SQL Assessment Project -2 submitted by Sujit Sonar:

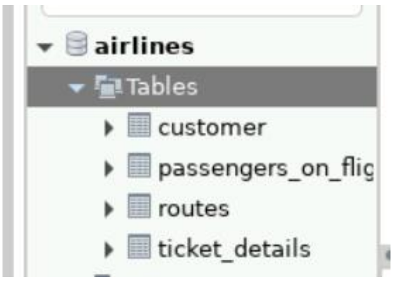
Air Cargo Analysis.

<u>Following operations should be performed:</u>

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

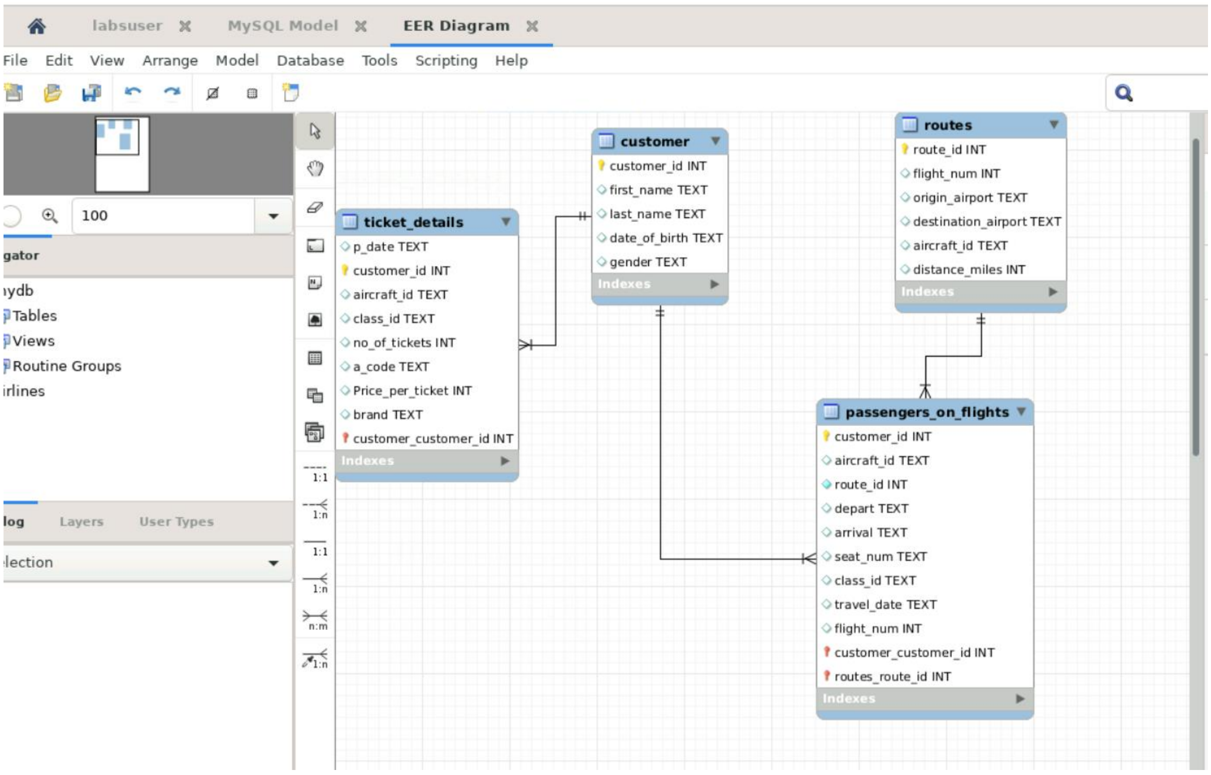
   Created database airlines using command "create database airlines"

   Loaded csv files into the lab environment and imported csv files to create the below tables using "Table data import wizard".



airlines
- Tables
  - customer
  - passengers_on_flig
  - routes
  - ticket_details

   Since the tables are loaded without any primary key, altering tables to create primary key for each table and not null using the MySQL Model



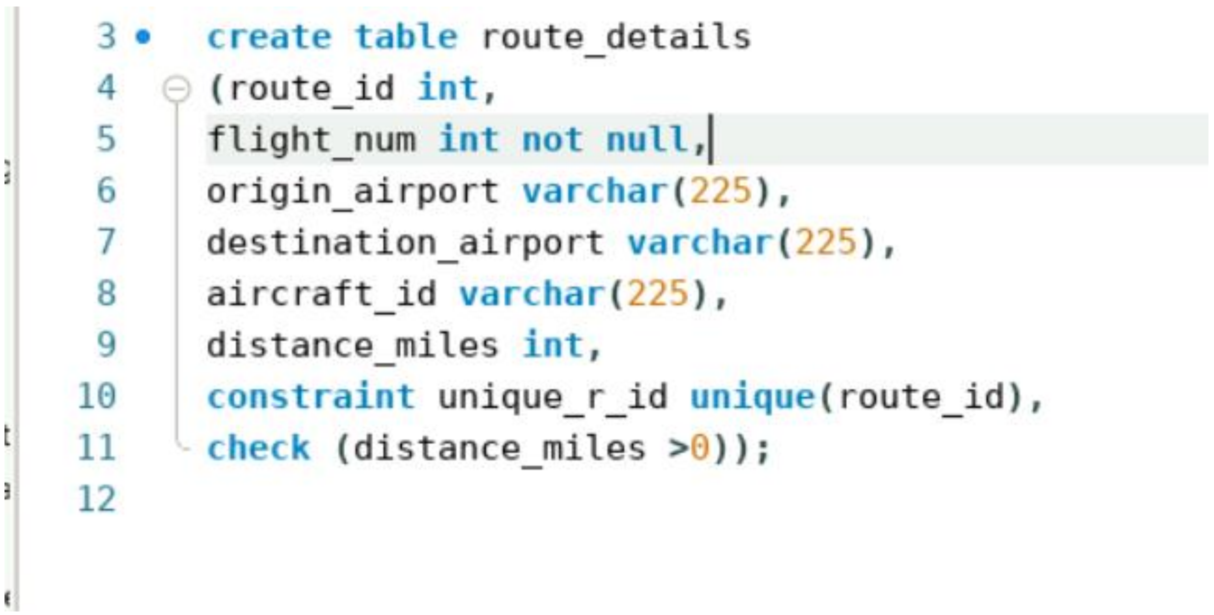
customer
- customer_id INT
- first_name TEXT
- last_name TEXT
- date_of_birth TEXT
- gender TEXT

routes
- route_id INT
- flight_num INT
- origin_airport TEXT
- destination_airport TEXT
- aircraft_id TEXT
- distance_miles INT

ticket_details
- p_date TEXT
- customer_id INT
- aircraft_id TEXT
- class_id TEXT
- no_of_tickets INT
- a_code TEXT
- Price_per_ticket INT
- brand TEXT
- customer_customer_id INT

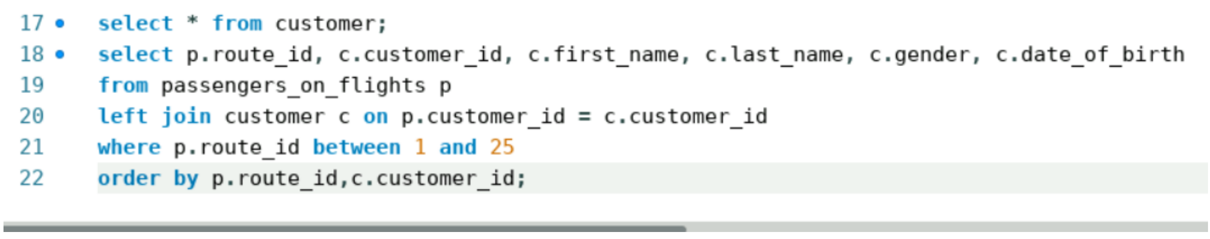
passengers_on_flights
- customer_id INT
- aircraft_id TEXT
- route_id INT
- depart TEXT
- arrival TEXT
- seat_num TEXT
- class_id TEXT
- travel_date TEXT
- flight_num INT
- customer_customer_id INT
- routes_route_id INT

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.

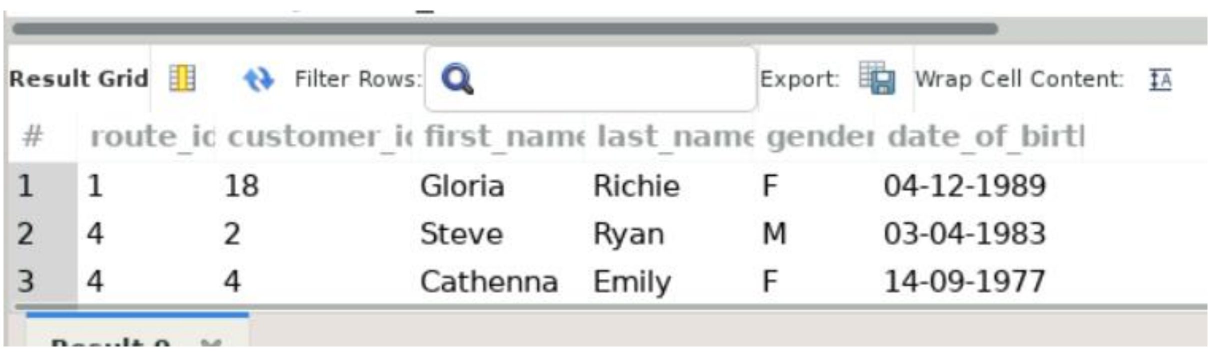


```
create table route_details
(route_id int,
flight_num int not null,
origin_airport varchar(225),
destination_airport varchar(225),
aircraft_id varchar(225),
distance_miles int,
constraint unique_r_id unique(route_id),
check (distance_miles >0));
```

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.

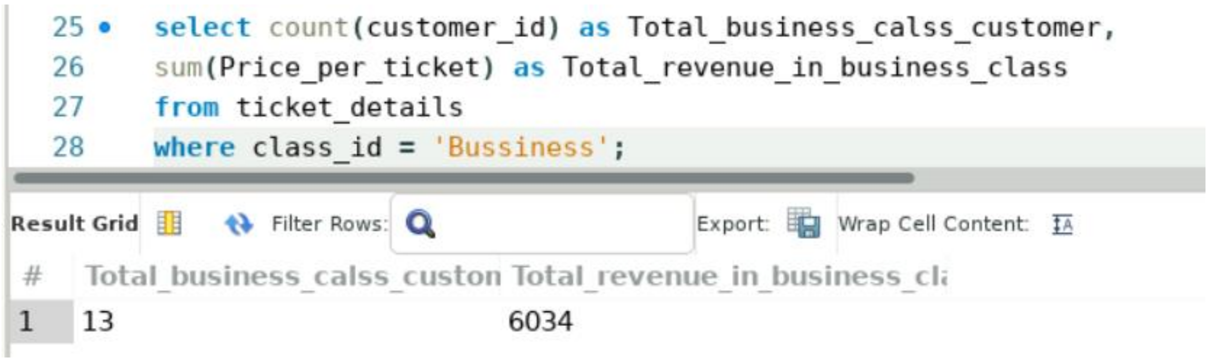


```
select * from customer;
select p.route_id, c.customer_id, c.first_name, c.last_name, c.gender, c.date_of_birth
from passengers_on_flights p
left join customer c on p.customer_id = c.customer_id
where p.route_id between 1 and 25
order by p.route_id,c.customer_id;
```



#	route_ic	customer_ic	first_name	last_name	gendei	date_of_birtl
1	1	18	Gloria	Richie	F	04-12-1989
2	4	2	Steve	Ryan	M	03-04-1983
3	4	4	Cathenna	Emily	F	14-09-1977

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

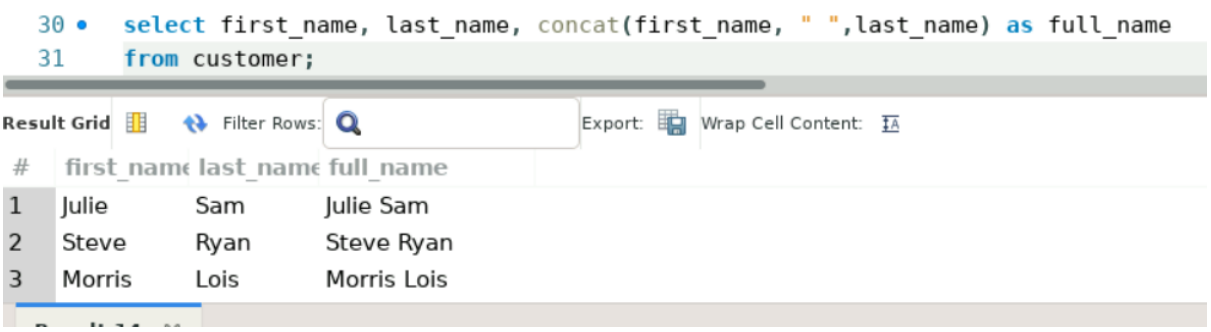


```sql
select count(customer_id) as Total_business_calss_customer,
sum(Price_per_ticket) as Total_revenue_in_business_class
from ticket_details
where class_id = 'Bussiness';
```

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

```
#  Total_business_calss_custon Total_revenue_in_business_cl
1  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.

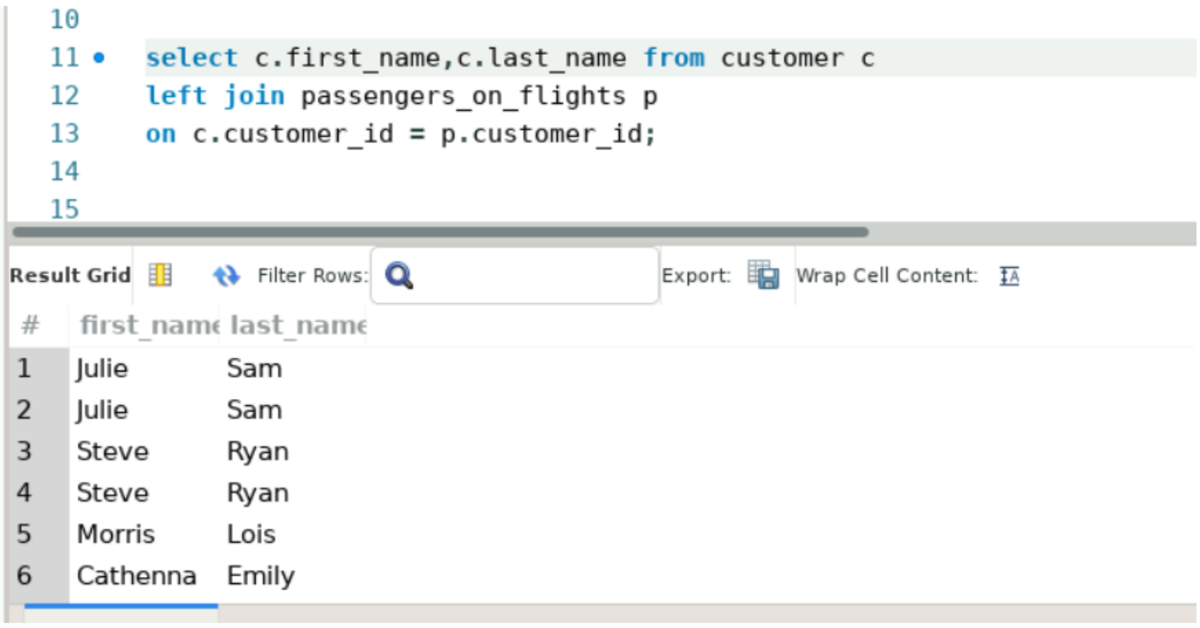


```sql
select first_name, last_name, concat(first_name, " ",last_name) as full_name
from customer;
```

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

```
#  first_name last_name full_name
1  Julie      Sam       Julie Sam
2  Steve      Ryan      Steve Ryan
3  Morris     Lois      Morris Lois
```

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

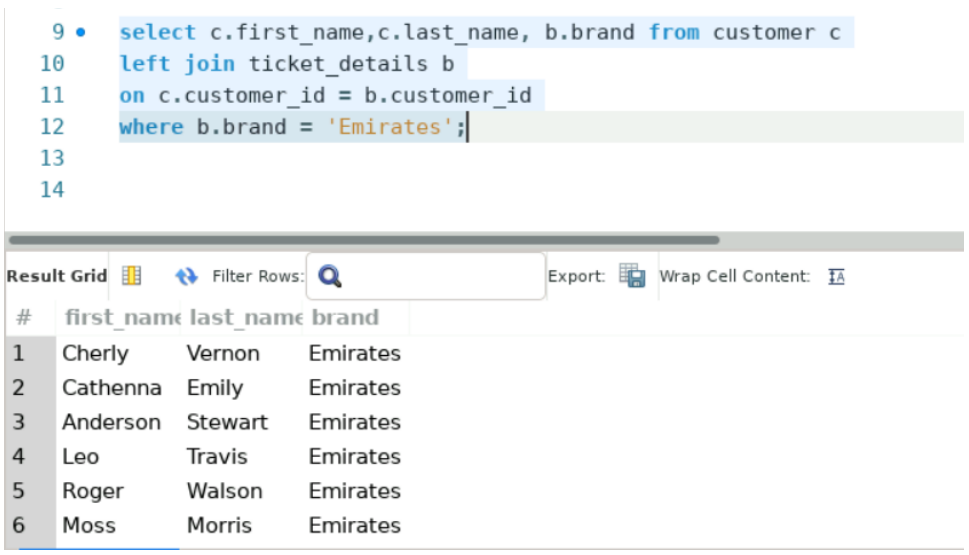


```sql

select c.first_name,c.last_name from customer c
left join passengers_on_flights p
on c.customer_id = p.customer_id;


```

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

```
#  first_name last_name
1  Julie      Sam
2  Julie      Sam
3  Steve      Ryan
4  Steve      Ryan
5  Morris     Lois
6  Cathenna   Emily
```

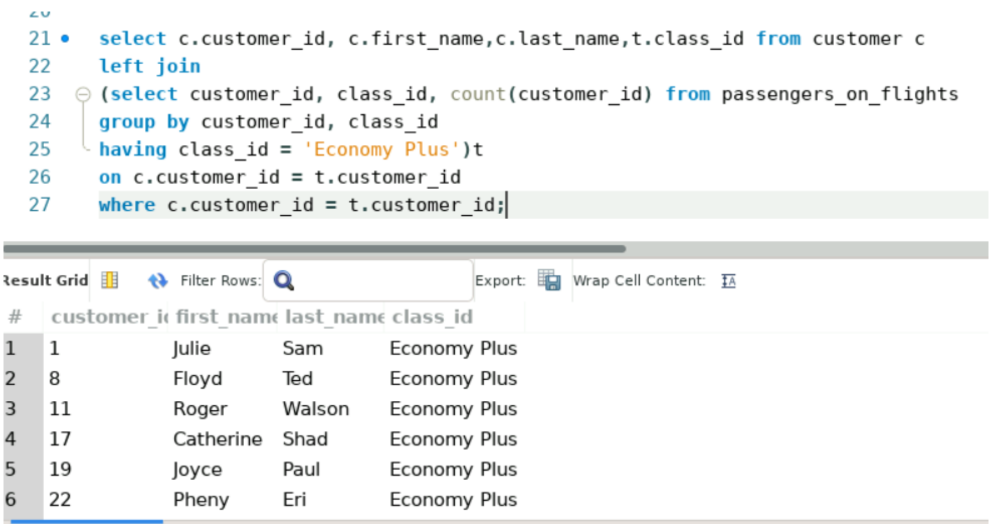
7. Write a query to identify the customer's first name and last name based on their customer ID and brand (Emirates) from the ticket_details table.



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

```
#  first_name last_name brand
1  Cherly     Vernon    Emirates
2  Cathenna   Emily     Emirates
3  Anderson   Stewart   Emirates
4  Leo        Travis    Emirates
5  Roger      Walson    Emirates
6  Moss       Morris    Emirates
```

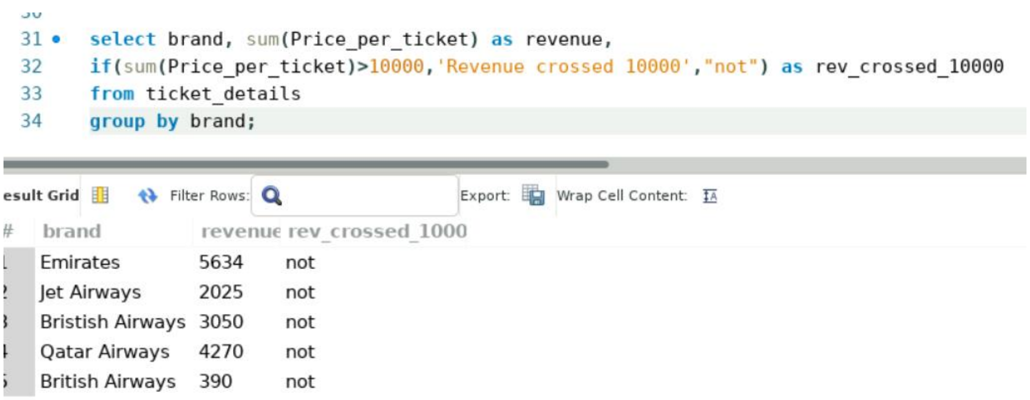
8. Write a query to identify the customers who have travelled by *Economy Plus* class using Group By and Having clause on the passengers_on_flights table.



```sql
select c.customer_id, c.first_name,c.last_name,t.class_id from customer c
left join
(select customer_id, class_id, count(customer_id) from passengers_on_flights
group by customer_id, class_id
having class_id = 'Economy Plus')t
on c.customer_id = t.customer_id
where c.customer_id = t.customer_id;
```

```
#  customer_id first_name last_name class_id
1  1           Julie      Sam       Economy Plus
2  8           Floyd      Ted       Economy Plus
3  11          Roger      Walson    Economy Plus
4  17          Catherine  Shad      Economy Plus
5  19          Joyce      Paul      Economy Plus
6  22          Pheny      Eri       Economy Plus
```

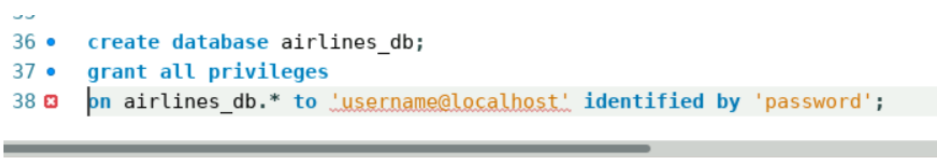
9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.



```sql
select brand, sum(Price_per_ticket) as revenue,
if(sum(Price_per_ticket)>10000,'Revenue crossed 10000',"not") as rev_crossed_10000
from ticket_details
group by brand;
```

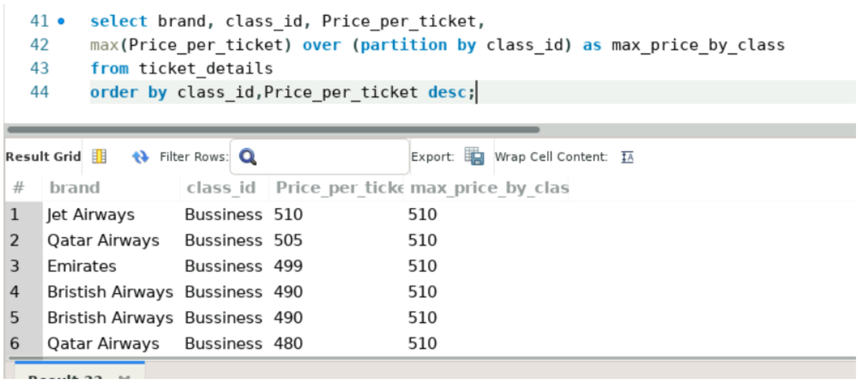
```
#  brand             revenue rev_crossed_1000
1  Emirates          5634    not
2  Jet Airways       2025    not
3  Bristish Airways  3050    not
4  Qatar Airways     4270    not
5  British Airways   390     not
```

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



```
create database airlines_db;
grant all privileges
on airlines_db.* to 'username@localhost' identified by 'password';
```

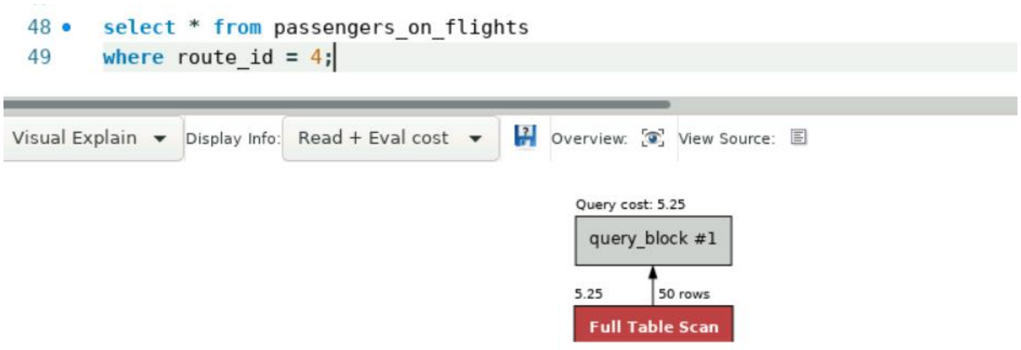
11. Write a query to find the maximum ticket price for each class using window functions on the ticket_details table.



```
select brand, class_id, Price_per_ticket,
max(Price_per_ticket) over (partition by class_id) as max_price_by_class
from ticket_details
order by class_id,Price_per_ticket desc;
```

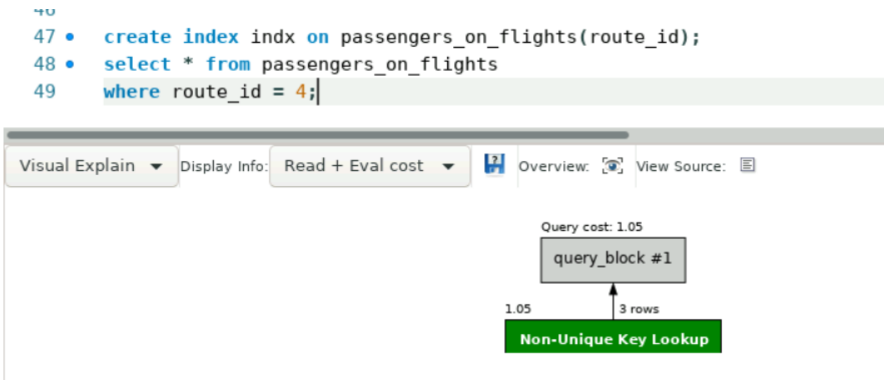
#	brand	class_id	Price_per_ticke	max_price_by_clas
1	Jet Airways	Bussiness	510	510
2	Qatar Airways	Bussiness	505	510
3	Emirates	Bussiness	499	510
4	Bristish Airways	Bussiness	490	510
5	Bristish Airways	Bussiness	490	510
6	Qatar Airways	Bussiness	480	510

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.



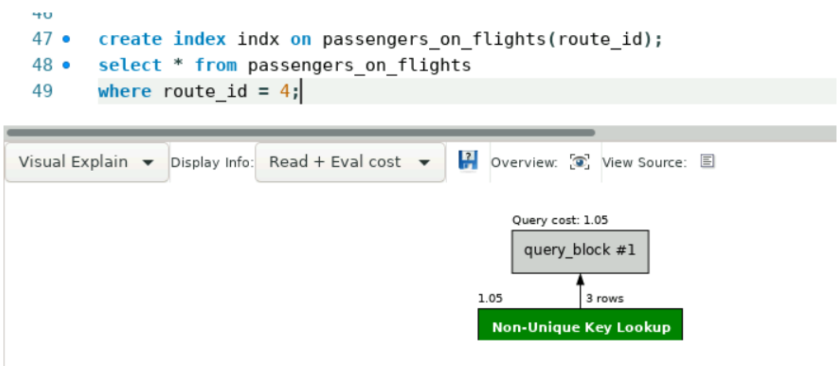
```
select * from passengers_on_flights
where route_id = 4;
```

After creating index on route_id column



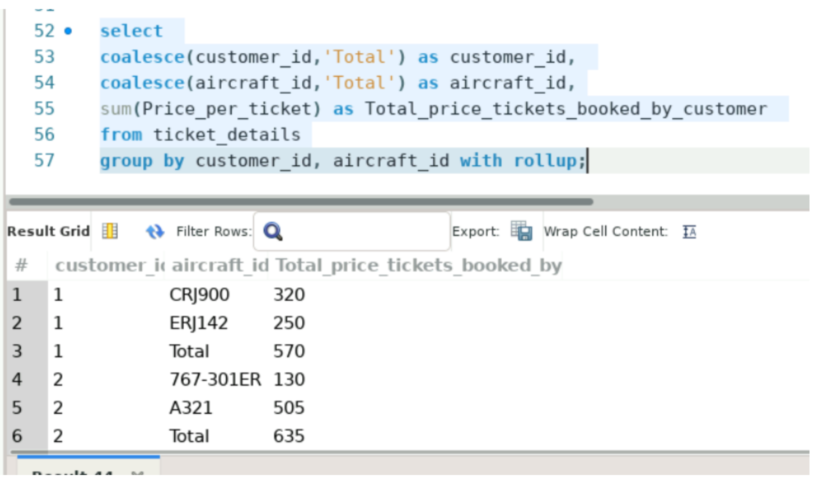
```
create index indx on passengers_on_flights(route_id);
select * from passengers_on_flights
where route_id = 4;
```

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



```sql
create index indx on passengers_on_flights(route_id);
select * from passengers_on_flights
where route_id = 4;
```

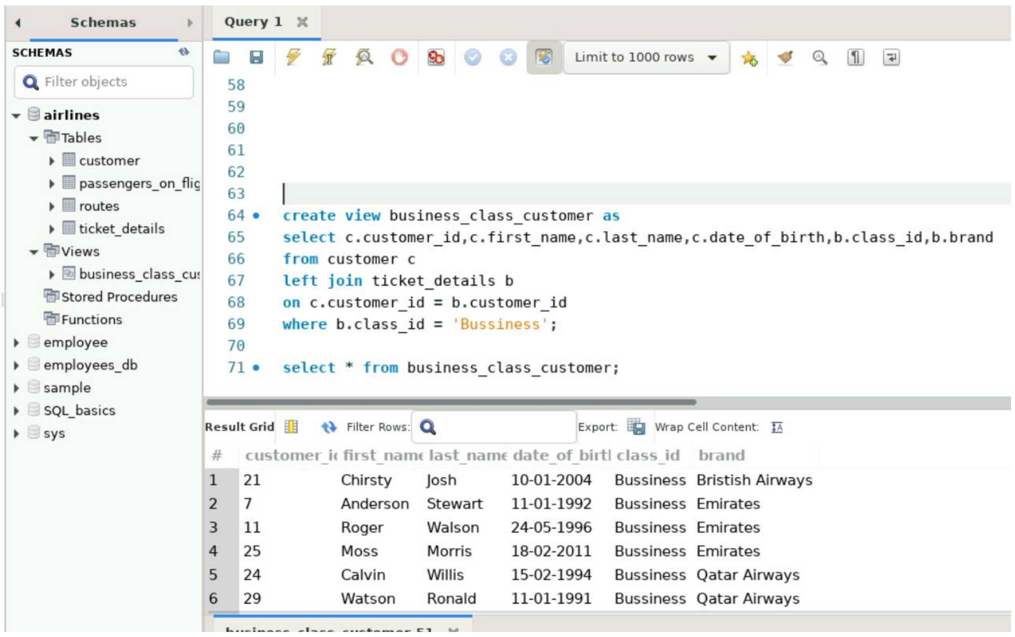
14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.



```sql
select
coalesce(customer_id,'Total') as customer_id,
coalesce(aircraft_id,'Total') as aircraft_id,
sum(Price_per_ticket) as Total_price_tickets_booked_by_customer
from ticket_details
group by customer_id, aircraft_id with rollup;
```

#	customer_i	aircraft_id	Total_price_tickets_booked_by
1	1	CRJ900	320
2	1	ERJ142	250
3	1	Total	570
4	2	767-301ER	130
5	2	A321	505
6	2	Total	635

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

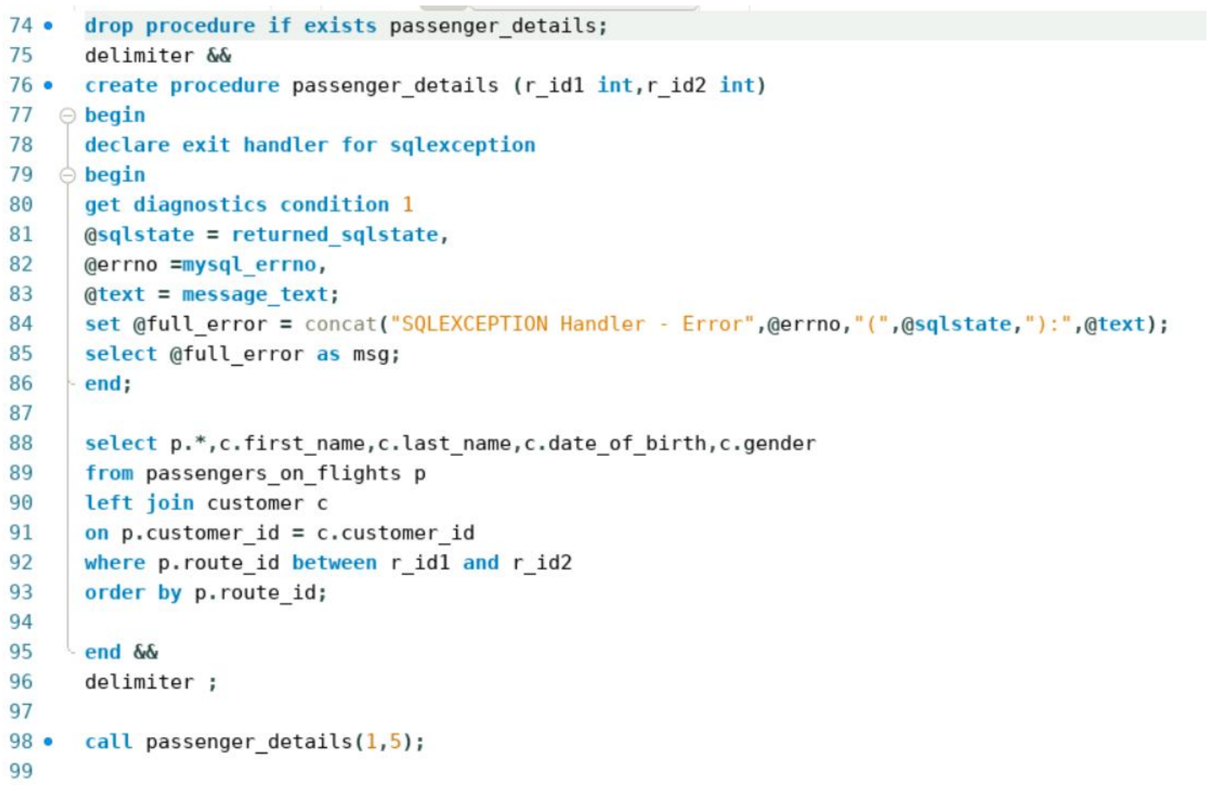


```sql
create view business_class_customer as
select c.customer_id,c.first_name,c.last_name,c.date_of_birth,b.class_id,b.brand
from customer c
left join ticket_details b
on c.customer_id = b.customer_id
where b.class_id = 'Bussiness';

select * from business_class_customer;
```

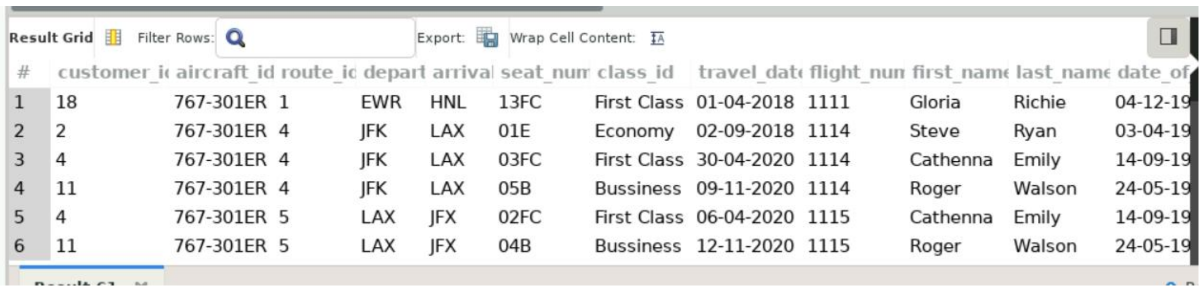
#	customer_i	first_nam	last_nam	date_of_birt	class_id	brand
1	21	Chirsty	Josh	10-01-2004	Bussiness	Bristish Airways
2	7	Anderson	Stewart	11-01-1992	Bussiness	Emirates
3	11	Roger	Walson	24-05-1996	Bussiness	Emirates
4	25	Moss	Morris	18-02-2011	Bussiness	Emirates
5	24	Calvin	Willis	15-02-1994	Bussiness	Qatar Airways
6	29	Watson	Ronald	11-01-1991	Bussiness	Qatar Airways

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.

Example: with correct table name: **passengers-on_flights**

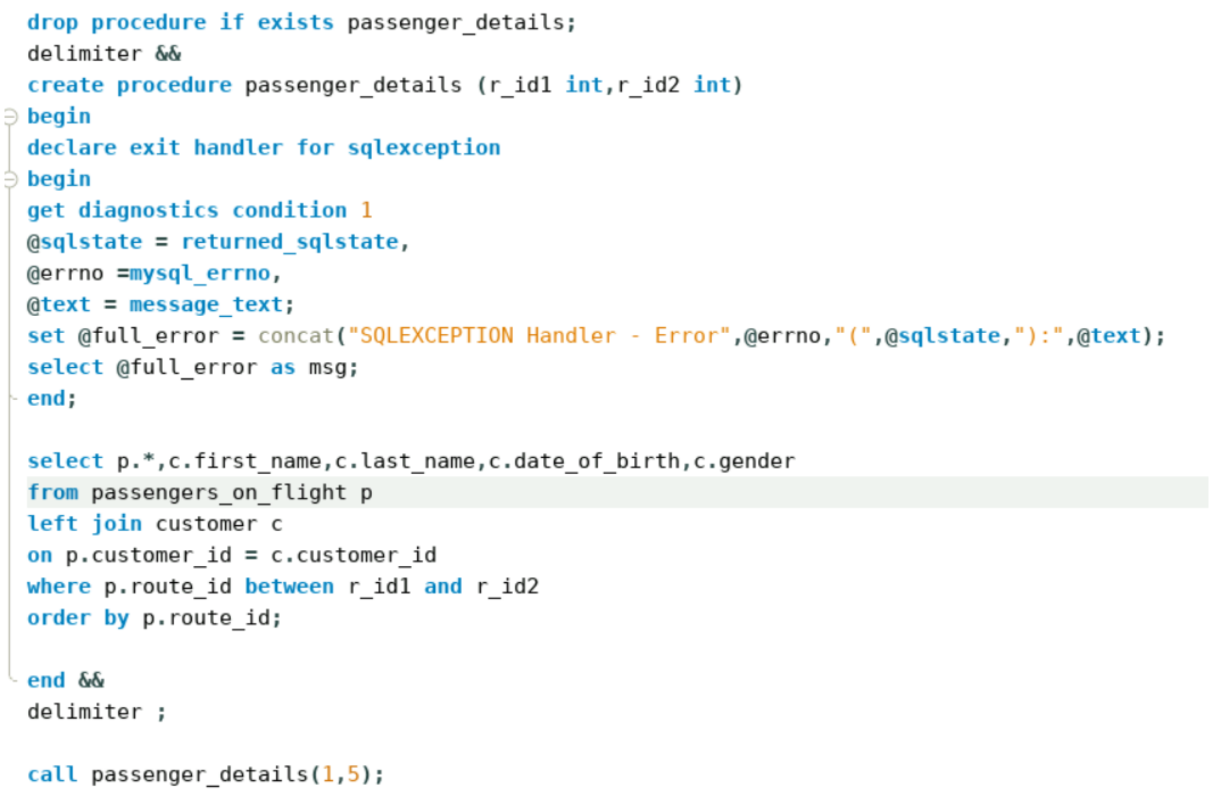


```
drop procedure if exists passenger_details;
delimiter &&
create procedure passenger_details (r_id1 int,r_id2 int)
begin
declare exit handler for sqlexception
begin
get diagnostics condition 1
@sqlstate = returned_sqlstate,
@errno =mysql_errno,
@text = message_text;
set @full_error = concat("SQLEXCEPTION Handler - Error",@errno,"(",@sqlstate,"):",@text);
select @full_error as msg;
end;

select p.*,c.first_name,c.last_name,c.date_of_birth,c.gender
from passengers_on_flights p
left join customer c
on p.customer_id = c.customer_id
where p.route_id between r_id1 and r_id2
order by p.route_id;

end &&
delimiter ;

call passenger_details(1,5);
```

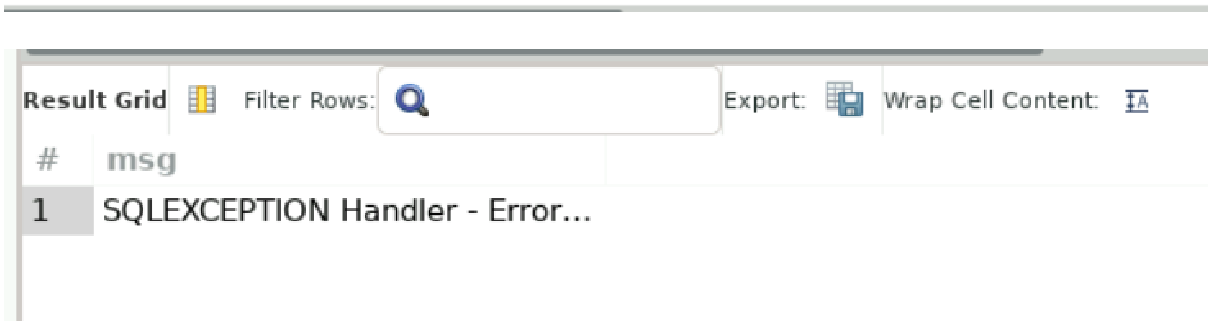


#	customer_i	aircraft_id	route_ic	depart	arrival	seat_num	class_id	travel_dat	flight_num	first_name	last_name	date_of
1	18	767-301ER	1	EWR	HNL	13FC	First Class	01-04-2018	1111	Gloria	Richie	04-12-19
2	2	767-301ER	4	JFK	LAX	01E	Economy	02-09-2018	1114	Steve	Ryan	03-04-19
3	4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114	Cathenna	Emily	14-09-19
4	11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114	Roger	Walson	24-05-19
5	4	767-301ER	5	LAX	JFX	02FC	First Class	06-04-2020	1115	Cathenna	Emily	14-09-19
6	11	767-301ER	5	LAX	JFX	04B	Bussiness	12-11-2020	1115	Roger	Walson	24-05-19

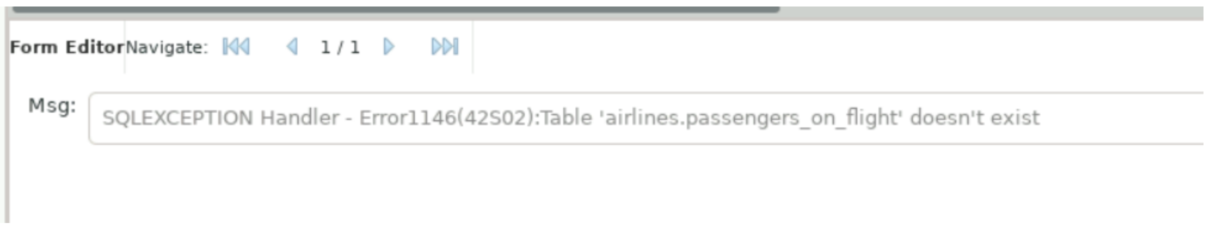
Example with incorrect table name : **passengers_on_flight (incorrect spelling)**



```
drop procedure if exists passenger_details;
delimiter &&
create procedure passenger_details (r_id1 int,r_id2 int)
begin
declare exit handler for sqlexception
begin
get diagnostics condition 1
@sqlstate = returned_sqlstate,
@errno =mysql_errno,
@text = message_text;
set @full_error = concat("SQLEXCEPTION Handler - Error",@errno,"(",@sqlstate,"):",@text);
select @full_error as msg;
end;

select p.*,c.first_name,c.last_name,c.date_of_birth,c.gender
from passengers_on_flight p
left join customer c
on p.customer_id = c.customer_id
where p.route_id between r_id1 and r_id2
order by p.route_id;

end &&
delimiter ;

call passenger_details(1,5);
```

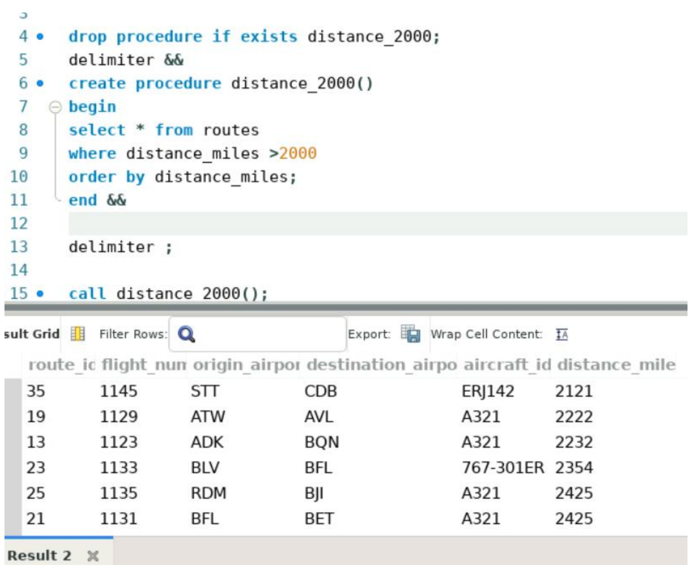


```
#   msg
1   SQLEXCEPTION Handler - Error...
```



Msg: SQLEXCEPTION Handler - Error1146(42S02):Table 'airlines.passengers_on_flight' doesn't exist

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.



```
drop procedure if exists distance_2000;
delimiter &&
create procedure distance_2000()
begin
select * from routes
where distance_miles >2000
order by distance_miles;
end &&

delimiter ;

call distance_2000();
```

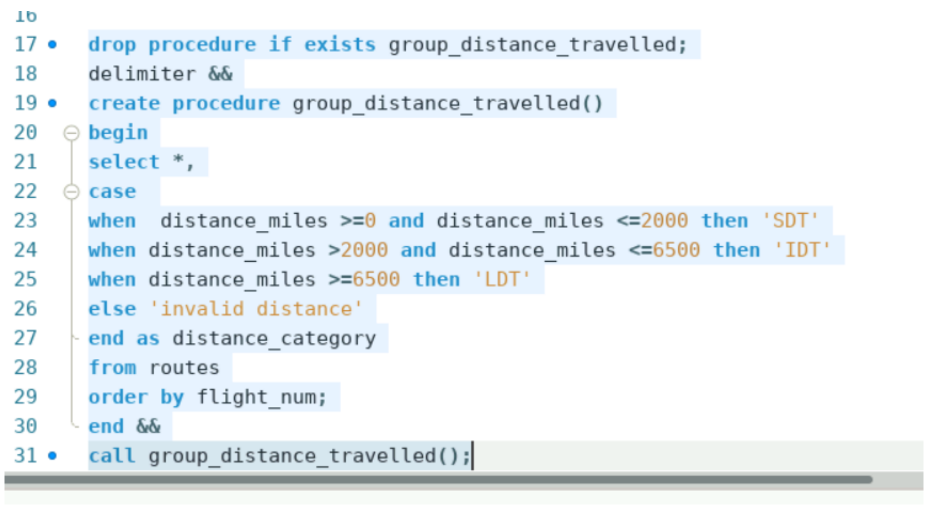
sult Grid Filter Rows: Export: Wrap Cell Content:

```
route_ic flight_nun origin_airpoi destination_airpo aircraft_id distance_mile
35       1145       STT           CDB               ERJ142      2121
19       1129       ATW           AVL               A321        2222
13       1123       ADK           BQN               A321        2232
23       1133       BLV           BFL               767-301ER   2354
25       1135       RDM           BJI               A321        2425
21       1131       BFL           BET               A321        2425
```

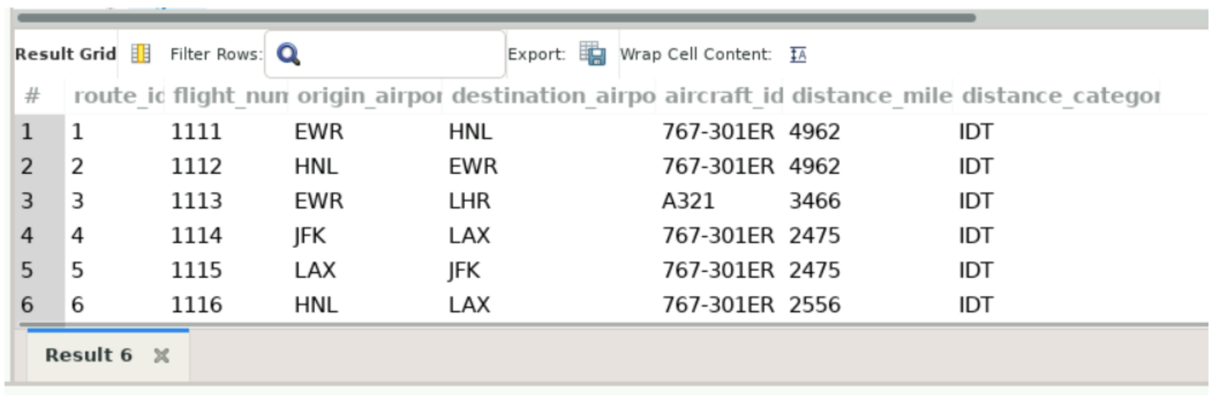
Result 2

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.

Without passing any parameters:



```
drop procedure if exists group_distance_travelled;
delimiter &&
create procedure group_distance_travelled()
begin
select *,
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 'invalid distance'
end as distance_category
from routes
order by flight_num;
end &&
call group_distance_travelled();
```

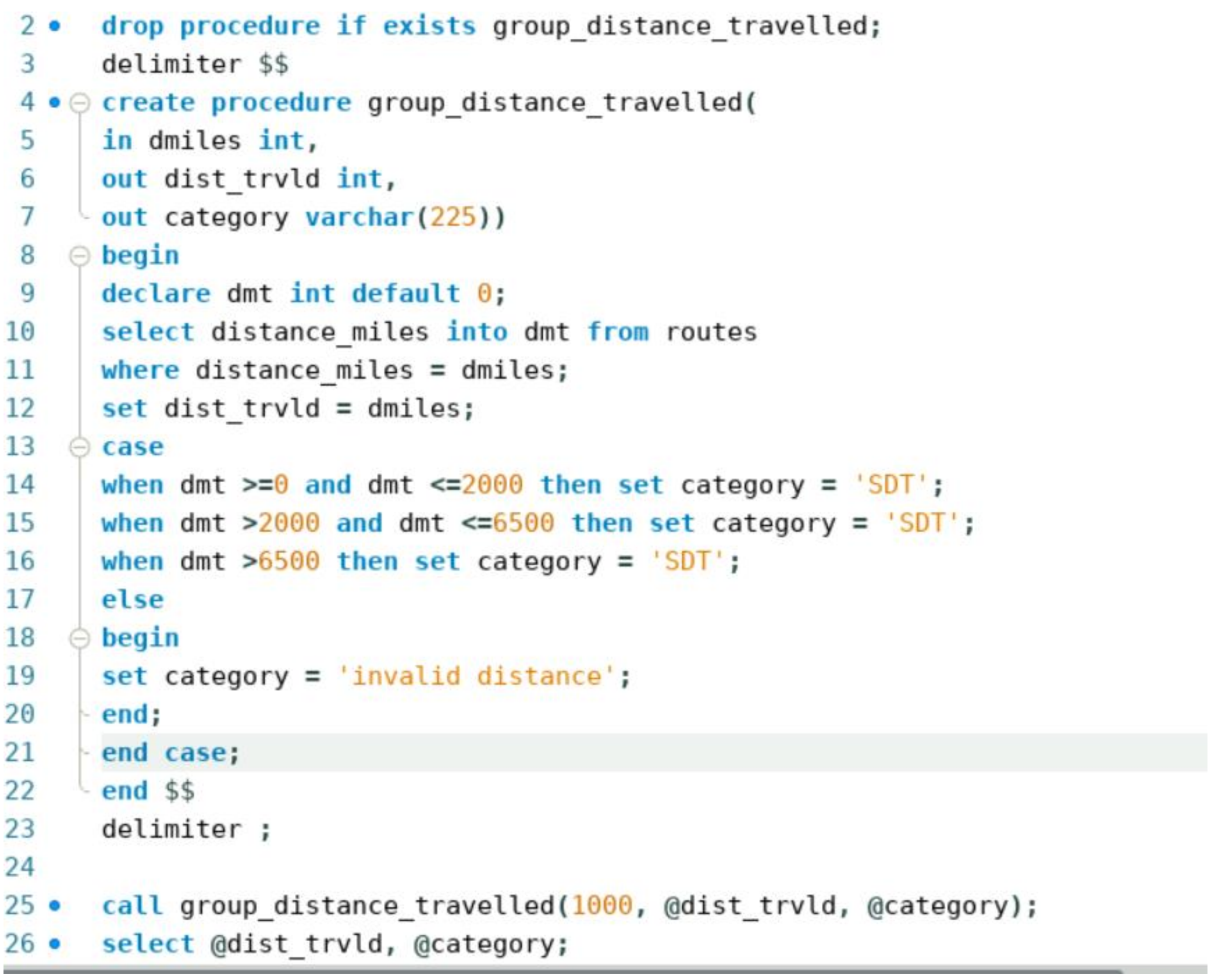


Result Grid Filter Rows: Export: Wrap Cell Content:

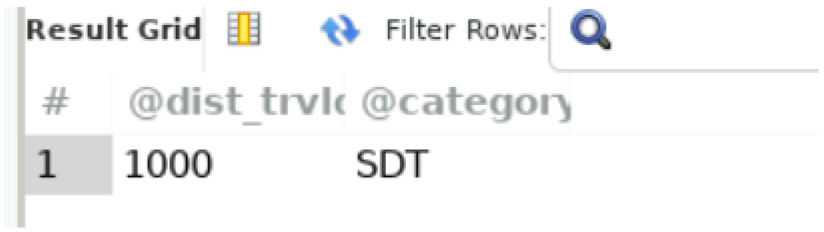
```
#  route_ic flight_nun origin_airpoi destination_airpo aircraft_id distance_mile distance_categoi
1  1        1111       EWR           HNL               767-301ER   4962          IDT
2  2        1112       HNL           EWR               767-301ER   4962          IDT
3  3        1113       EWR           LHR               A321        3466          IDT
4  4        1114       JFK           LAX               767-301ER   2475          IDT
5  5        1115       LAX           JFK               767-301ER   2475          IDT
6  6        1116       HNL           LAX               767-301ER   2556          IDT
```

Result 6

With passing parameters:



```
drop procedure if exists group_distance_travelled;
delimiter $$
create procedure group_distance_travelled(
in dmiles int,
out dist_trvld int,
out category varchar(225))
begin
declare dmt int default 0;
select distance_miles into dmt from routes
where distance_miles = dmiles;
set dist_trvld = dmiles;
case
when dmt >=0 and dmt <=2000 then set category = 'SDT';
when dmt >2000 and dmt <=6500 then set category = 'SDT';
when dmt >6500 then set category = 'SDT';
else
begin
set category = 'invalid distance';
end;
end case;
end $$
delimiter ;

call group_distance_travelled(1000, @dist_trvld, @category);
select @dist_trvld, @category;
```



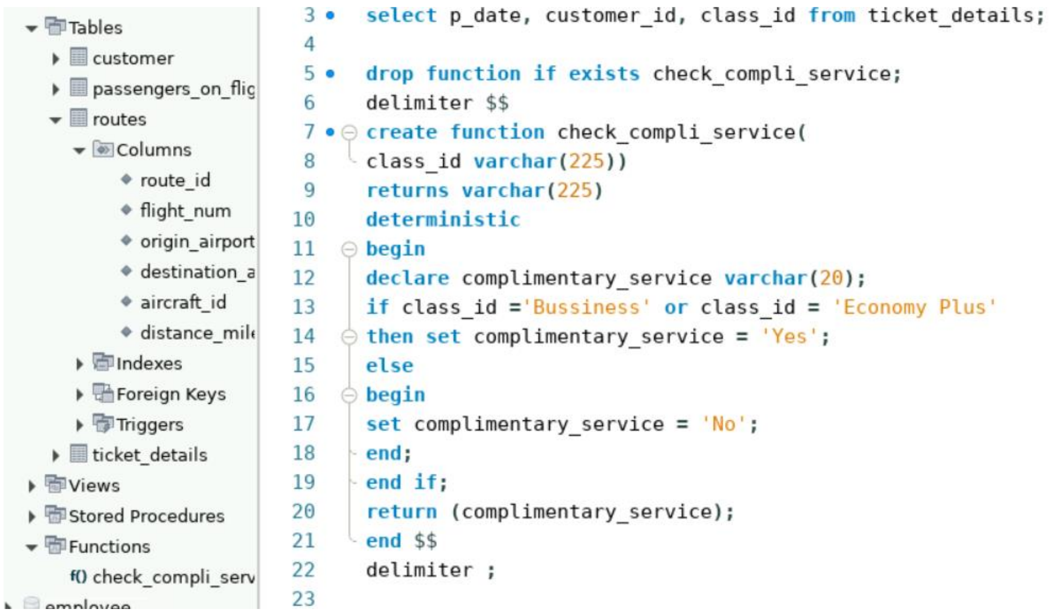
```
#  @dist_trvl( @categor)
1  1000        SDT
```

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:

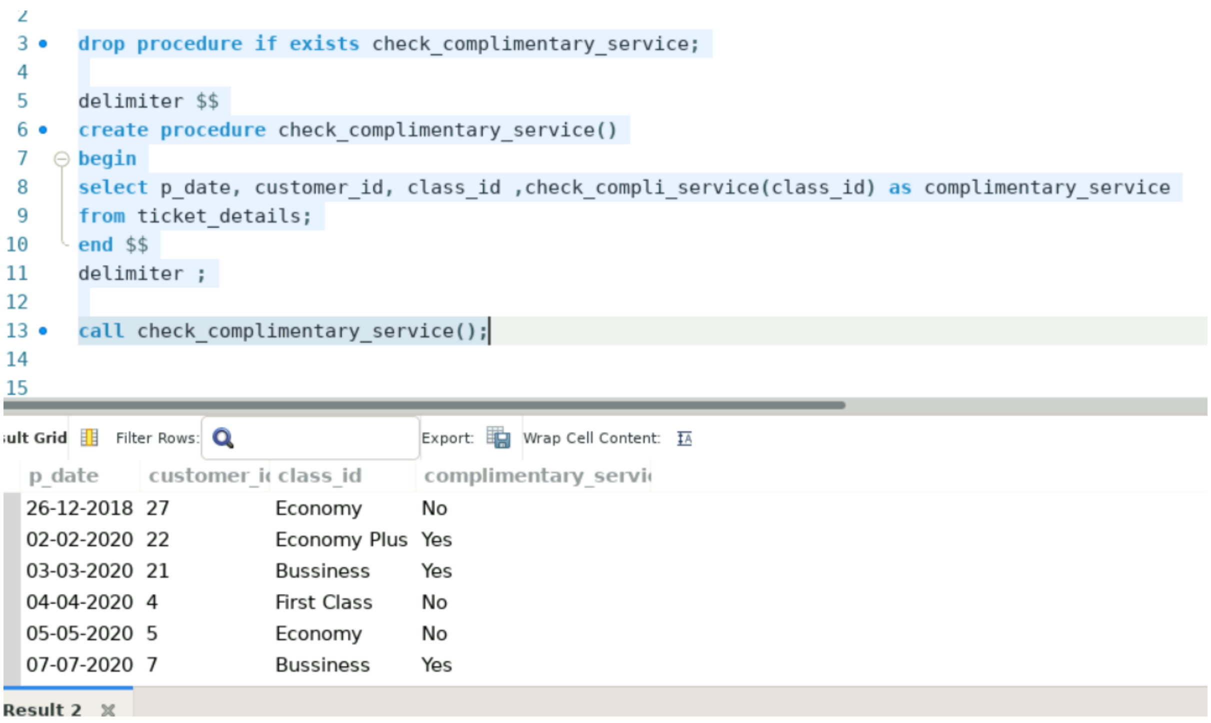
- If the class is *Business* and *Economy Plus*, then complimentary services are given as *Yes*, else it is *No*

Stored function:



```
select p_date, customer_id, class_id from ticket_details;

drop function if exists check_compli_service;
delimiter $$
create function check_compli_service(
class_id varchar(225))
returns varchar(225)
deterministic
begin
declare complimentary_service varchar(20);
if class_id ='Bussiness' or class_id = 'Economy Plus'
then set complimentary_service = 'Yes';
else
begin
set complimentary_service = 'No';
end;
end if;
return (complimentary_service);
end $$
delimiter ;
```

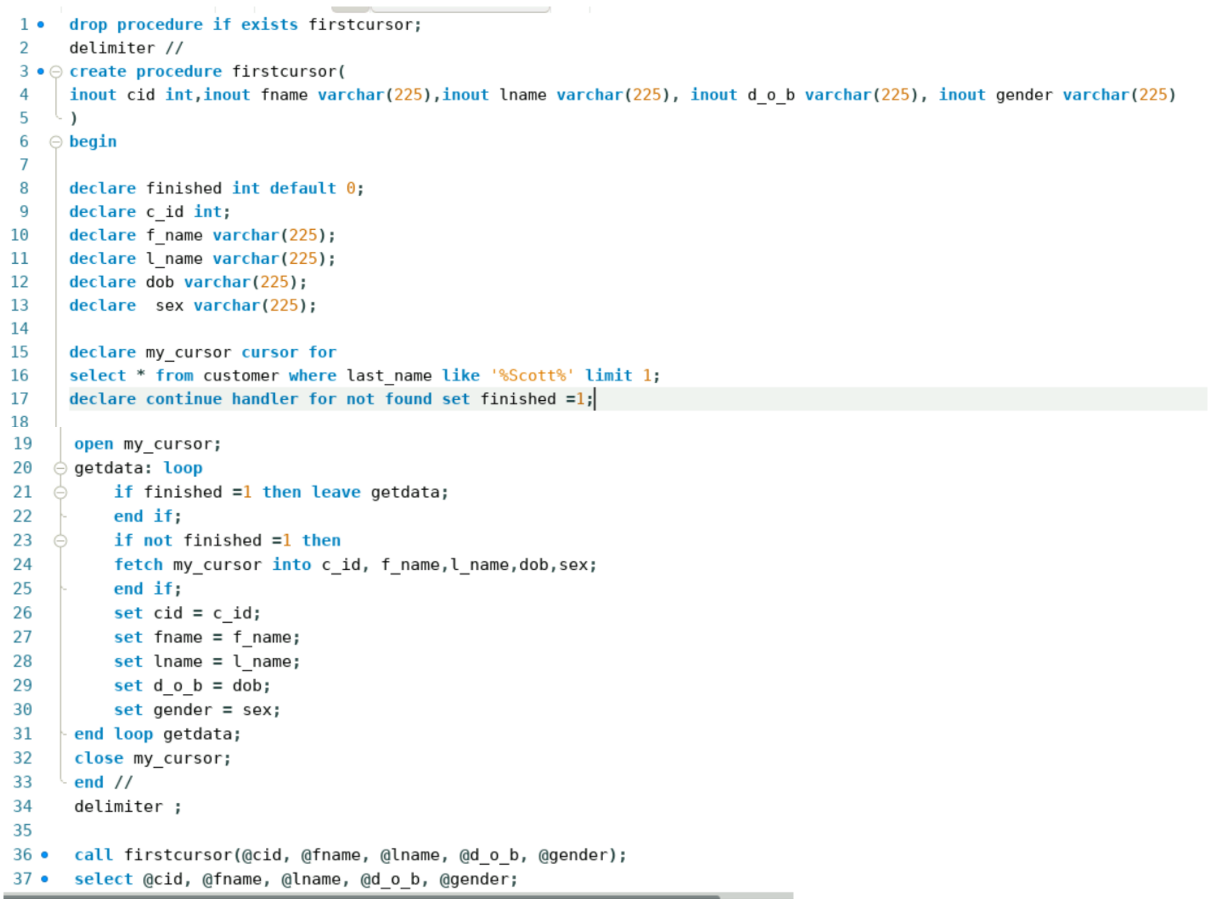
Stored procedure using stored function:



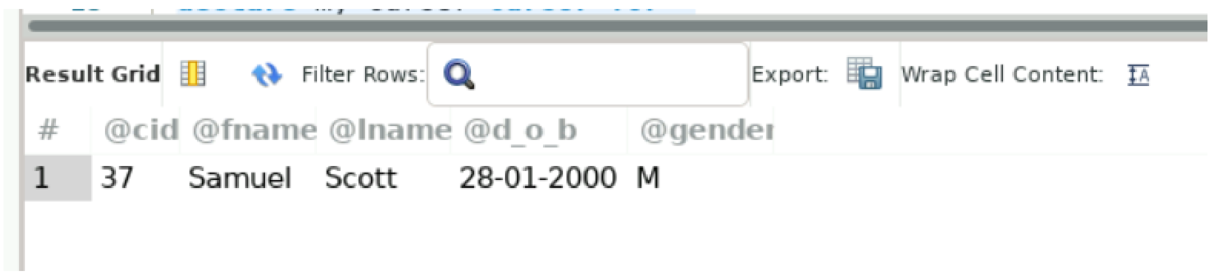
```
drop procedure if exists check_complimentary_service;

delimiter $$
create procedure check_complimentary_service()
begin
select p_date, customer_id, class_id ,check_compli_service(class_id) as complimentary_service
from ticket_details;
end $$
delimiter ;

call check_complimentary_service();
```

```
p_date      customer_i class_id       complimentary_servi
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
```

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.



```
drop procedure if exists firstcursor;
delimiter //
create procedure firstcursor(
inout cid int,inout fname varchar(225),inout lname varchar(225), inout d_o_b varchar(225), inout gender varchar(225)
)
begin

declare finished int default 0;
declare c_id int;
declare f_name varchar(225);
declare l_name varchar(225);
declare dob varchar(225);
declare  sex varchar(225);

declare my_cursor cursor for
select * from customer where last_name like '%Scott%' limit 1;
declare continue handler for not found set finished =1;

 open my_cursor;
 getdata: loop
     if finished =1 then leave getdata;
     end if;
     if not finished =1 then
     fetch my_cursor into c_id, f_name,l_name,dob,sex;
     end if;
     set cid = c_id;
     set fname = f_name;
     set lname = l_name;
     set d_o_b = dob;
     set gender = sex;
 end loop getdata;
 close my_cursor;
 end //
 delimiter ;

 call firstcursor(@cid, @fname, @lname, @d_o_b, @gender);
 select @cid, @fname, @lname, @d_o_b, @gender;
```



#	@cid	@fname	@lname	@d_o_b	@gender
1	37	Samuel	Scott	28-01-2000	M

################################### END###################################