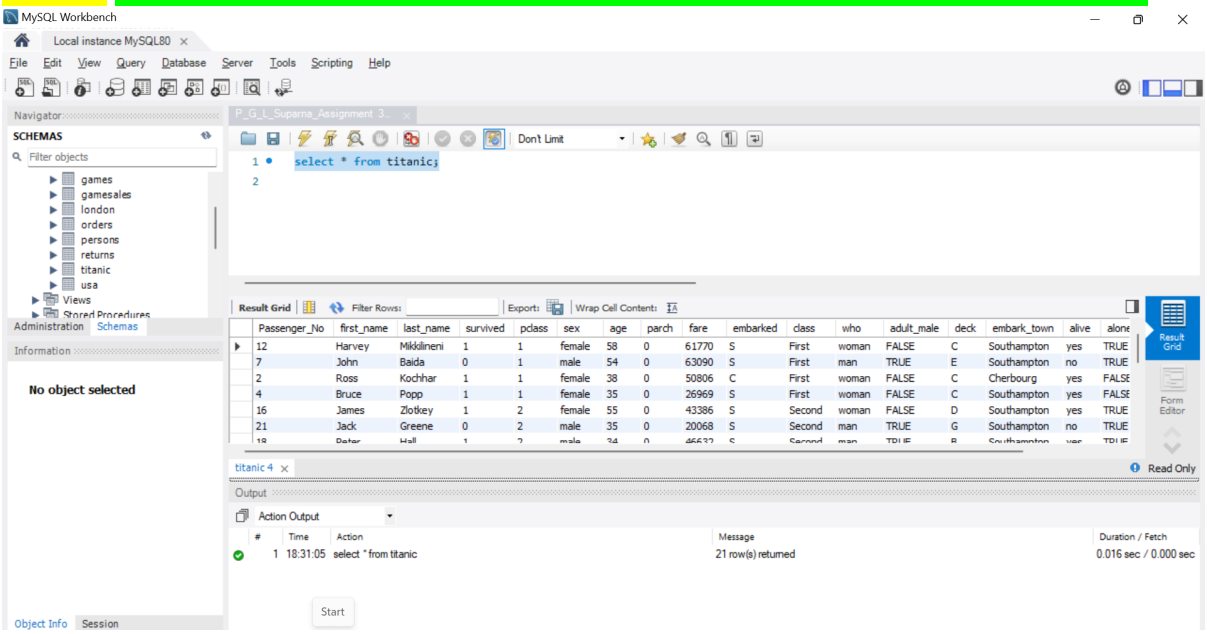
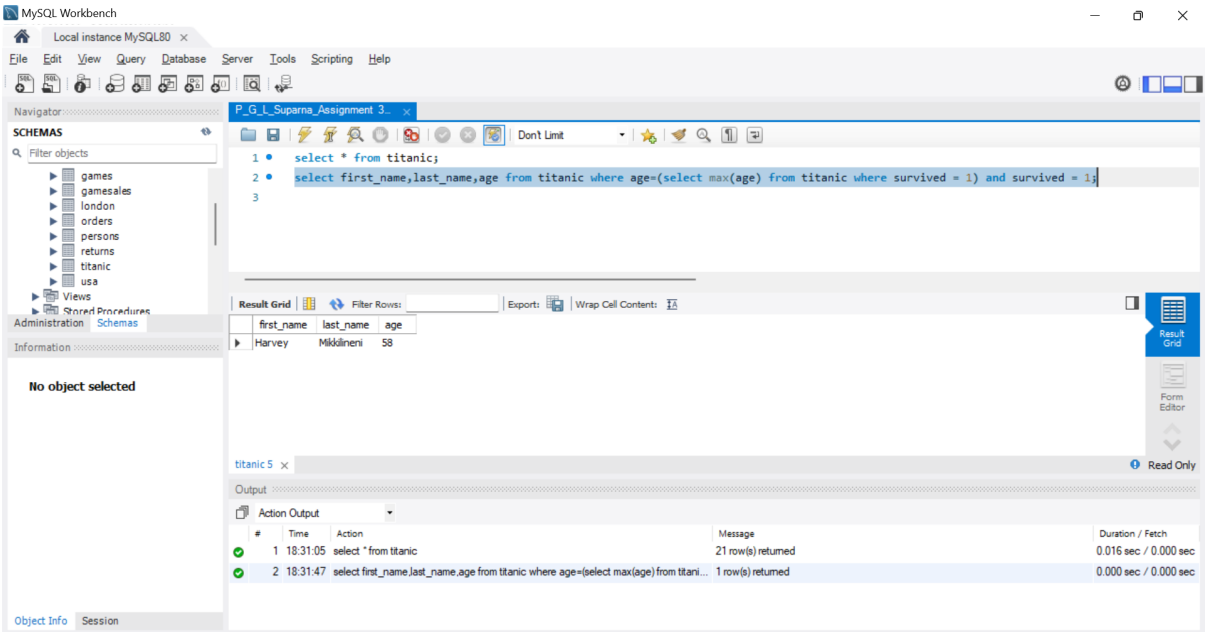
Assignment 3 Analyzing the Titanic Passengers Dataset Using Advanced MySQL Queries

Task 1: **Write a query to find the name and age of the oldest passenger who survived:**

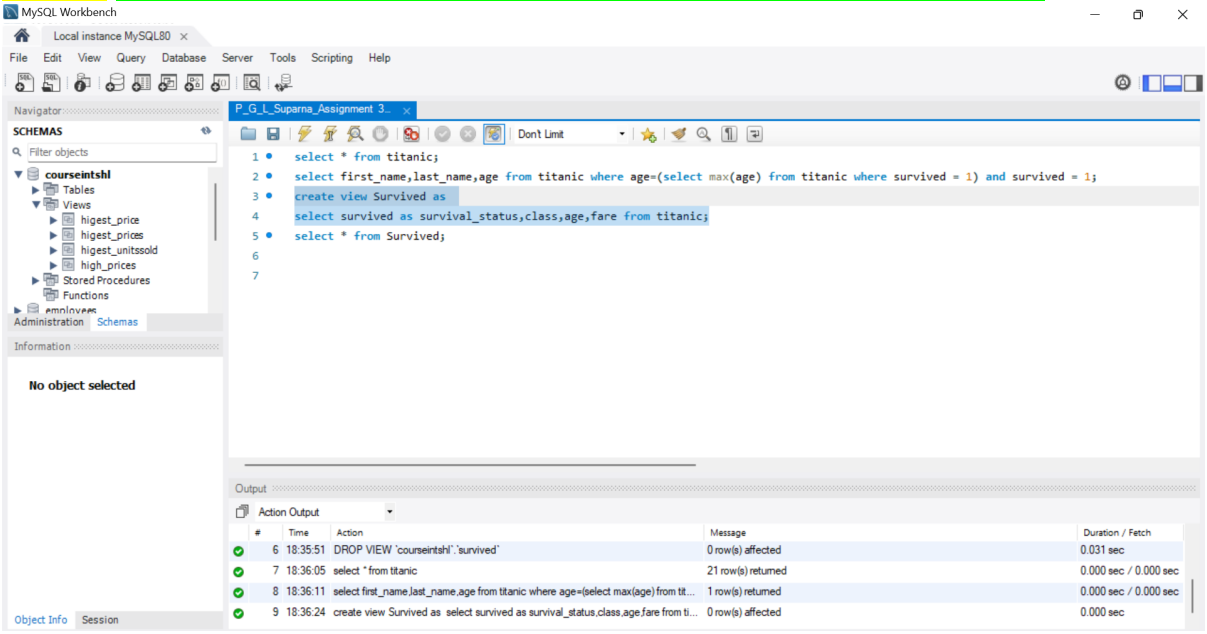


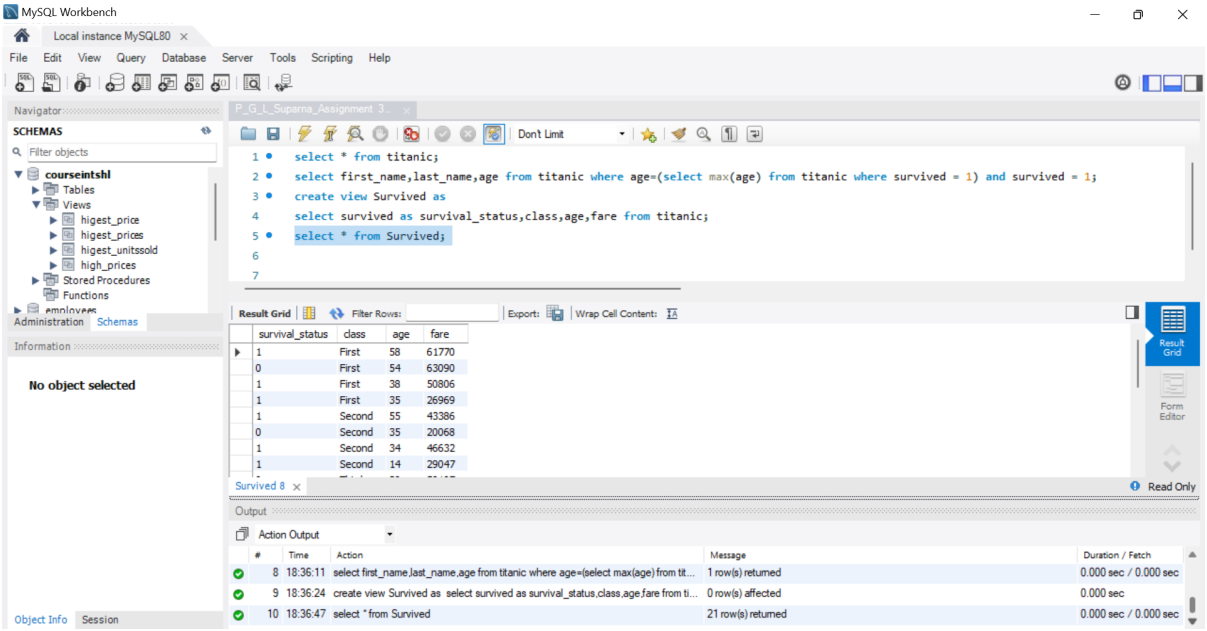


QUERY:

```
select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;
```

Task 2: Create a view to display passenger survival status, class, age, and fare:





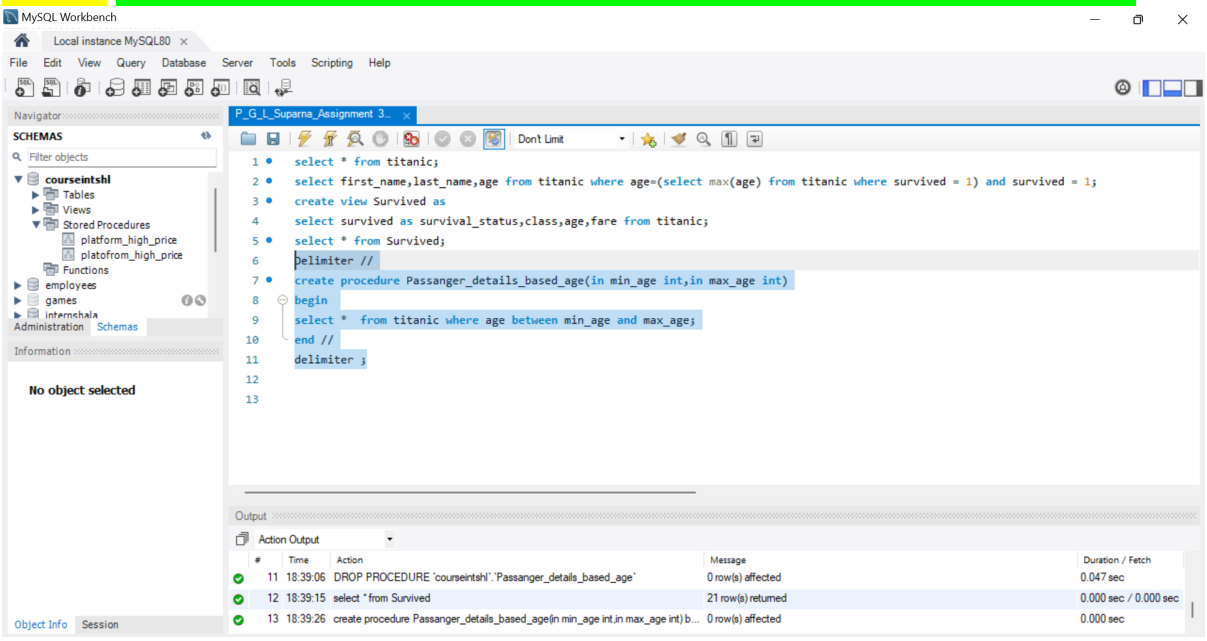
QUERY:

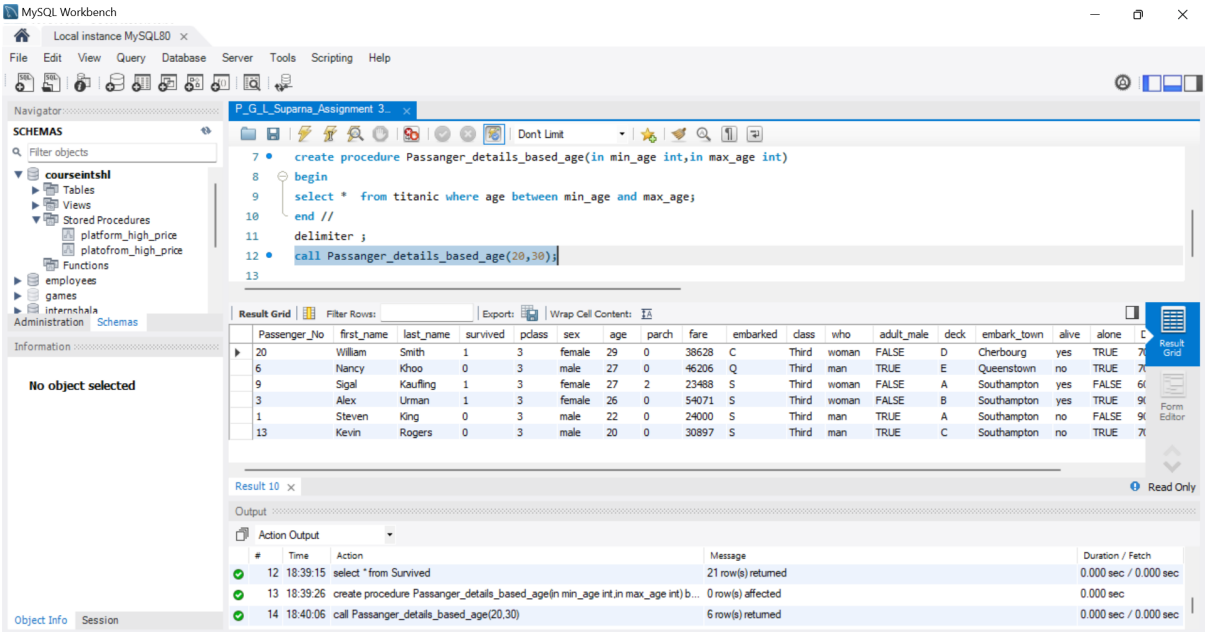
```
create view Survived as

select survived as survival_status,class,age,fare from titanic;

select * from Survived;
```

Task 3: Create a stored procedure to retrieve passengers based on a given age range:



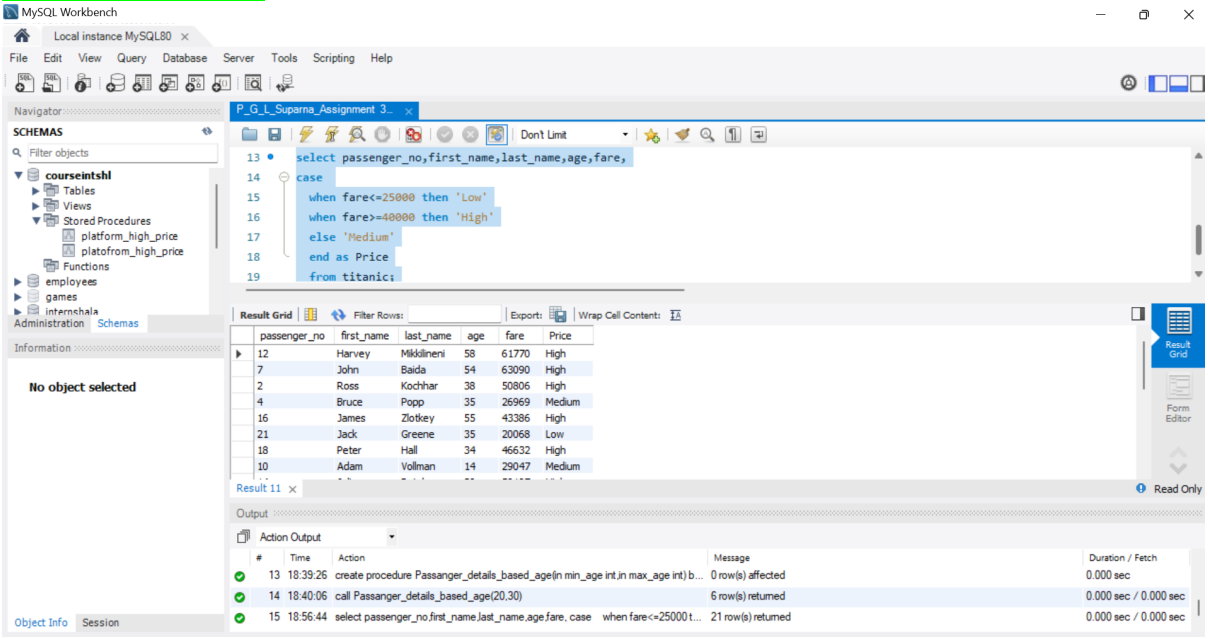


QUERY:

```
Delimiter //
create procedure Passanger_details_based_age(in min_age int,in max_age int)
begin
select *  from titanic where age between min_age and max_age;
end //
delimiter ;
```

call Passanger_details_based_age(20,30);

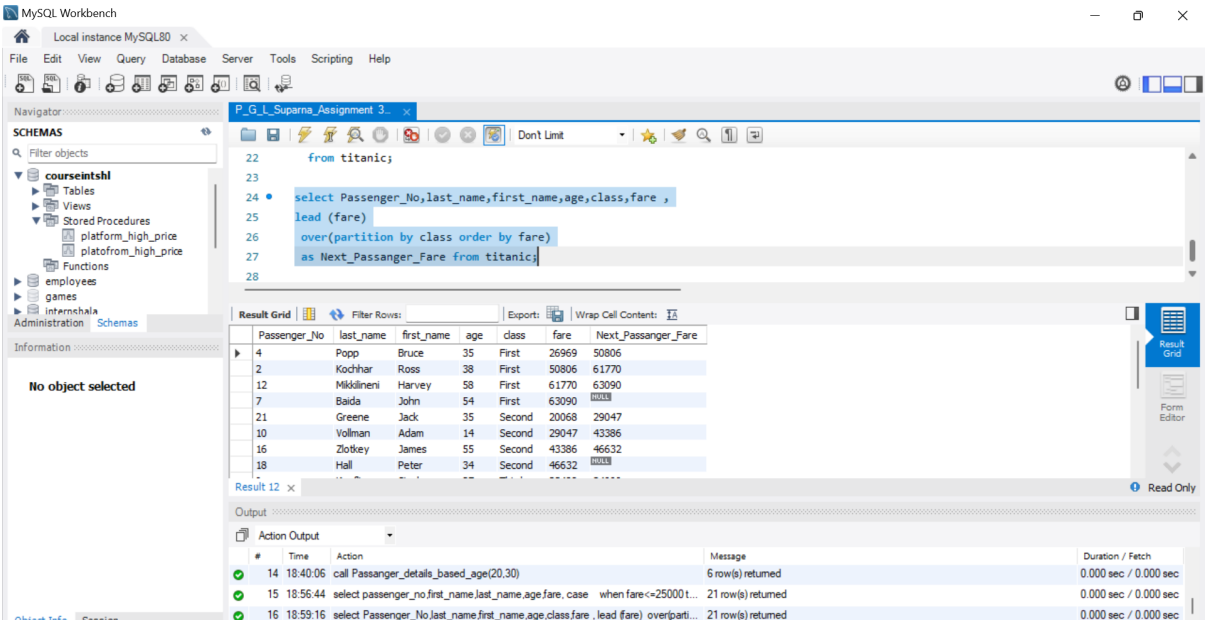
Task 4: Write a query to categorize passengers based on the fare they paid: 'Low', 'Medium', or 'High':



QUERY:

```
select passenger_no,first_name,last_name,age,fare,
case
 when fare<=25000 then 'Low'
 when fare>=40000 then 'High'
 else 'Medium'
 end as Price
 from titanic;
```

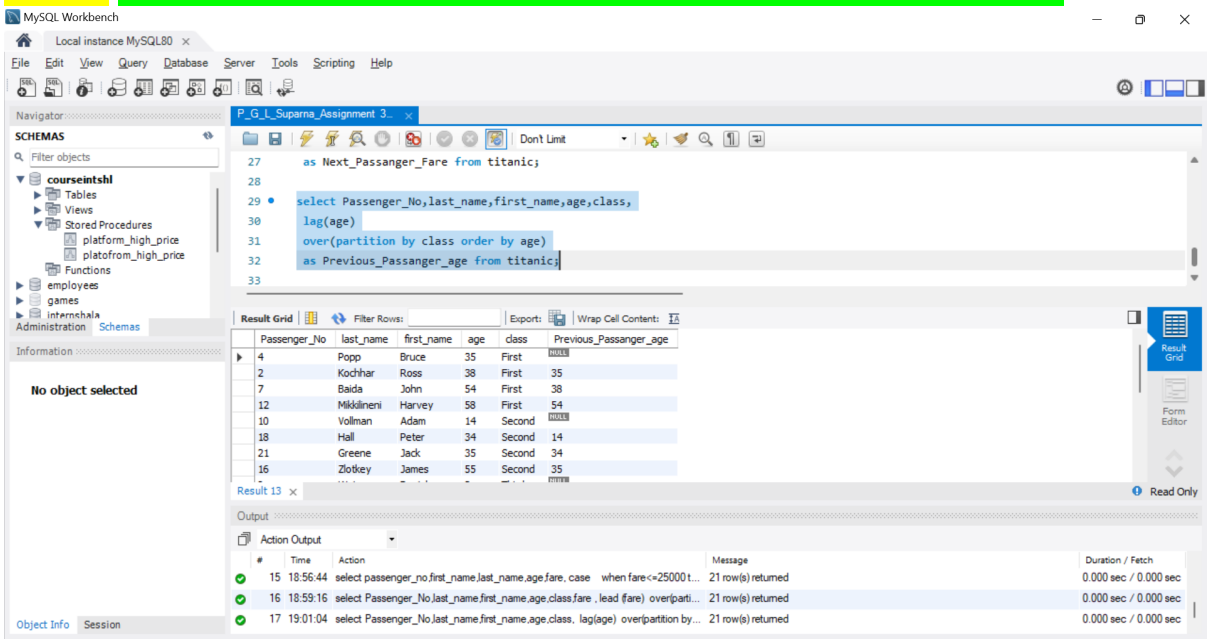
Task 5: Show each passenger's fare and the fare of the next passenger:



QUERY:

```
select Passenger_No,last_name,first_name,age,class,fare ,
lead (fare)
 over(partition by class order by fare)
 as Next_Passanger_Fare from titanic;
```

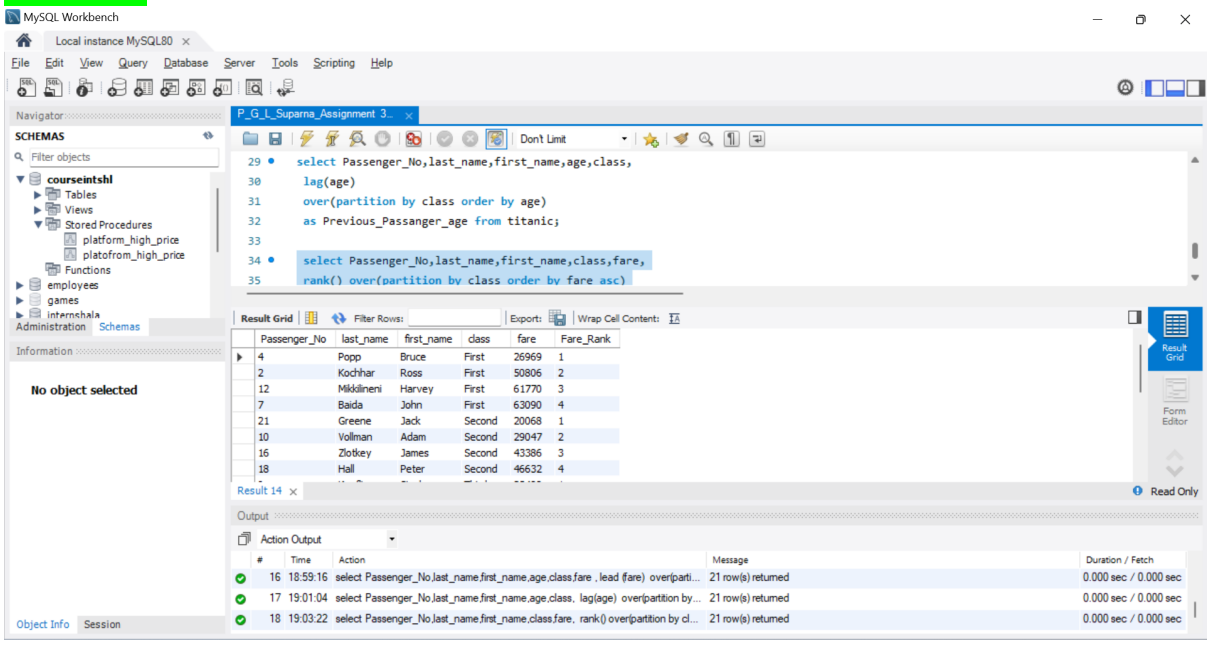
Task 6: Show the age of each passenger and the age of the previous passenger:



QUERY:

```
select Passenger_No,last_name,first_name,age,class,
lag(age)
over(partition by class order by age)
as Previous_Passanger_age from titanic;
```

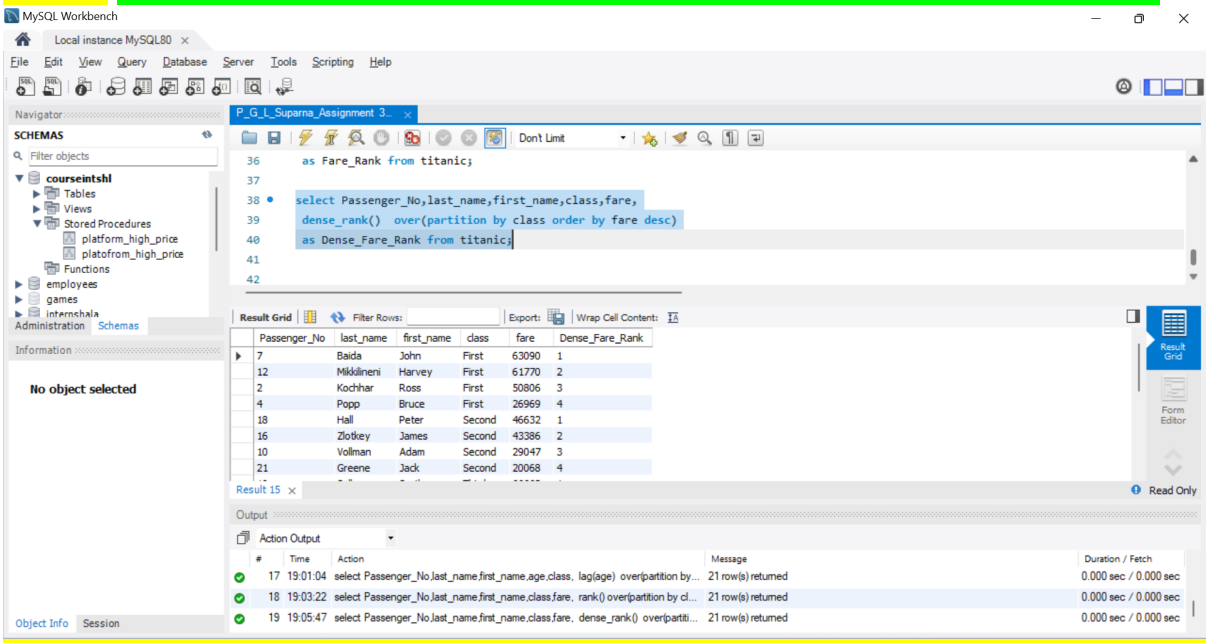
Task 7: **Write a query to rank passengers based on their fare, displaying rank for each passenger:**



QUERY:

```
select Passenger_No,last_name,first_name,class,fare,
 rank() over(partition by class order by fare asc)
 as Fare_Rank from titanic;
```

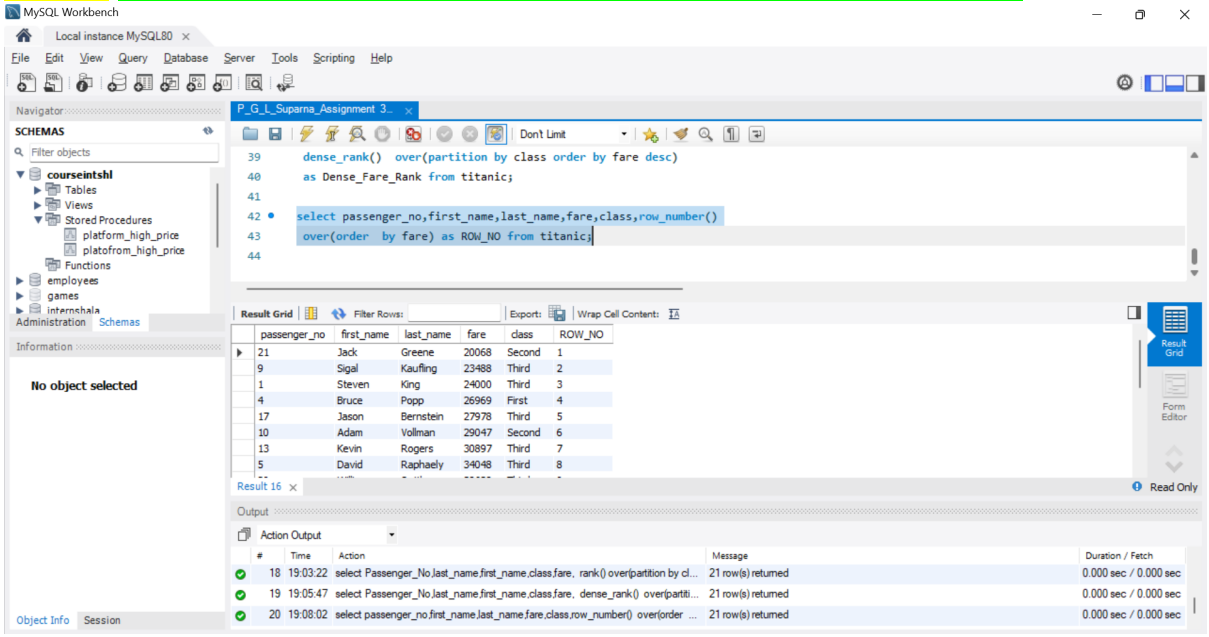
Task 8: **Write a query to rank passengers based on their fare, ensuring no gaps in rank:**

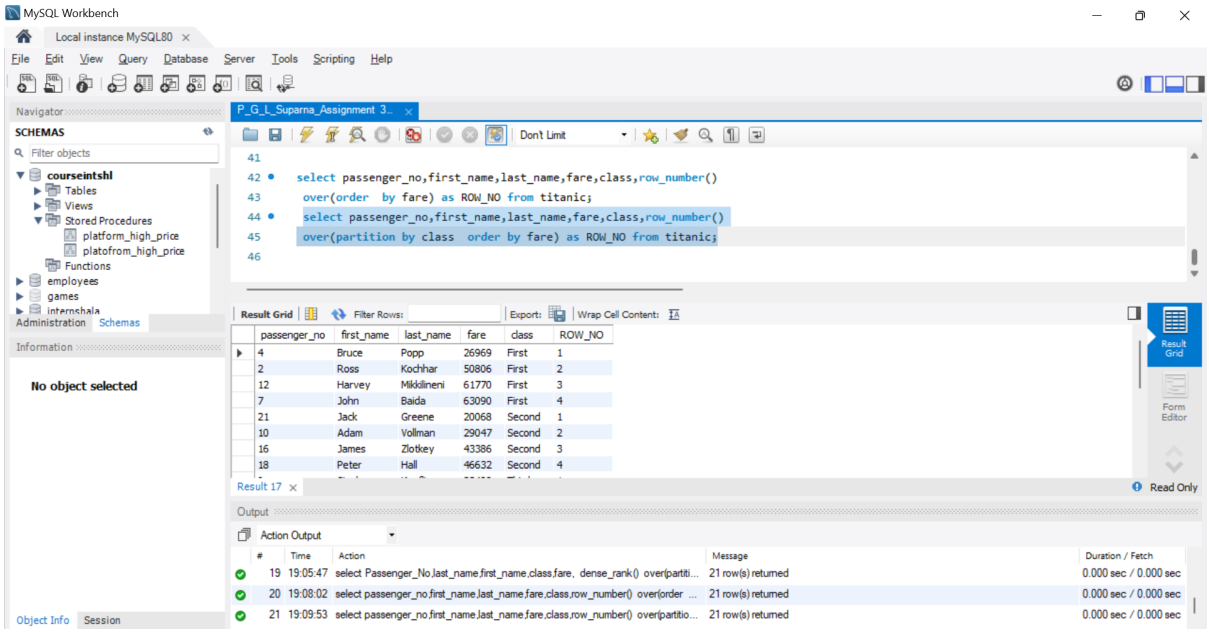


QUERY:

```
select Passenger_No,last_name,first_name,class,fare,
dense_rank()  over(partition by class order by fare desc)
as Dense_Fare_Rank from titanic;
```

Task 9: Assign row numbers to passengers based on the order of their fares:

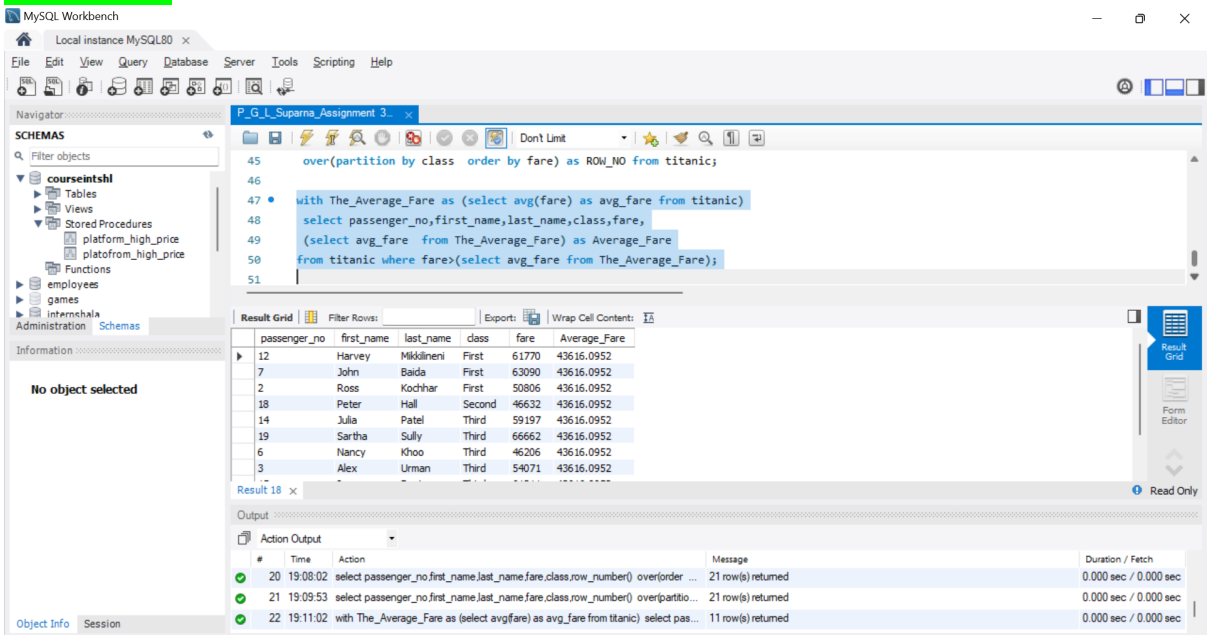




QUERY:

```
select passenger_no,first_name,last_name,fare,class,row_number()
 over(order  by fare) as ROW_NO from titanic;
select passenger_no,first_name,last_name,fare,class,row_number()
 over(partition by class  order by fare) as ROW_NO from titanic;
```

Task 10: **Use a CTE to calculate the average fare and find passengers who paid more than the Average:**



QUERY:

```
with The_Average_Fare as (select avg(fare) as avg_fare from titanic)
 select passenger_no,first_name,last_name,class,fare,
 (select avg_fare  from The_Average_Fare) as Average_Fare
from titanic where fare>(select avg_fare from The_Average_Fare);
```

Prepared By,

Utkarsh Anand