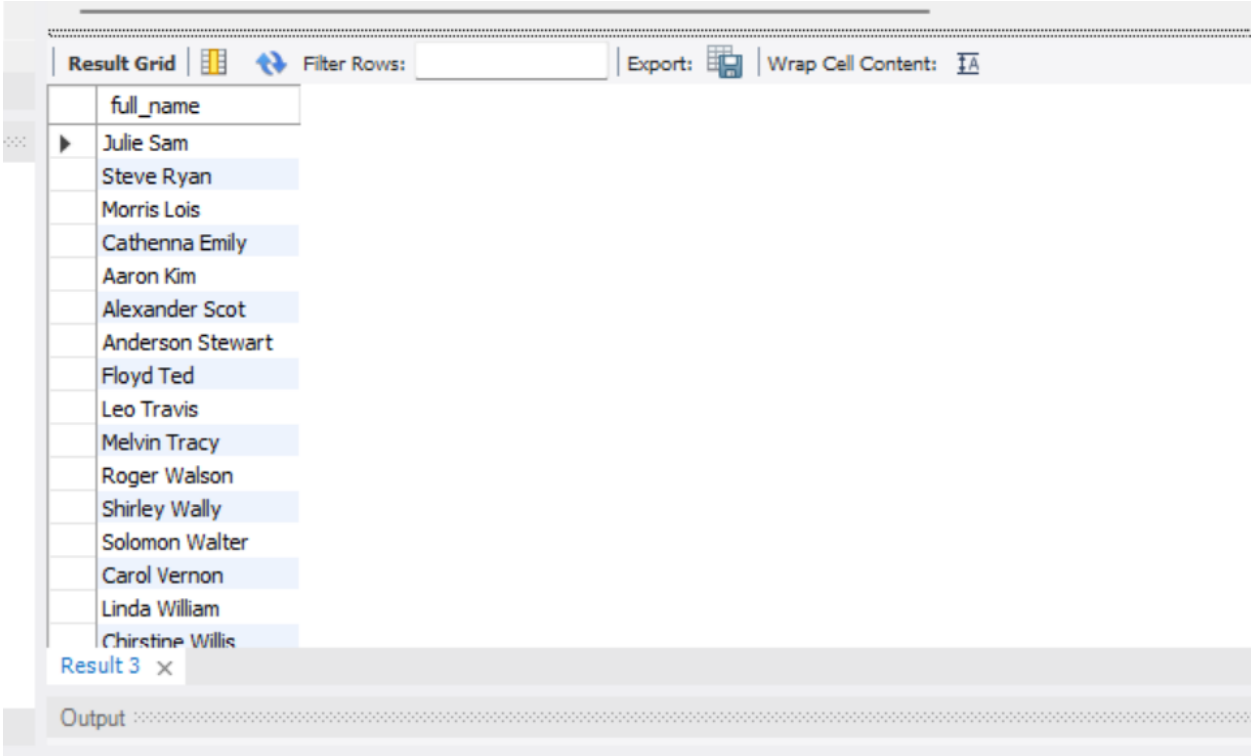


5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.

   - **Sql code**

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

output



6. 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 customer.*
from customer
join ticket_details on customer.customer_id =
ticket_details.customer_id;
```

output

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

customer_id	first_name	last_name	date_of_birth	gender
27	Cherly	Vernon	19-03-1992	F
22	Pheny	Eri	29-01-1999	M
21	Chirsty	Josh	10-01-2004	M
4	Cathenna	Emily	14-09-1977	F
5	Aaron	Kim	18-02-1991	M
7	Anderson	Stewart	11-01-1992	M
8	Floyd	Ted	21-02-1993	M
9	Leo	Travis	22-03-1994	M
10	Melvin	Tracy	23-04-1995	M
11	Roger	Walson	24-05-1996	M
19	Joyce	Paul	02-06-1990	F
13	Solomon	Walter	26-07-1998	M
14	Carol	Vernon	27-08-1999	F
25	Moss	Morris	18-02-2011	M
16	Chirstine	Chirstine	06-10-1987	F
17	Catherine	[illegible]	09-11-1988	F

Result 4

Output

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.

   - **Sql code**

```
select customer.first_name,customer.last_name
from customer
join ticket_details on customer.customer_id =
ticket_details.customer_id
where brand = 'Emirates';
```

output

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

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

Result 5

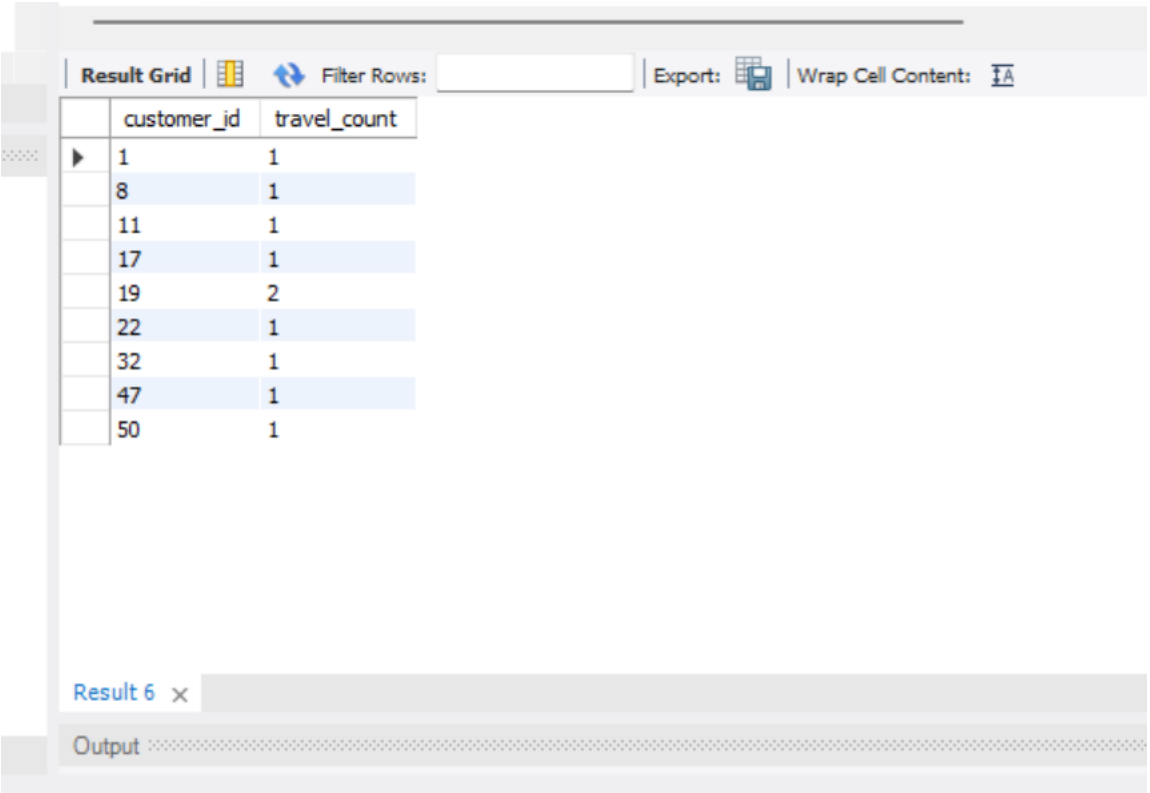
Output

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.

   - **Sql code**

```
select customer_id, count(*) as travel_count
from passengers_on_flights
where class_id = 'Economy Plus'
group by customer_id
having count(*) > 0;
```

output



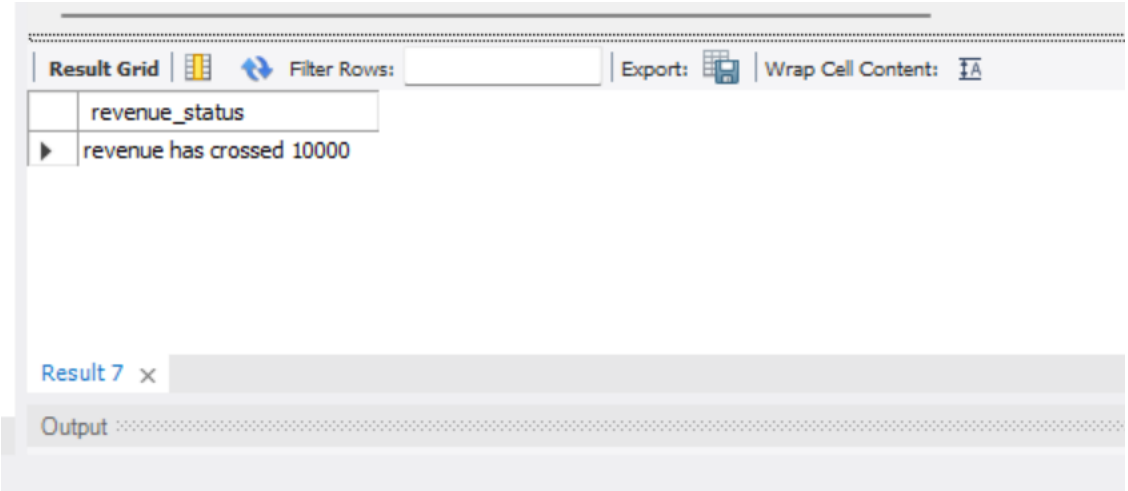
customer_id	travel_count
1	1
8	1
11	1
17	1
19	2
22	1
32	1
47	1
50	1

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

   - **Sql code**

```
select
        case
                when sum(Price_per_ticket) > 10000 then "revenue has
crossed 10000"
                else "revenue has not crossed 10000"
   end as revenue_status
from ticket_details;
```

output

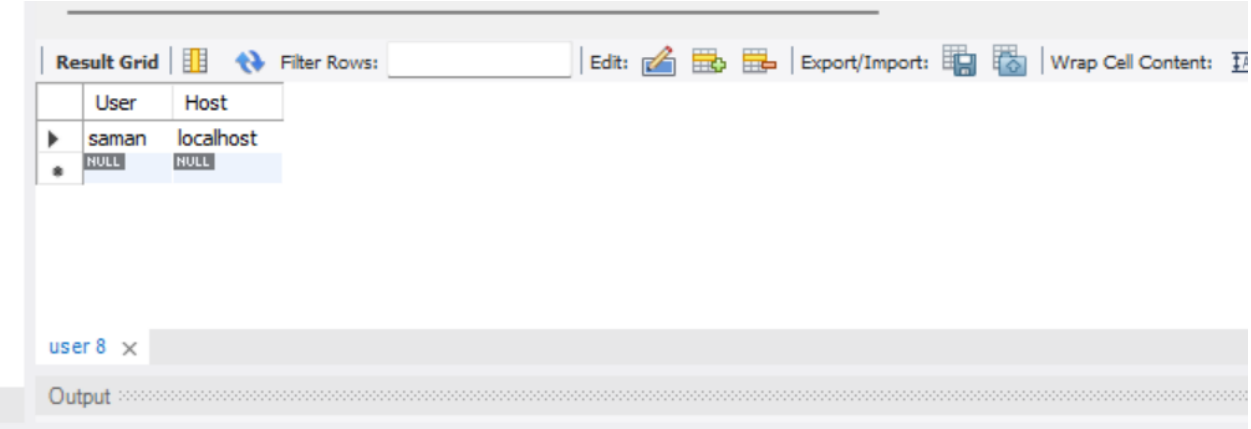


10. Write a query to create and grant access to a new user to perform operations on a database.

- **Sql code**

```
CREATE USER 'saman'@'localhost' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON air_cargo_db.* TO 'saman'@'localhost';

SELECT User, Host FROM mysql.user WHERE User = 'saman';
```

Output



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

- **Sql code**

```
SELECT
    class_id,
    Price_per_ticket,
    MAX(Price_per_ticket) OVER (PARTITION BY class_id) AS
max_ticket_price
FROM
    ticket_details;
```

output

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

class_id	Price_per_ticket	max_ticket_price
Bussiness	499	510
Bussiness	430	510
Bussiness	490	510
Bussiness	490	510
Bussiness	510	510
Bussiness	430	510
Bussiness	480	510
Bussiness	430	510
Bussiness	505	510
Bussiness	465	510
Bussiness	410	510
Bussiness	430	510
Bussiness	465	510
Economy	130	190
Economy	130	190
Economy	170	190

510

Result 9

Output

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.

- **Sql code**

```
SELECT
    class_id,
    Price_per_ticket,
    MAX(Price_per_ticket) OVER (PARTITION BY class_id) AS
max_ticket_price
FROM
    ticket_details;
```

output

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

class_id	Price_per_ticket	max_ticket_price
Bussiness	499	510
Bussiness	430	510
Bussiness	490	510
Bussiness	490	510
Bussiness	510	510
Bussiness	430	510
Bussiness	480	510
Bussiness	430	510
Bussiness	505	510
Bussiness	465	510
Bussiness	410	510
Bussiness	430	510
Bussiness	465	510
Economy	130	190
Economy	130	190
Economy	170	190

Result 10

Output

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

- **Sql code**

```
SELECT *
FROM passengers_on_flights
WHERE route_id = 4;
CREATE INDEX idx_route_id ON passengers_on_flights(route_id);
```

output

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

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

Bussiness

passengers_on_flights 11

Output

14. 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(Price_per_ticket) as total_price
from
    ticket_details
group by
    customer_id, aircraft_id
with rollup;
```

output

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

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

Result 12

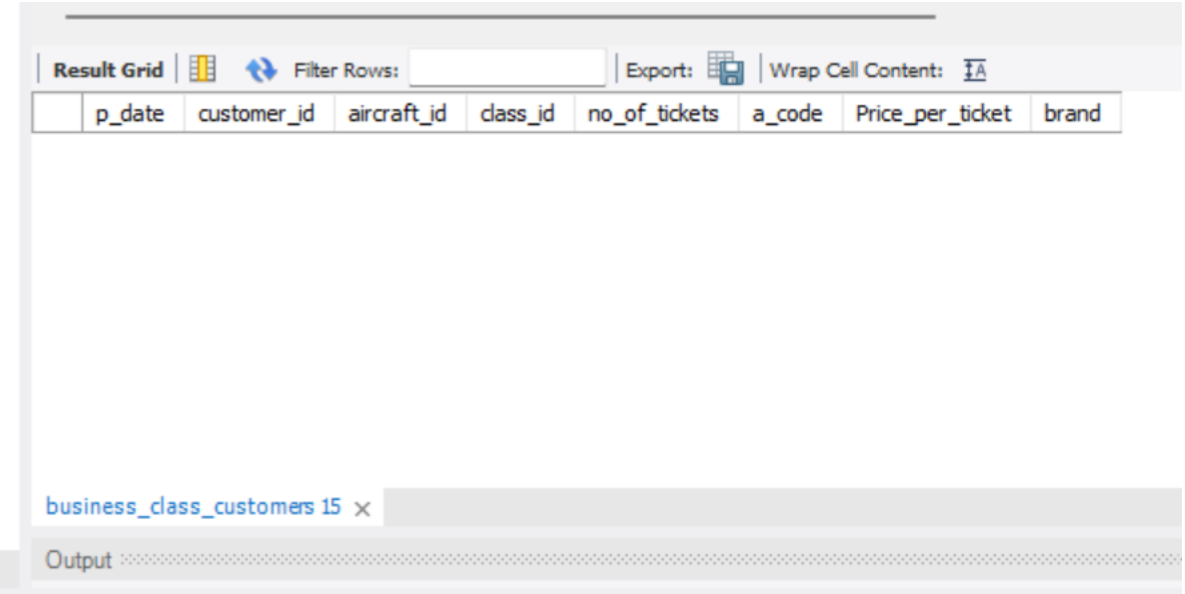
Output

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

- **Sql code**

```
create view business_class_customers as
select *
from ticket_details
where brand = 'Business';

select * from business_class_customers;
```

output



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.

- **Sql code**

```
delimiter //
create procedure passengers_flying(
        in start_rout_id int,
   in end_rout_id int)
 begin
        select *
   from passengers_on_Flights
   where route_id between start_rout_id and end_rout_id;
end //
delimiter ;
CALL passengers_flying(1,10);
```

Output

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

customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
18	767-301ER	1	EWR	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	Bussiness	09-11-2020	1114
4	767-301ER	5	LAX	JFX	02FC	First Class	06-04-2020	1115
11	767-301ER	5	LAX	JFX	04B	Bussiness	12-11-2020	1115
46	A321	8	ORD	EWR	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	Bussiness	03-05-2018	1119
10	A321	10	HNL	DEN	05E	Economy	11-10-2020	1120

Result 16

Output

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.

- **<u>Sql code</u>**

```
delimiter //
create procedure getLongDistanceMiles()
begin
   select *
   from routes
   where distance_miles > 2000;
end //
delimiter ;

call getLongDistanceMiles();
```

<u>output</u>

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
20	1130	AVL	BOI	767-301ER	3134
21	1131	BFL	BET	A321	2425
23	1133	BLV	BFL	767-301ER	2354
25	1135	BDM	BIT	A321	2425

Result 17

Output

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.

- **Sql code**

```
DELIMITER //

CREATE PROCEDURE GroupFlightDistances()
BEGIN
   SELECT
      route_id,
      flight_num,
      distance_miles,
      CASE
         WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN
'SDT'
         WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN
'IDT'
         WHEN distance_miles > 6500 THEN 'LDT'
         ELSE 'Unknown'
      END AS travel_category
   FROM
      routes;
END //

DELIMITER ;

 CALL GroupFlightDistances();
```

Output

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

route_id	flight_num	distance_miles	travel_category
1	1111	4962	IDT
2	1112	4962	IDT
3	1113	3466	IDT
4	1114	2475	IDT
5	1115	2475	IDT
6	1116	2556	IDT
7	1117	1745	SDT
8	1118	719	SDT
9	1119	862	SDT
10	1120	3365	IDT
12	1122	4300	IDT
13	1123	2232	IDT
14	1124	2445	IDT
15	1125	2000	SDT
16	1126	1700	SDT
17	1127	1900	SDT

IDT

Result 18

Output

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*

    - **<u>Sql code</u>**

```
DELIMITER //

CREATE FUNCTION GetComplimentaryServices1(class_id VARCHAR(50))
RETURNS VARCHAR(3)
DETERMINISTIC  -- Indicate that the function is deterministic
BEGIN
   DECLARE services VARCHAR(3);

   IF class_id IN ('Business', 'Economy Plus') THEN
      SET services = 'Yes';
   ELSE
      SET services = 'No';
   END IF;

   RETURN services;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE GetTicketDetailsWithServices1()
BEGIN

   SELECT
      p_date,
```

```
        customer_id,
        class_id,
        GetComplimentaryServices1(class_id) AS complimentary_services
    FROM
        ticket_details;
END //
DELIMITER ;


CALL GetTicketDetailsWithServices1();
```

Output

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

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	Bussiness	No
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Bussiness	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	Bussiness	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	Bussiness	No
04-04-2019	16	First Class	No
03-05-2019	17	Economy Plus	Yes

Result 19

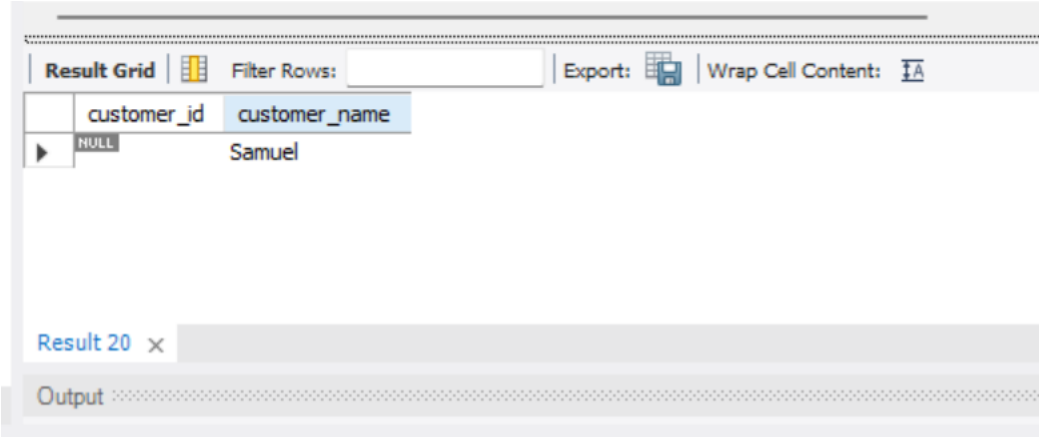
Output

20. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

- **Sql code**

```
DELIMITER //
CREATE PROCEDURE GetFirstCustomerWithLastNameScott1()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE customer_id INT;
   DECLARE customer_name VARCHAR(255);
   DECLARE cur CURSOR FOR
      SELECT customer_id, first_name
      FROM customer
      WHERE last_name LIKE '%Scott';

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cur;
   FETCH cur INTO customer_id, customer_name;

   IF NOT done THEN
      SELECT customer_id, customer_name;
   ELSE
      SELECT 'No customer found' AS message;
   END IF;

   CLOSE cur;
END //
DELIMITER ;


CALL GetFirstCustomerWithLastNameScott1();
```

Output


Result Grid
Filter Rows:
Export:
Wrap Cell Content:
customer_id
customer_name
NULL
Samuel
Result 20
Output