

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:

The screenshot shows the MySQL Workbench interface with a query editor and result grid. The query is:

```
1 • select * from titanic;
```

The result grid displays 21 rows of passenger data. The top row is highlighted in yellow. The columns are: Passenger_No, first_name, last_name, survived, pdass, sex, age, parch, fare, embarked, class, who, adult_male, deck, embark_town, alive, alone.

Passenger_No	first_name	last_name	survived	pdass	sex	age	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
12	Harvey	Mikkilineni	1	1	female	58	0	61770	S	First	woman	FALSE	C	Southampton	yes	TRUE
7	John	Baile	0	1	male	54	0	63090	S	First	man	TRUE	E	Southampton	no	TRUE
2	Rose	Kochhar	1	1	female	38	0	50805	C	First	woman	FALSE	C	Cherbourg	yes	FALSE
4	Bruce	Popp	1	1	female	35	0	26569	S	First	woman	FALSE	C	Southampton	yes	FALSE
16	James	Zlotkey	1	2	female	55	0	43386	S	Second	woman	FALSE	D	Southampton	yes	TRUE
21	Jack	Greene	0	2	male	35	0	20068	S	Second	man	TRUE	G	Southampton	no	TRUE
19	Dexter	Hall	1	2	male	34	0	34627	C	Second	man	TRUE	B	Southampton	yes	TRUE

The output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	18:31:05	select * from titanic	21 row(s) returned	0.016 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface with a query editor and result grid. The queries are:

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

The result grid displays 1 row of data. The columns are: first_name, last_name, age.

first_name	last_name	age
Harvey	Mikkilineni	58

The output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	18:31:05	select * from titanic	21 row(s) returned	0.016 sec / 0.000 sec
2	18:31:47	select first_name, last_name, age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;	1 row(s) returned	0.000 sec / 0.000 sec

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:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema for 'courseinsthi'. Under 'Views', there are several views like 'highest_price', 'highest_prices', etc., and one view named 'Survived' which is currently selected.
- Query Editor:** Displays the SQL code used to create the view:

```
1 • select * from titanic;
2 • select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;
3 • create view Survived as
4 • select survived as survival_status,class,age,fare from titanic;
5 • select * from Survived;
```
- Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
6	18:35:51	DROP VIEW 'courseinsthi'.`survived`	0 row(s) affected	0.031 sec
7	18:36:05	select * from titanic	21 row(s) returned	0.000 sec / 0.000 sec
8	18:36:11	select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;	1 row(s) returned	0.000 sec / 0.000 sec
9	18:36:24	create view Survived as select survived as survival_status,class,age,fare from titanic;	0 row(s) affected	0.000 sec
10	18:36:47	select * from Survived	21 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema for 'courseinsthi'. Under 'Views', there are several views like 'highest_price', 'highest_prices', etc., and one view named 'Survived'.
- Query Editor:** Displays the SQL code used to create the view:

```
1 • select * from titanic;
2 • select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;
3 • create view Survived as
4 • select survived as survival_status,class,age,fare from titanic;
5 • select * from Survived;
```
- Result Grid:** Shows the data returned by the query 'select * from Survived;'. The data is as follows:

	survival_status	class	age	fare
1	First	58	61770	
0	First	54	63090	
1	First	38	50806	
1	First	35	26969	
1	Second	55	43386	
0	Second	35	20068	
1	Second	34	46632	
1	Second	14	29047	
- Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
8	18:36:11	select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;	1 row(s) returned	0.000 sec / 0.000 sec
9	18:36:24	create view Survived as select survived as survival_status,class,age,fare from titanic;	0 row(s) affected	0.000 sec
10	18:36:47	select * from Survived	21 row(s) returned	0.000 sec / 0.000 sec

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:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema 'courseintsh1' with objects like Tables, Views, Stored Procedures, Functions, employees, games, and Internshala.
- SQL Editor:** Contains the following SQL code:

```

1 • select * from titanic;
2 • select first_name,last_name,age from titanic where age=(select max(age) from titanic where survived = 1) and survived = 1;
3 • create view Survived as
4 select survived as survival_status,class,age,fare from titanic;
5 • select * from Survived;
6 delimiter //
7 • create procedure Passanger_details_based_age(in min_age int,in max_age int)
8 begin
9 select * from titanic where age between min_age and max_age;
10 end //
11 delimiter ;
12
13

```
- Output:** Shows the execution results:

#	Time	Action	Message	Duration / Fetch
11	18:39:06	DROP PROCEDURE 'courseintsh1'.'Passanger_details_based_age'	0 row(s) affected	0.047 sec
12	18:39:15	select * from Survived	21 row(s) returned	0.000 sec / 0.000 sec
13	18:39:26	create procedure Passanger_details_based_age(in min_age int,in max_age int) b...	0 row(s) affected	0.000 sec

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema 'courseintsh1' with objects like Tables, Views, Stored Procedures, Functions, employees, games, and Internshala.
- SQL Editor:** Contains the following SQL code:

```

7 • create procedure Passanger_details_based_age(in min_age int,in max_age int)
8 begin
9 select * from titanic where age between min_age and max_age;
10 end //
11 delimiter ;
12 • call Passanger_details_based_age(20,30);
13

```
- Result Grid:** Displays the results of the query:

Passenger_No	first_name	last_name	survived	pclass	sex	age	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
20	William	Smith	1	3	female	29	0	38628	C	Third	woman	FALSE	D	Cherbourg	yes	TRUE
6	Nancy	Khoob	0	3	male	27	0	46206	Q	Third	man	TRUE	E	Queenstown	no	TRUE
9	Sigal	Kaufling	1	3	female	27	2	23488	S	Third	woman	FALSE	A	Southampton	yes	FALSE
3	Alex	Urman	1	3	female	26	0	54071	S	Third	woman	FALSE	B	Southampton	yes	TRUE
1	Steven	King	0	3	male	22	0	24000	S	Third	man	TRUE	A	Southampton	no	FALSE
13	Kevin	Rogers	0	3	male	20	0	30897	S	Third	man	TRUE	C	Southampton	no	TRUE
- Output:** Shows the execution results:

#	Time	Action	Message	Duration / Fetch
12	18:39:15	select * from Survived	21 row(s) returned	0.000 sec / 0.000 sec
13	18:39:26	create procedure Passanger_details_based_age(in min_age int,in max_age int) b...	0 row(s) affected	0.000 sec
14	18:40:06	call Passanger_details_based_age(20,30)	6 row(s) returned	0.000 sec / 0.000 sec

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':

The screenshot shows the MySQL Workbench interface with a query editor window titled 'P_G_L_Suparna_Assignment 3...'. The code entered is:

```
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;
```

The results grid displays 11 rows of data from the 'titanic' table, with an additional row labeled 'Result 11'. The columns are: passenger_no, first_name, last_name, age, fare, and Price. The 'Price' column contains the categorized fare values: High, High, Medium, Medium, High, High, Low, High, Medium, Medium.

passenger_no	first_name	last_name	age	fare	Price
12	Harvey	Mikkilineni	58	61770	High
7	John	Baida	54	63090	High
2	Rose	Kochhar	38	50806	High
4	Bruce	Popp	35	26969	Medium
16	James	Zlotkey	55	43386	High
21	Jack	Greene	35	20068	Low
18	Peter	Hall	34	46632	High
10	Adam	Vollman	14	29047	Medium
...
Result 11					

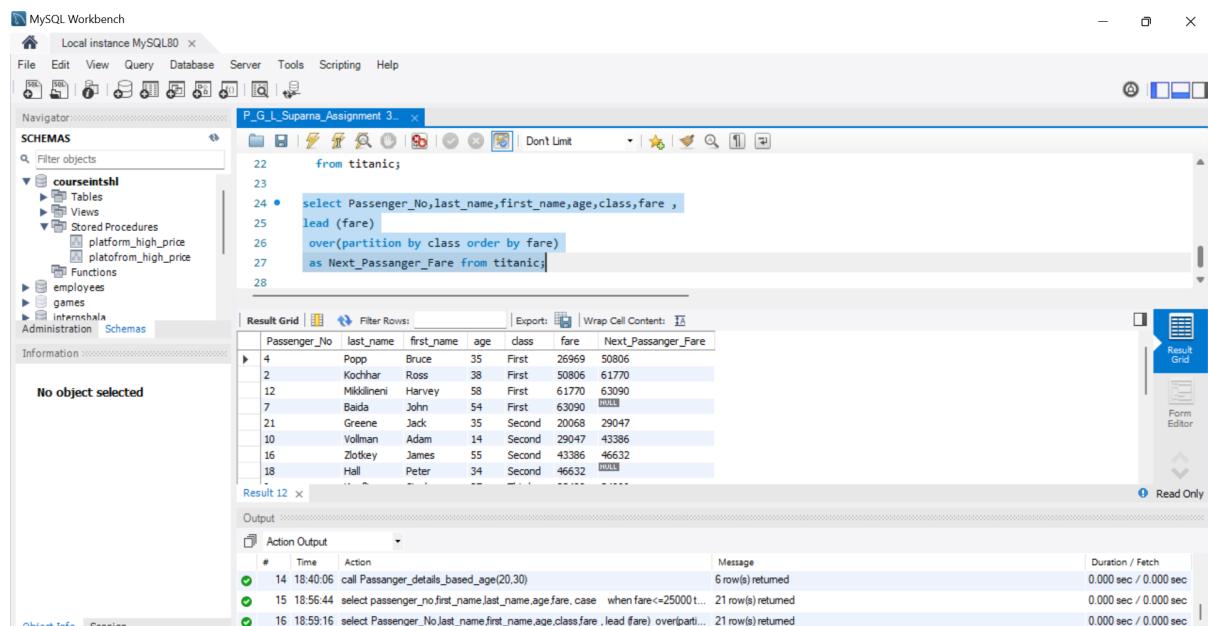
The 'Output' pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
13	18:39:26	create procedure Passanger_details_based_age(in min_age int,in max_age int) b...	0 row(s) affected	0.000 sec
14	18:40:06	call Passanger_details_based_age(20,30)	6 row(s) returned	0.000 sec / 0.000 sec
15	18:56:44	select passenger_no,first_name,last_name,age,fare,case when fare<=25000 t...	21 row(s) returned	0.000 sec / 0.000 sec

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:



The screenshot shows the MySQL Workbench interface with a query editor window titled "P_G_L_Suparna_Assignment 3...". The code entered is:

```
from titanic;
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;
```

The Result Grid displays the following data:

Passenger_No	last_name	first_name	age	class	fare	Next_Passanger_Fare
4	Popp	Bruce	35	First	26969	50806
2	Kochhar	Ross	38	First	50806	61770
12	Mikkilineni	Harvey	58	First	61770	63090
7	Baida	John	54	First	63090	HULL
21	Greene	Jack	35	Second	20068	29047
10	Vollman	Adam	14	Second	29047	43386
16	Zlotkey	James	55	Second	43386	46632
18	Hall	Peter	34	Second	46632	HULL

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
14	18:40:06	call Passenger_details_based_age(20,30)	6 row(s) returned	0.000 sec / 0.000 sec
15	18:56:44	select passenger_no,first_name,last_name,age,fare,class when fare<=25000 t...	21 row(s) returned	0.000 sec / 0.000 sec
16	18:59:16	select Passenger_No, last_name, first_name, age, class, fare , lead (fare) over(partition by class order by fare)	21 row(s) returned	0.000 sec / 0.000 sec

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:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `coursetest1` with objects like `Tables`, `Views`, and `Stored Procedures`.
- Query Editor:** Contains the following SQL query:

```
27      as Next_Passanger_Age from titanic;
28
29 • select Passenger_No, last_name, first_name, age, class,
30       lag(age)
31   over(partition by class order by age)
32      as Previous_Passenger_Age from titanic;
33
```
- Result Grid:** Displays the results of the query in a table format. The columns are `Passenger_No`, `last_name`, `first_name`, `age`, `class`, and `Previous_Passenger_Age`. The data includes rows for passengers like Popp, Kochhar, Baida, Mikkilineni, Vollman, Hall, Greene, and Zlotkey.
- Output:** Shows the execution log with three entries, all marked as successful (green checkmarks). The log includes the time, action, message, and duration.

QUERY:

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

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

The screenshot shows the MySQL Workbench interface with a query editor window titled 'P_G_L_Suparna_Assignment 3...'. The code entered is:

```
29 •    select Passenger_No, last_name, first_name, class,
30      lag(age)
31      over(partition by class order by age)
32      as Previous_Passenger_Age from titanic;
33
34 •    select Passenger_No, last_name, first_name, class, fare,
35      rank() over(partition by class order by fare asc)
```

The results grid displays 14 rows of data:

Passenger_No	last_name	first_name	class	fare	Fare_Rank
4	Popp	Bruce	First	26969	1
2	Kochhar	Ross	First	50806	2
12	Mikkilineni	Harvey	First	61770	3
7	Baida	John	First	63090	4
21	Greene	Jack	Second	20068	1
10	Vollman	Adam	Second	29047	2
16	Zlotkey	James	Second	43386	3
18	Hall	Peter	Second	46632	4
...

The 'Action Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
16	18:59:16	select Passenger_No, last_name, first_name, class, fare, rank() over(partition by class order by fare asc) as Fare_Rank from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
17	19:01:04	select Passenger_No, last_name, first_name, age, class, lag(age) over(partition by class order by age asc) as Previous_Passenger_Age from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
18	19:03:22	select Passenger_No, last_name, first_name, class, fare, rank() over(partition by class order by fare asc) as Fare_Rank from titanic;	21 row(s) returned	0.000 sec / 0.000 sec

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:

The screenshot shows the MySQL Workbench interface with a query editor window titled "P_G_L_Suparna_Assignment_3...". The code entered is:

```
as Fare_Rank from titanic;
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;
```

The "Result Grid" pane displays the following data:

Passenger_No	last_name	first_name	class	fare	Dense_Fare_Rank
7	Baida	John	First	63090	1
12	Mikkilineni	Harvey	First	61770	2
2	Kochhar	Ross	First	50806	3
4	Popp	Bruce	First	26969	4
18	Hall	Peter	Second	46632	1
16	Zlotkey	James	Second	43386	2
10	Vollman	Adam	Second	29047	3
21	Greene	Jack	Second	20068	4

The "Output" pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
17	19:01:04	select Passenger_No, last_name, first_name, class, fare,	21 row(s) returned	0.000 sec / 0.000 sec
18	19:03:22	dense_rank() over(partition by class order by fare desc)	21 row(s) returned	0.000 sec / 0.000 sec
19	19:05:47	as Dense_Fare_Rank from titanic;	21 row(s) returned	0.000 sec / 0.000 sec

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:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: P_G_L_Suparna_Assignment 3...

SCHEMAS: courseintsh1

Result Grid:

passenger_no	first_name	last_name	fare	class	ROW_NO
21	Jack	Greene	20068	Second	1
9	Sigal	Kaufing	23488	Third	2
1	Steven	King	24000	Third	3
4	Bruce	Popp	26969	First	4
17	Jason	Bernstein	27978	Third	5
10	Adam	Volman	29047	Second	6
13	Kevin	Rogers	30897	Third	7
5	David	Raphaely	34048	Third	8

Output:

#	Time	Action	Message	Duration / Fetch
18	19:03:22	select Passenger_No, last_name, first_name, class, fare, rank() over(partition by class order by fare desc) as Dense_Fare_Rank from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
19	19:05:47	select passenger_no, first_name, last_name, fare, class, row_number() over(order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
20	19:08:02	select passenger_no, first_name, last_name, fare, class, row_number() over(order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: P_G_L_Suparna_Assignment 3...

SCHEMAS: courseintsh1

Result Grid:

passenger_no	first_name	last_name	fare	class	ROW_NO
4	Bruce	Popp	26969	First	1
2	Ross	Kochhar	50806	First	2
12	Harvey	Mikkilineni	61770	First	3
7	John	Bada	63090	First	4
21	Jack	Green	20068	Second	1
10	Adam	Volman	29047	Second	2
16	James	Zlotkey	43386	Second	3
18	Peter	Hall	46632	Second	4

Output:

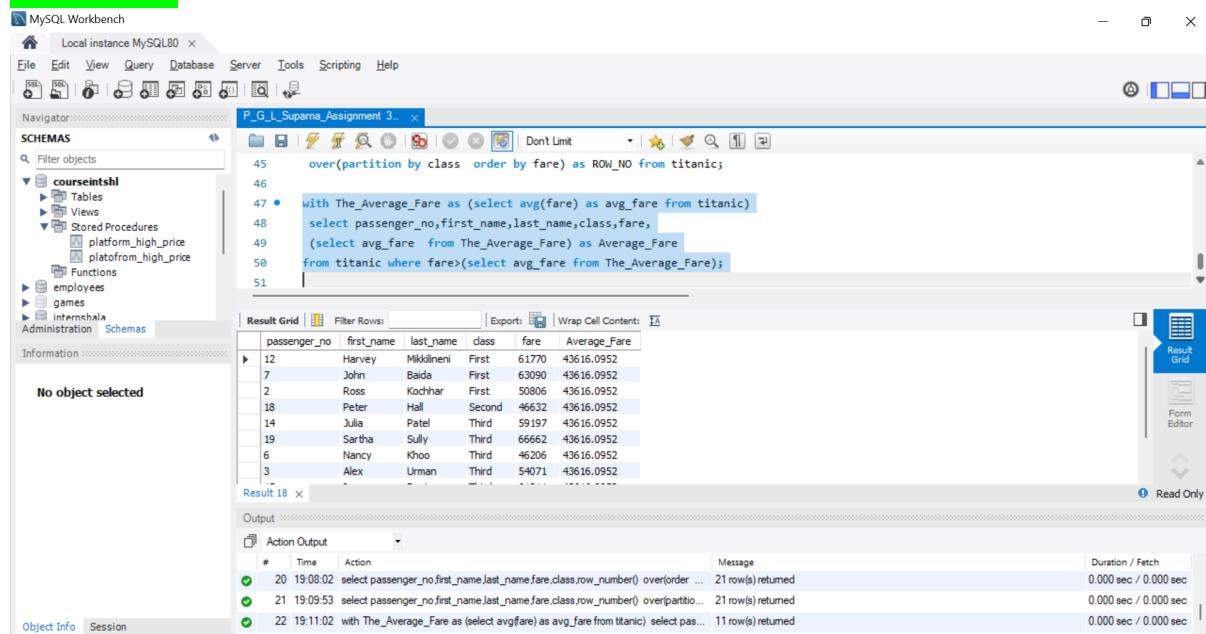
#	Time	Action	Message	Duration / Fetch
19	19:05:47	select Passenger_No, last_name, first_name, class, fare, dense_rank() over(partition by class order by fare desc) as Dense_Fare_Rank from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
20	19:08:02	select passenger_no, first_name, last_name, fare, class, row_number() over(order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
21	19:09:53	select passenger_no, first_name, last_name, fare, class, row_number() over(partition by class order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec

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:



The screenshot shows the MySQL Workbench interface with a query editor window titled "P_G_L_Suparna_Assignment_3...". The code entered is:

```
over(partition by class order by fare) as ROW_NO from titanic;
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);
```

The Result Grid shows 18 rows of passenger data. The columns are: passenger_no, first_name, last_name, class, fare, and Average_Fare. The fare values are all 43616.0952 except for the last row which is 54071.

passenger_no	first_name	last_name	class	fare	Average_Fare
12	Harvey	Mikkilineni	First	61770	43616.0952
7	John	Baida	First	63090	43616.0952
2	Ross	Kochhar	First	50804	43616.0952
18	Peter	Hall	Second	46632	43616.0952
14	Julia	Patel	Third	59197	43616.0952
19	Sartha	Sully	Third	66662	43616.0952
6	Nancy	Khoo	Third	46206	43616.0952
3	Alex	Urman	Third	54071	43616.0952
...
Result 18					

The Output pane shows three log entries:

#	Time	Action	Message	Duration / Fetch
20	19:08:02	select passenger_no,first_name,last_name,fare,class,ROW_NUMBER() over(partition by class order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
21	19:09:53	select passenger_no,first_name,last_name,fare,class,ROW_NUMBER() over(partition by class order by fare) as ROW_NO from titanic;	21 row(s) returned	0.000 sec / 0.000 sec
22	19:11:02	with The_Average_Fare as (select avg(fare) as avg_fare from titanic) select passenger_no,first_name,last_name,fare,(select avg_fare from The_Average_Fare) as Average_Fare from titanic where fare>(select avg_fare from The_Average_Fare);	11 row(s) returned	0.000 sec / 0.000 sec

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