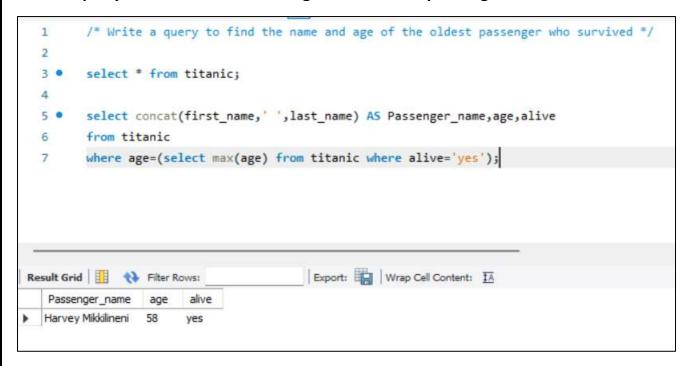
# Assignment 3

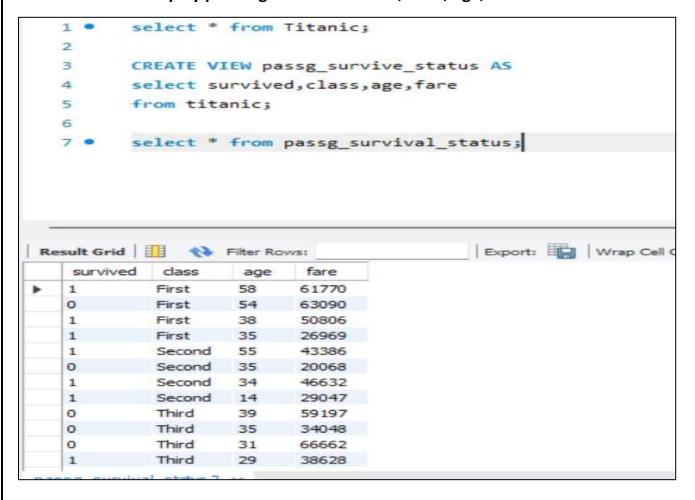
## Task 1:

Write a query to find the name and age of the oldest passenger who survived.



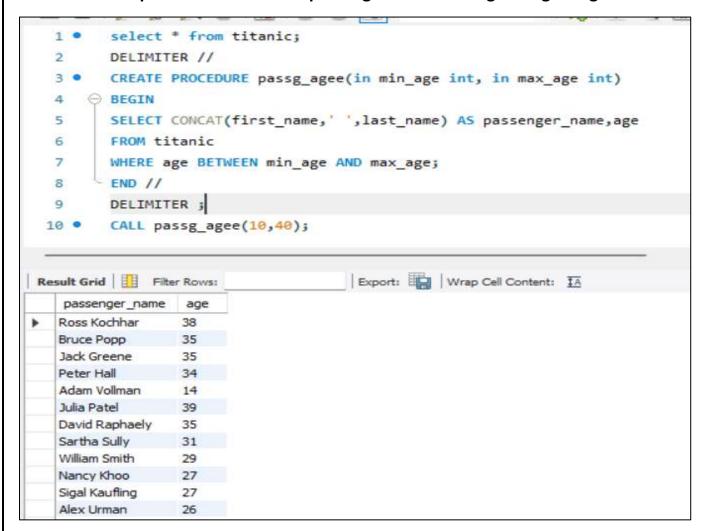
## Task 2:

Create a view to display passenger survival status, class, age, and fare.



## Task 3:

Create a stored procedure to retrieve passengers based on a given age range.



#### Task 4:

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

```
select * from titanic;
         select CONCAT(first_name,' ',last_name) AS passenger_name,age,
      CASE
        WHEN fare < 30000 then 'Low'
        WHEN fare BETWEEN 30000 AND 50000 then 'Medium'
        ELSE 'High'
       END AS passg_detail
  8
  9
        from titanic;
 10
                                        Export: Wrap Cell Content: IA
passenger_name
                       passg_detail
                 age
  Harvey Mikkilineni
                 58
                       High
              54 High
  John Baida
  Ross Kochhar
                 38
                       High
            35
  Bruce Popp
                      Low
   James Zlotkey
                 55
                       Medium
  Jack Greene 35 Low
  Peter Hall
                       Medium
  Adam Vollman
                14
  David Raphaely
                35
                      Medium
   Sartha Sully
                 31
                       High
             29
  William Smith
                       Medium
```

## Task 5:

Show each passenger's fare and the fare of the next passenger

```
1 .
        select * from titanic;
  2
       select concat(first_name,' ',last_name) as FULL_NAME,fare,
  3
        LEAD(fare) Over(order by fare desc) as next_passg_fare
       from titanic;
                                        Export: Wrap Cell Content: TA
FULL_NAME
                fare
                       next_passg_fare
  Sartha Sully
                 66662
                        63090
               63090 61770
  John Baida
  Harvey Mikkilineni
                 61770
                        61211
  Irene Davies 61211 59197
  Julia Patel
                 59197
                       56356
  Daniel Weiss 56356 54071
  Alex Urman
                 54071
                       51428
  Mathew Himuro 51428 50806
  Ross Kochhar
                50806 46632
               46632 46206
  Peter Hall
  Nancy Khoo
                 46206
                      43386
  James Zlotkey 43386 38628
```

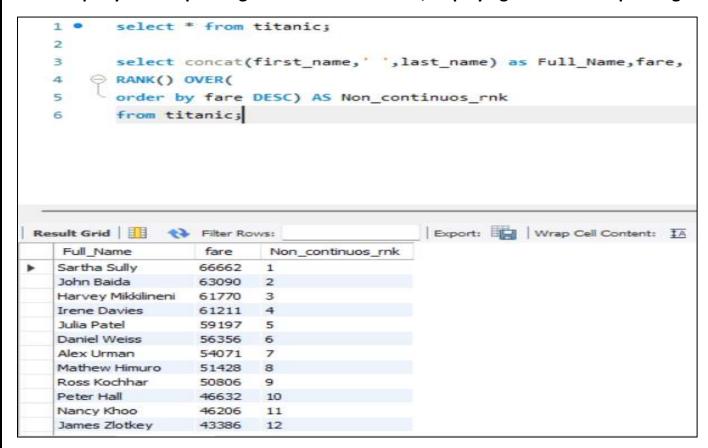
#### Task 6:

Show the age of each passenger and the age of the previous passenger

```
select * from titanic;
  1 •
  2
         select concat(first_name, ' ',last_name) AS Full_Name, age,
  3
      O LAG(age) OVER(
      order by age desc) as prev_passg_age
  5
        from titanic;
  6
Export: Wrap Cell Content: TA
   Full_Name
                age
                        prev_passg_age
                       MULL
  Harvey Mikkilineni
                  58
  James Zlotkey 55
                      58
                 54
  John Baida
                        55
               39
  Julia Patel
                       54
  Ross Kochhar
                35
                      38
  Bruce Popp
  Jack Greene
                 35
                        35
  David Raphaely 35
                      35
  Peter Hall
                 34
                       35
  Sartha Sully
                31
                       34
  William Smith
                  29
                        31
                  27
   Nancy Khoo
                        29
```

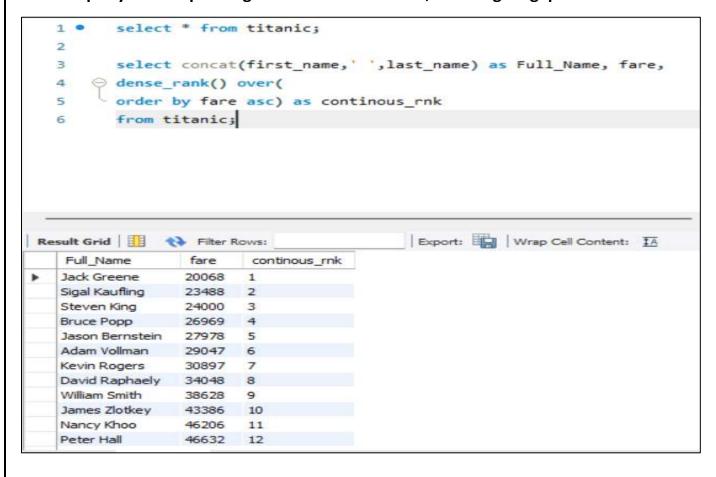
## Task 7:

Write a query to rank passengers based on their fare, displaying rank for each passenger.



#### **Task 8:**

Write a query to rank passengers based on their fare, ensuring no gaps in rank



## Task 9:

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

```
select * from titanic;
  1 .
  2
         select concat(first_name, ' ',last_name)as Full_Name,fare,
  3
  4 @ row_number() over(
       order by fare desc) as row no
         from titanic;
  6
                                            Export: Wrap Cell Content: IA
Result Grid Filter Rows:
   Full Name
                   fare
                          row_no
  Sartha Sully
                   66662
                          1
  John Baida
                  63090
                          2
  Harvey Mikkilineni
                  61770
                          3
  Irene Davies 61211 4
   Julia Patel
                   59197
                          5
                 56356
  Daniel Weiss
                          6
   Alex Urman
                          7
                   54071
  Mathew Himuro 51428 8
  Ross Kochhar
                   50806
   Peter Hall
                  46632 10
  Nancy Khoo
                  46206
                          11
   James Zlotkey
                  43386 12
```

### **Task 10:**

Use a CTE to calculate the average fare and find passengers who paid more than the average.

```
select * from titanic;
         WITH Avg_Fare AS
         (select round(avg(fare),2) as avg_fare from titanic)
         select CONCAT(t.first_name, ' ',t.last_name) AS Full_Name,
  6
         t.fare, Avg_Fare.avg_fare from titanic t JOIN Avg_Fare ON t.fare > avg_fare;
                                       Export: Wrap Cell Content: IA
Result Grid | | Filter Rows:
   Full_Name
                          avg_fare
  Harvey Mikkilineni
                  61770
                          43616,10
  John Baida 63090 43616, 10
  Ross Kochhar
                  50806
                         43616.10
                 46632 43616.10
   Peter Hall
   Julia Patel
                  59197
                         43616.10
  Sartha Sully 66662 43616.10
  Nancy Khoo
                   46206
                         43616.10
   Alex Urman
                 54071 43616.10
   Irene Davies
                  61211
                          436 16.10
  Mathew Himuro 51428 43616.10
   Daniel Weiss
                  56356 43616.10
```