MySQL

Assignment 3rd

Project Objective: The objective is to perform advanced data analysis and reporting on the Titanic passengers dataset using key MySQL concepts, such as subqueries, views, stored procedures, CTE(Common Table Expressions), and window functions like LEAD, LAG, RANK, and DENSE_RANK.

Task 1st: To get the name and age of the oldest passenger who survived, I have used the following query:

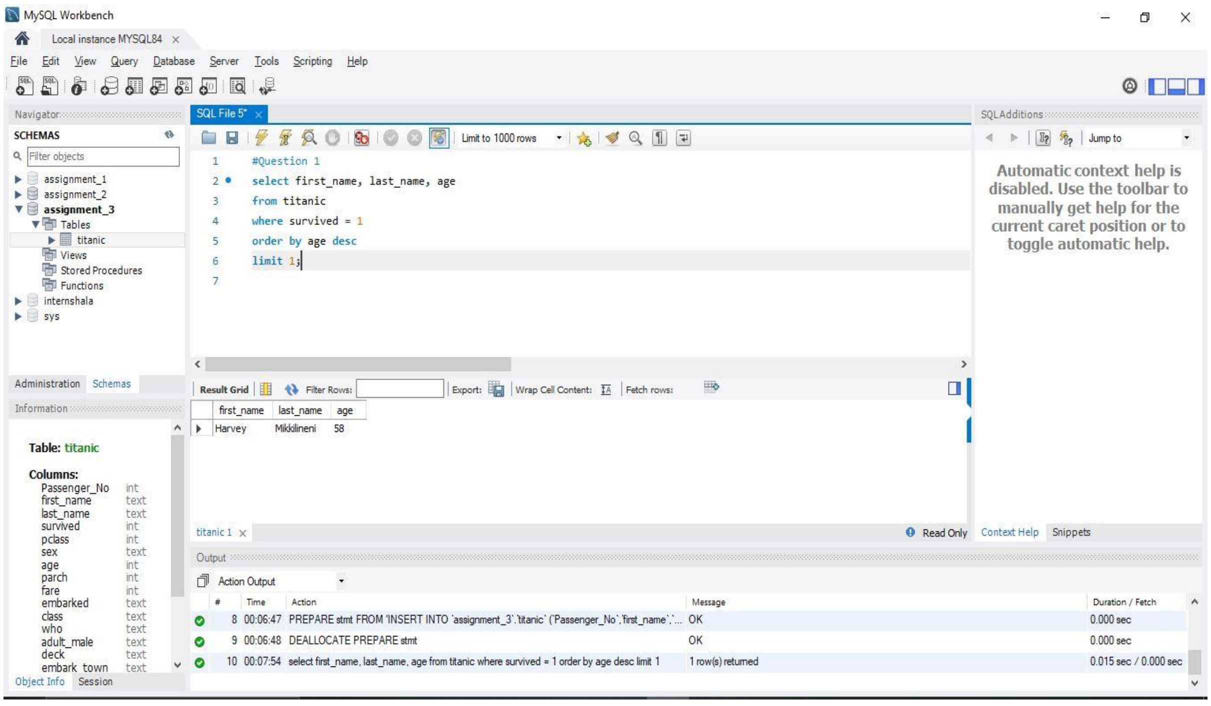
select first_name, last_name, age

from titanic

where survived = 1

order by age desc

limit 1;



```
#Question 1
select first_name, last_name, age
from titanic
where survived = 1
order by age desc
limit 1;
```

first_name	last_name	age
Harvey	Mikkilineni	58

```
#   Time      Action                                                                          Message            Duration / Fetch
8   00:06:47  PREPARE stmt FROM 'INSERT INTO `assignment_3`.`titanic` (`Passenger_No`,`first_name`,`...  OK   0.000 sec
9   00:06:48  DEALLOCATE PREPARE stmt                                                         OK                 0.000 sec
10  00:07:54  select first_name, last_name, age from titanic where survived = 1 order by age desc limit 1   1 row(s) returned   0.015 sec / 0.000 sec
```

Task 2nd : To create a view that displays the survival status, class, age, and fare, I have used the following SQL statement:

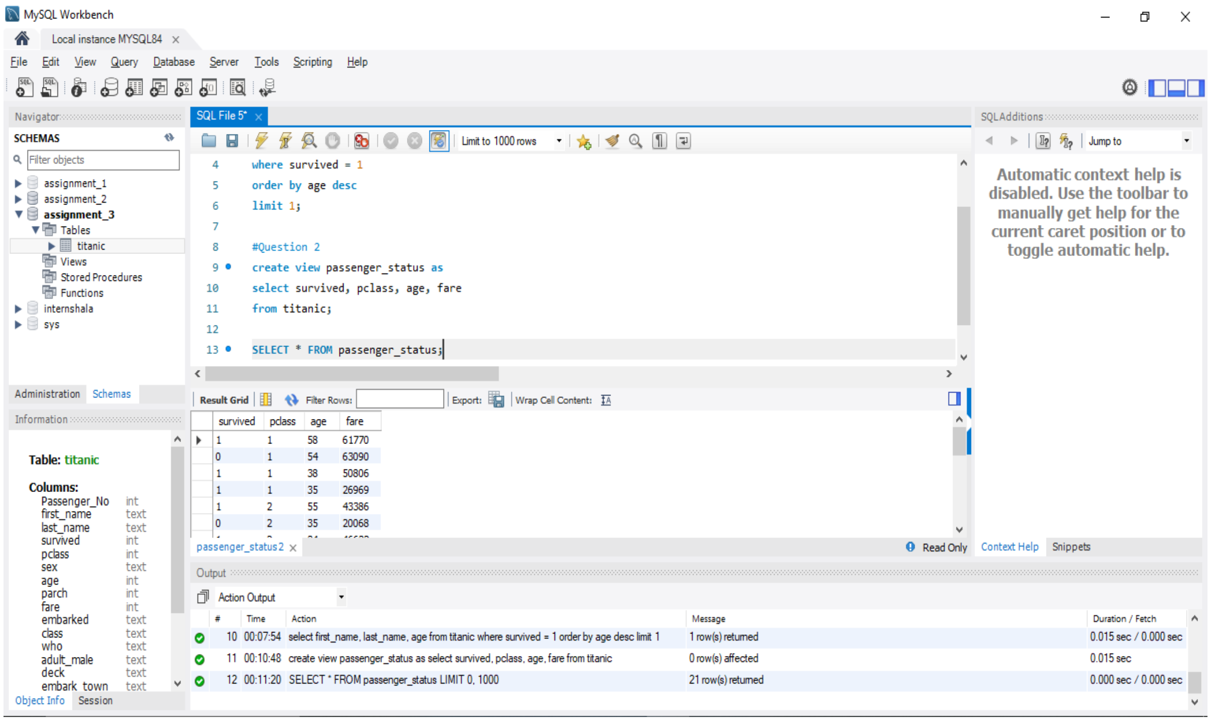
create view passenger_status as

select survived, pclass, age, fare

from titanic;

after that you have to display by using this query

select * from passenger_status



```
where survived = 1
order by age desc
limit 1;

#Question 2
create view passenger_status as
select survived, pclass, age, fare
from titanic;

SELECT * FROM passenger_status;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

survived	pclass	age	fare
1	1	58	61770
0	1	54	63090
1	1	38	50806
1	1	35	26969
1	2	55	43386
0	2	35	20068

Table: titanic

Columns:

Passenger_No	int
first_name	text
last_name	text
survived	int
pclass	int
sex	text
age	int
parch	int
fare	int
embarked	text
class	text
who	text
adult_male	text
deck	text
embark_town	text

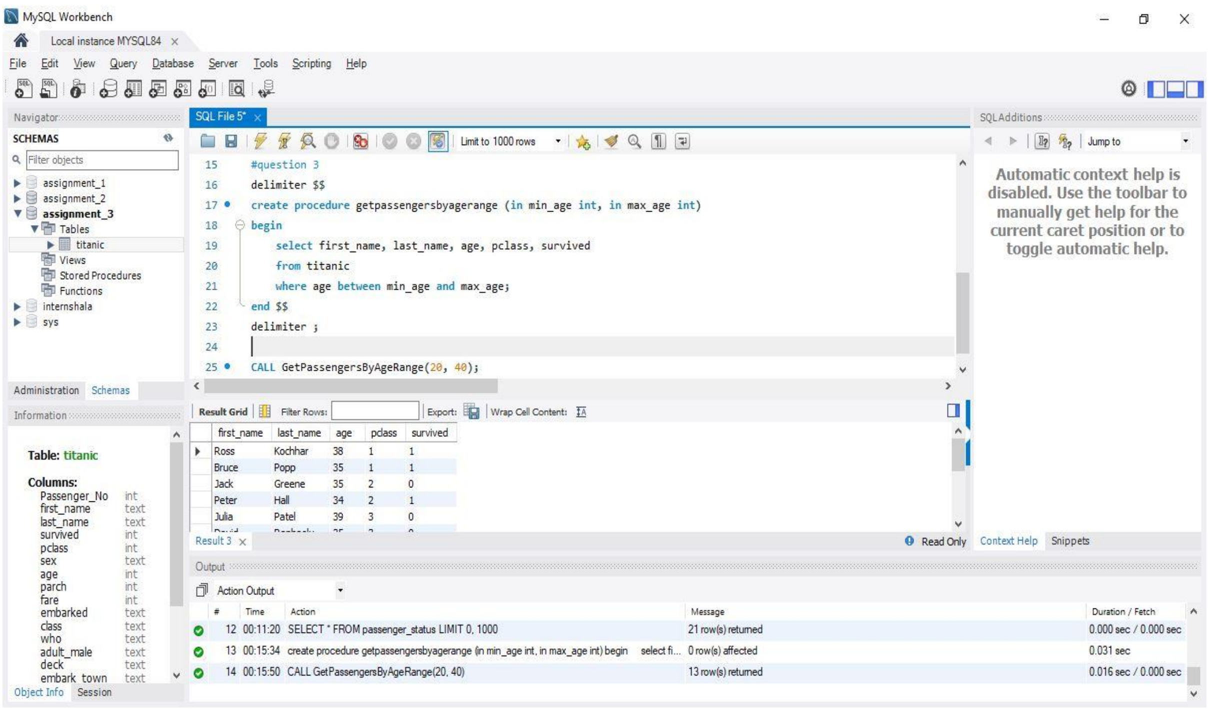
```
#   Time      Action                                                                            Message             Duration / Fetch
10  00:07:54  select first_name, last_name, age from titanic where survived = 1 order by age desc limit 1   1 row(s) returned   0.015 sec / 0.000 sec
11  00:10:48  create view passenger_status as select survived, pclass, age, fare from titanic             0 row(s) affected   0.015 sec
12  00:11:20  SELECT * FROM passenger_status LIMIT 0, 1000                                                21 row(s) returned  0.000 sec / 0.000 sec
```

Task 3rd : To create a stored procedure that retrieves passengers based on a given age range, you can use the following SQL:

```
delimiter $$
create procedure getpassengersbyagerange (in min_age int, in max_age int)
begin
    select first_name, last_name, age, pclass, survived
    from titanic
    where age between min_age and max_age;
end $$
delimiter ;
```

To call this stored procedure with specific values for the age range:

```
CALL GetPassengersByAgeRange(20, 40);
```



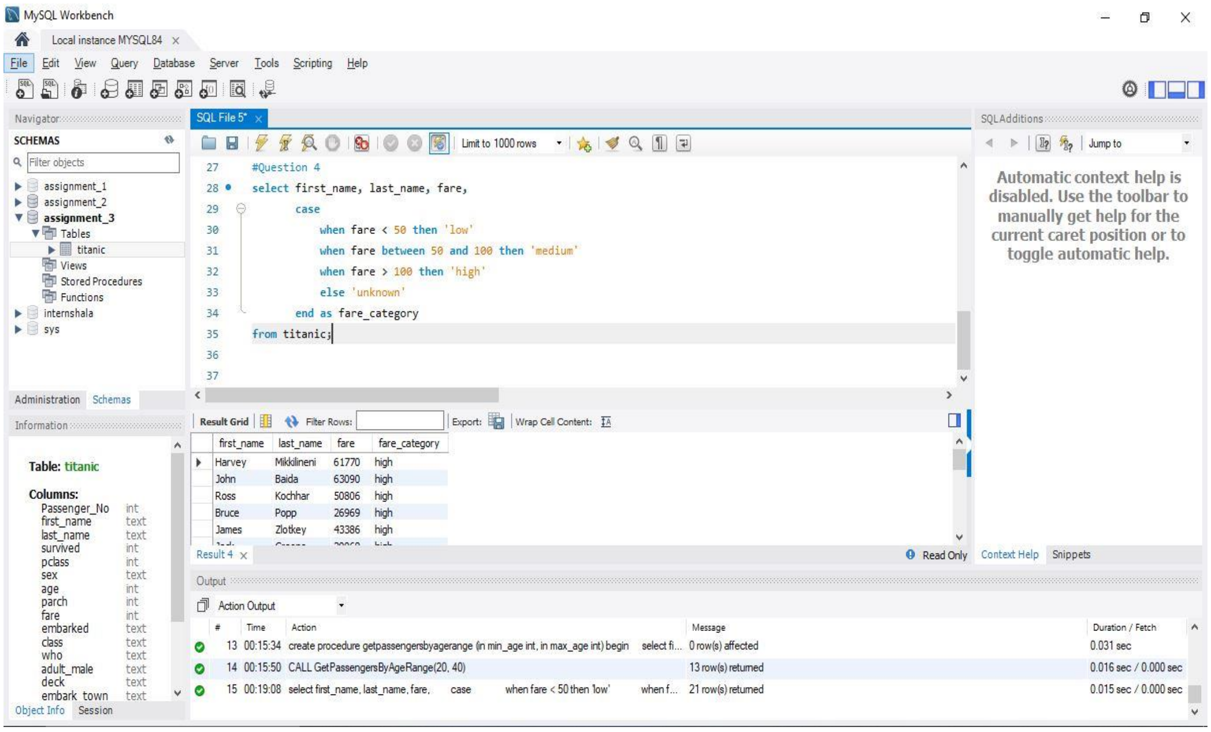
```
#question 3
delimiter $$
create procedure getpassengersbyagerange (in min_age int, in max_age int)
begin
    select first_name, last_name, age, pclass, survived
    from titanic
    where age between min_age and max_age;
end $$
delimiter ;

CALL GetPassengersByAgeRange(20, 40);
```

first_name	last_name	age	pclass	survived
Ross	Kochhar	38	1	1
Bruce	Popp	35	1	1
Jack	Greene	35	2	0
Peter	Hall	34	2	1
Julia	Patel	39	3	0

```
#   Time      Action                                                                                   Message             Duration / Fetch
12  00:11:20  SELECT * FROM passenger_status LIMIT 0, 1000                                             21 row(s) returned  0.000 sec / 0.000 sec
13  00:15:34  create procedure getpassengersbyagerange (in min_age int, in max_age int) begin select fi...  0 row(s) affected   0.031 sec
14  00:15:50  CALL GetPassengersByAgeRange(20, 40)                                                     13 row(s) returned  0.016 sec / 0.000 sec
```

Task 4th : I have categorized passengers based on their fare by using the CASE statement. Here's the query to classify passengers into fare categories:

```
select first_name, last_name, fare,
       case
           when fare < 50 then 'low'
           when fare between 50 and 100 then 'medium'
           when fare > 100 then 'high'
           else 'unknown'
       end as fare_category
from titanic;
```



```
#Question 4
select first_name, last_name, fare,
       case
           when fare < 50 then 'low'
           when fare between 50 and 100 then 'medium'
           when fare > 100 then 'high'
           else 'unknown'
       end as fare_category
from titanic;
```

first_name	last_name	fare	fare_category
Harvey	Mikkilineni	61770	high
John	Baida	63090	high
Ross	Kochhar	50806	high
Bruce	Popp	26969	high
James	Zlotkey	43386	high

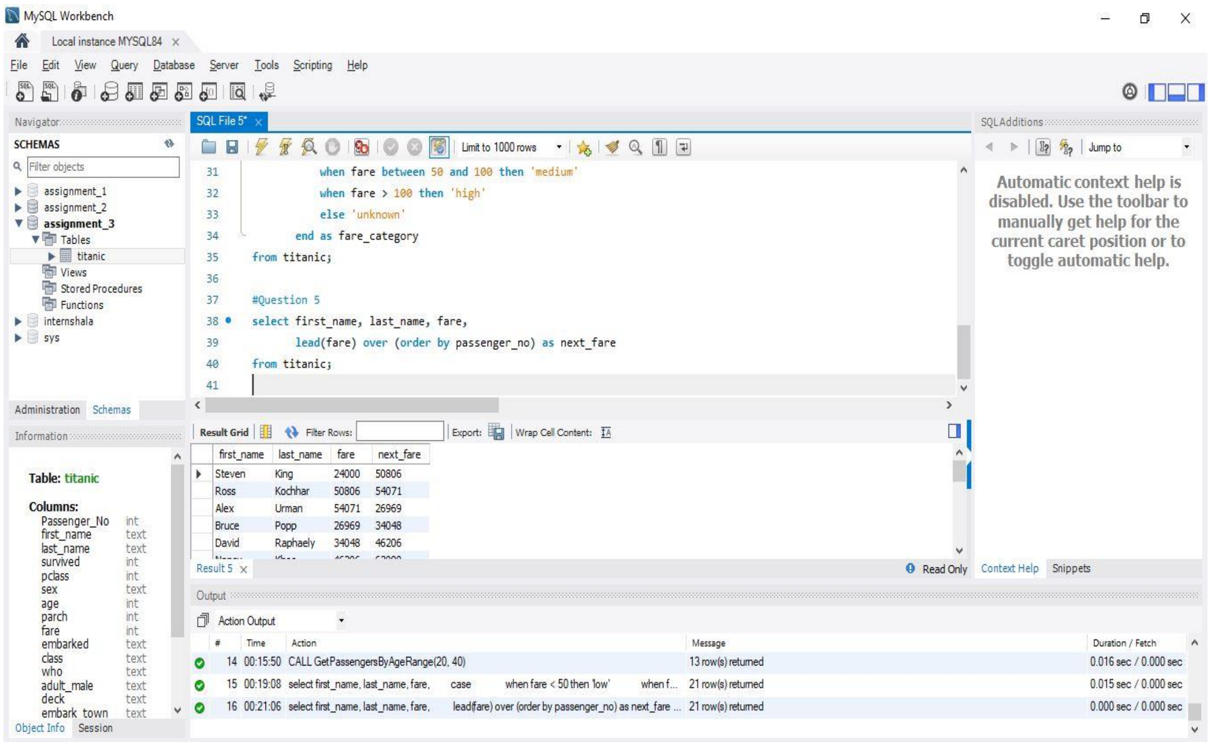
```
#   Time      Action                                                                              Message              Duration / Fetch
13  00:15:34  create procedure getpassengersbyagerange (in min_age int, in max_age int) begin  select fi...  0 row(s) affected    0.031 sec
14  00:15:50  CALL GetPassengersByAgeRange(20, 40)                                               13 row(s) returned   0.016 sec / 0.000 sec
15  00:19:08  select first_name, last_name, fare,    case    when fare < 50 then 'low'    when f...  21 row(s) returned   0.015 sec / 0.000 sec
```

Task 5th : To show the fare of the next passenger, you can use the LEAD() window function. This function provides access to the value of the next row within the same result set.

select first_name, last_name, fare,

lead(fare) over (order by passenger_no) as next_fare

from titanic;



```
            when fare between 50 and 100 then 'medium'
            when fare > 100 then 'high'
            else 'unknown'
        end as fare_category
    from titanic;

    #Question 5
    select first_name, last_name, fare,
        lead(fare) over (order by passenger_no) as next_fare
    from titanic;
```

first_name	last_name	fare	next_fare
Steven	King	24000	50806
Ross	Kochhar	50806	54071
Alex	Urman	54071	26969
Bruce	Popp	26969	34048
David	Raphaely	34048	46206

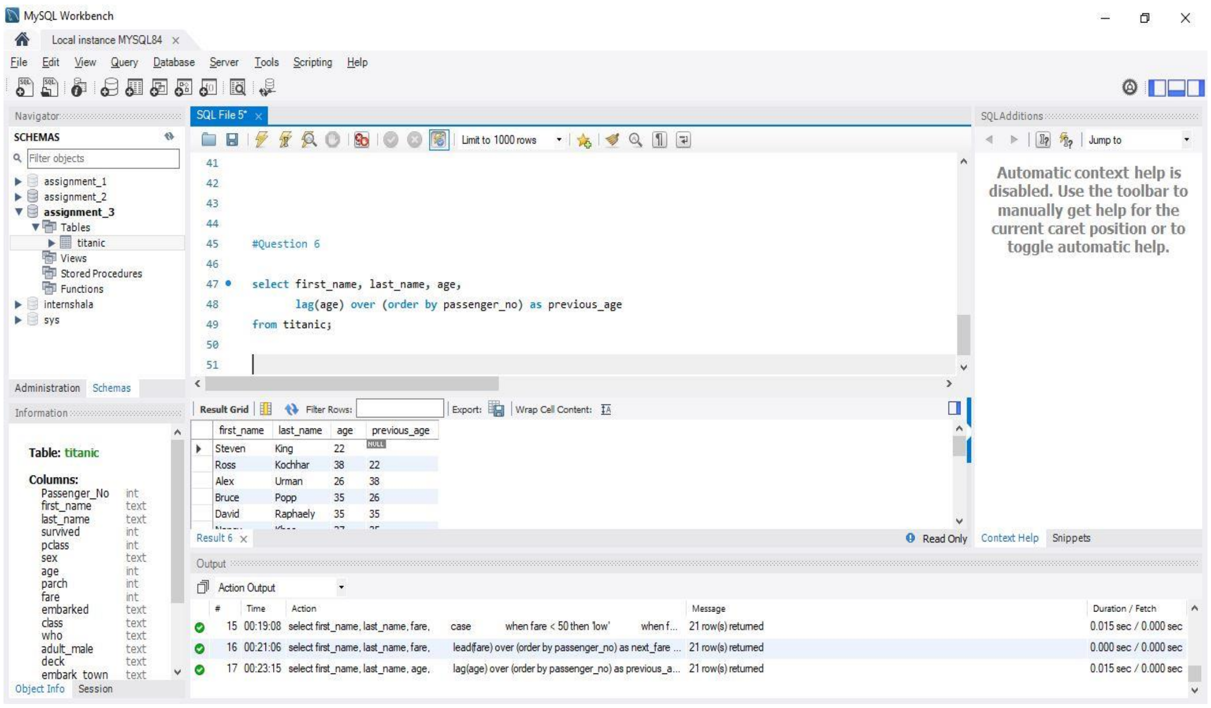
```
#   Time      Action                                                                        Message             Duration / Fetch
14  00:15:50  CALL GetPassengersByAgeRange(20, 40)                                          13 row(s) returned  0.016 sec / 0.000 sec
15  00:19:08  select first_name, last_name, fare,   case   when fare < 50 then 'low'   when f...   21 row(s) returned  0.015 sec / 0.000 sec
16  00:21:06  select first_name, last_name, fare,   lead(fare) over (order by passenger_no) as next_fare ...   21 row(s) returned  0.000 sec / 0.000 sec
```

Task 6th: To get the age of the previous passenger, you can use the LAG() window function.This function retrieves the value from the previous row within the result set.

select first_name, last_name, age,

lag(age) over (order by passenger_no) as previous_age

from titanic;



```
#Question 6

select first_name, last_name, age,
       lag(age) over (order by passenger_no) as previous_age
from titanic;
```

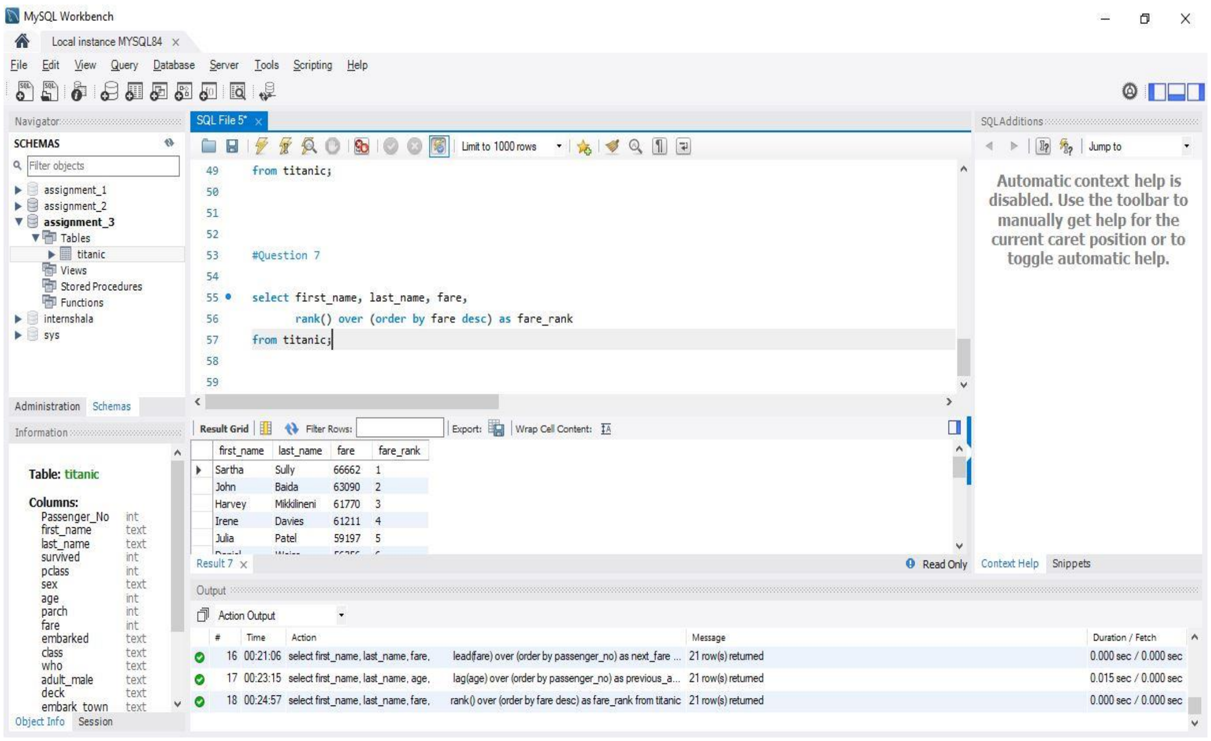
first_name	last_name	age	previous_age
Steven	King	22	NULL
Ross	Kochhar	38	22
Alex	Urman	26	38
Bruce	Popp	35	26
David	Raphaely	35	35

```
#   Time      Action                                                                                Message             Duration / Fetch
15  00:19:08  select first_name, last_name, fare,    case    when fare < 50 then 'low'    when f...   21 row(s) returned  0.015 sec / 0.000 sec
16  00:21:06  select first_name, last_name, fare,    lead(fare) over (order by passenger_no) as next_fare ...  21 row(s) returned  0.000 sec / 0.000 sec
17  00:23:15  select first_name, last_name, age,    lag(age) over (order by passenger_no) as previous_a...  21 row(s) returned  0.015 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Task 7th : To rank passengers based on their fare, you can use the RANK() window function. This will rank the passengers in descending order of their fare.

```
select first_name, last_name, fare,
      rank() over (order by fare desc) as fare_rank
from titanic;
```



```
from titanic;



#Question 7

select first_name, last_name, fare,
      rank() over (order by fare desc) as fare_rank
from titanic;
```

first_name	last_name	fare	fare_rank
Sartha	Sully	66662	1
John	Baida	63090	2
Harvey	Mikkilineni	61770	3
Irene	Davies	61211	4
Julia	Patel	59197	5

Table: titanic

Columns:

Passenger_No	int
first_name	text
last_name	text
survived	int
pclass	int
sex	text
age	int
parch	int
fare	int
embarked	text
class	text
who	text
adult_male	text
deck	text
embark_town	text

```
#   Time      Action                                                                                Message             Duration / Fetch
16  00:21:06  select first_name, last_name, fare,    lead(fare) over (order by passenger_no) as next_fare ...   21 row(s) returned  0.000 sec / 0.000 sec
17  00:23:15  select first_name, last_name, age,     lag(age) over (order by passenger_no) as previous_a...     21 row(s) returned  0.015 sec / 0.000 sec
18  00:24:57  select first_name, last_name, fare,    rank() over (order by fare desc) as fare_rank from titanic 21 row(s) returned  0.000 sec / 0.000 sec
```

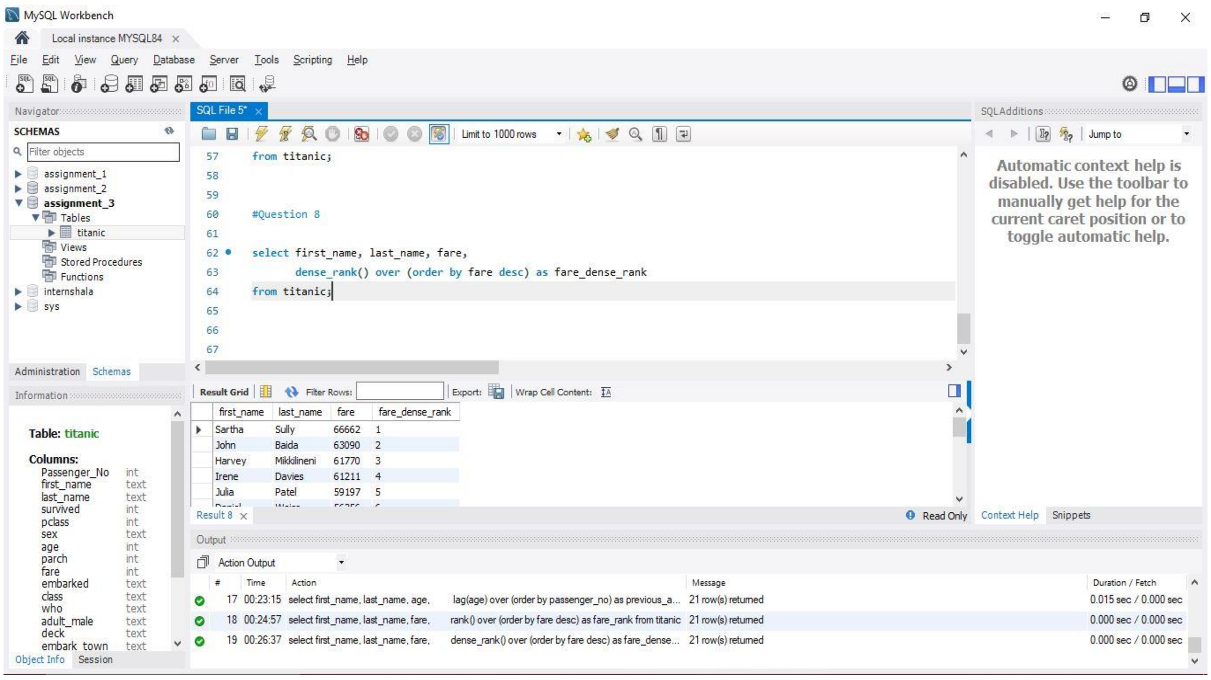
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Task 8th : To rank passengers with no gaps (i.e., if two passengers share the same fare, they receive the same rank, but the next rank is sequential), you can use the DENSE_RANK() window function. This ensures that ranks are consecutive, without gaps.

select first_name, last_name, fare,

dense_rank() over (order by fare desc) as fare_dense_rank

from titanic;



```
from titanic;


#Question 8

select first_name, last_name, fare,
       dense_rank() over (order by fare desc) as fare_dense_rank
from titanic;
```

	first_name	last_name	fare	fare_dense_rank
▶	Sartha	Sully	66662	1
	John	Baida	63090	2
	Harvey	Mikkilineni	61770	3
	Irene	Davies	61211	4
	Julia	Patel	59197	5

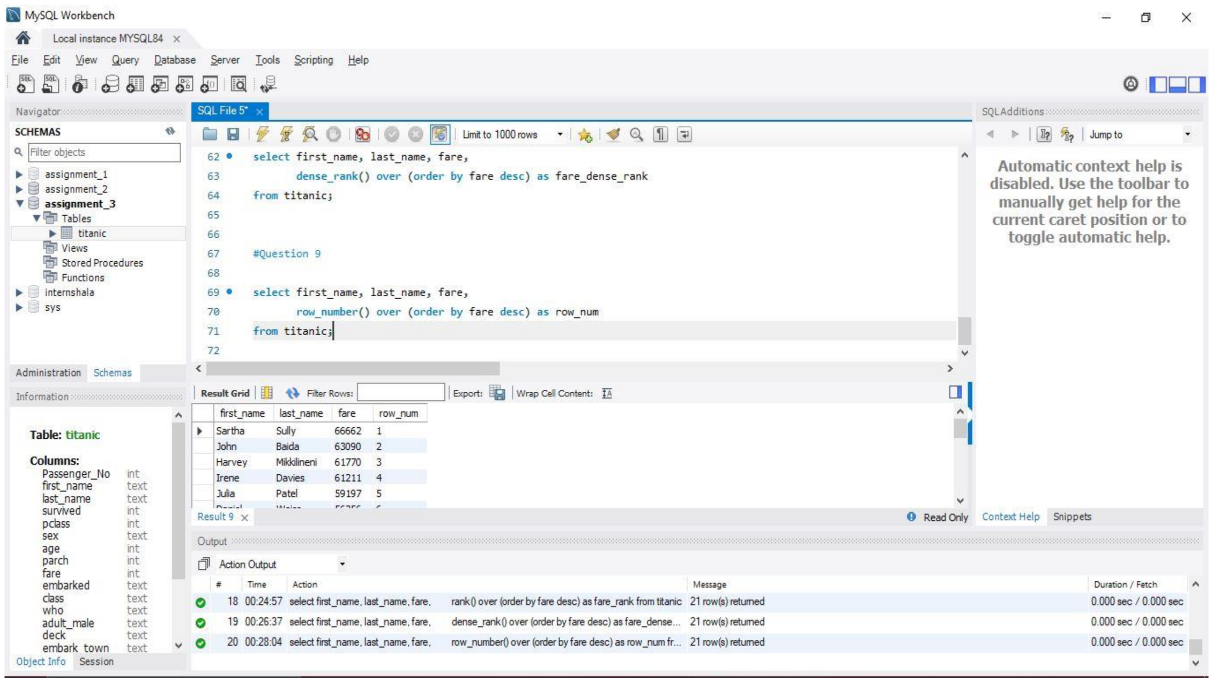
```
#   Time      Action                                                                               Message             Duration / Fetch
17  00:23:15  select first_name, last_name, age,  lag(age) over (order by passenger_no) as previous_a...  21 row(s) returned  0.015 sec / 0.000 sec
18  00:24:57  select first_name, last_name, fare,  rank() over (order by fare desc) as fare_rank from titanic  21 row(s) returned  0.000 sec / 0.000 sec
19  00:26:37  select first_name, last_name, fare,  dense_rank() over (order by fare desc) as fare_dense...  21 row(s) returned  0.000 sec / 0.000 sec
```

Task 9th : To assign row numbers to passengers based on their fare, you can use the ROW_NUMBER() window function. This will give each passenger a unique row number based on the ordering of their fare.

select first_name, last_name, fare,

row_number() over (order by fare desc) as row_num

from titanic;



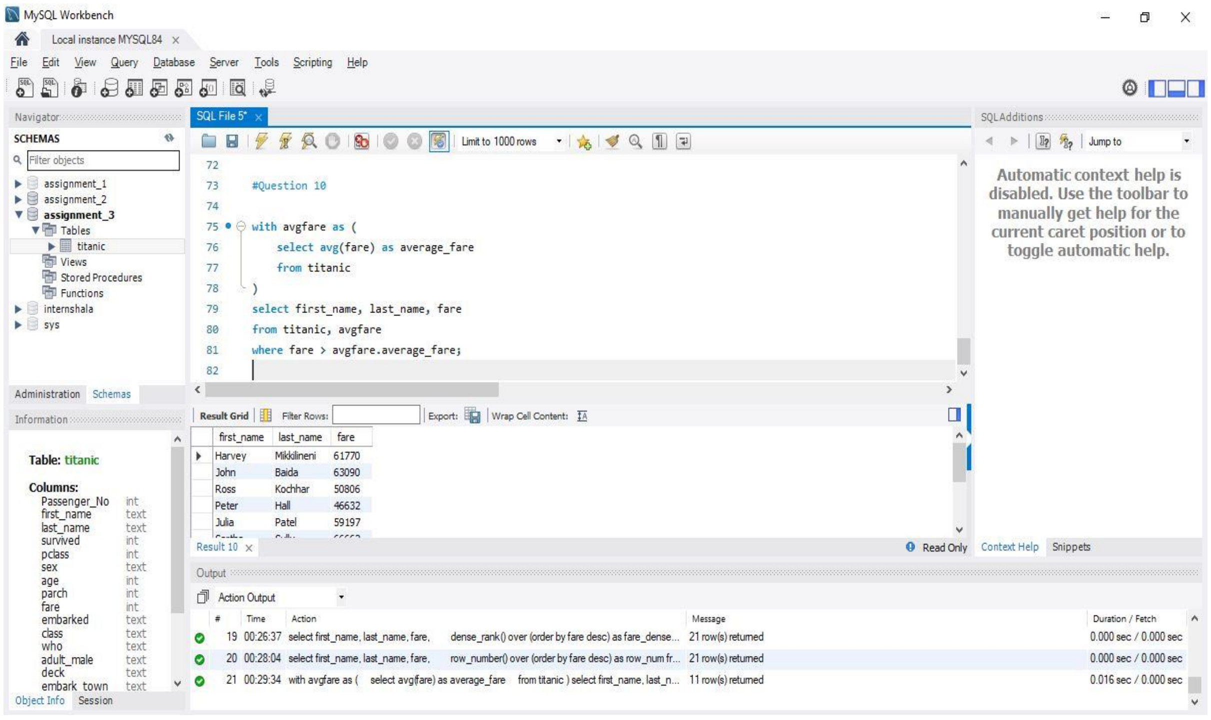
```
select first_name, last_name, fare,
        dense_rank() over (order by fare desc) as fare_dense_rank
from titanic;


#Question 9

select first_name, last_name, fare,
        row_number() over (order by fare desc) as row_num
from titanic;
```

first_name	last_name	fare	row_num
Sartha	Sully	66662	1
John	Baida	63090	2
Harvey	Mikkilineni	61770	3
Irene	Davies	61211	4
Julia	Patel	59197	5

#	Time	Action	Message	Duration / Fetch
18	00:24:57	select first_name, last_name, fare, rank() over (order by fare desc) as fare_rank from titanic	21 row(s) returned	0.000 sec / 0.000 sec
19	00:26:37	select first_name, last_name, fare, dense_rank() over (order by fare desc) as fare_dense...	21 row(s) returned	0.000 sec / 0.000 sec
20	00:28:04	select first_name, last_name, fare, row_number() over (order by fare desc) as row_num fr...	21 row(s) returned	0.000 sec / 0.000 sec

Task 10th : A Common Table Expression (CTE) can be used to first calculate the average fare, and then you can use that result to filter passengers who paid more than the average fare.

```
with avgfare as (
   select avg(fare) as average_fare
   from titanic
)
select first_name, last_name, fare
from titanic, avgfare
where fare > avgfare.average_fare;
```



```
#Question 10

with avgfare as (
    select avg(fare) as average_fare
    from titanic
)
select first_name, last_name, fare
from titanic, avgfare
where fare > avgfare.average_fare;
```

first_name	last_name	fare
Harvey	Mikkilineni	61770
John	Baida	63090
Ross	Kochhar	50806
Peter	Hall	46632
Julia	Patel	59197

```
#   Time      Action                                                                                      Message
19  00:26:37  select first_name, last_name, fare,    dense_rank() over (order by fare desc) as fare_dense...  21 row(s) returned   0.000 sec / 0.000 sec
20  00:28:04  select first_name, last_name, fare,    row_number() over (order by fare desc) as row_num fr...  21 row(s) returned   0.000 sec / 0.000 sec
21  00:29:34  with avgfare as (    select avg(fare) as average_fare    from titanic ) select first_name, last_n...  11 row(s) returned   0.016 sec / 0.000 sec
```