COMMON QUESTIONS WITH ANSWERS

```
/*
1. List the 10th highest Employee's Annual Salary.
2. What is the difference between Rank() and Dense_Rank().
3. List the running total of the annual salary.
*/
CREATE DATABASE ChrisWorkDB
GO
IF OBJECT_ID('dbo.ChrisWorkDB', 'U') IS NOT NULL
 DROP TABLE [dbo].[ChrisWorkDB];
 GO
 CREATE TABLE Employee
 ( EmployeeId INT
  ,EmployeeName VARCHAR(50)
  ,AnnualSalary INT
 )

 USE ChrisWorkDB;
 --truncate table Employee

 INSERT INTO Employee
 VALUES(1, 'John', 35000),
       (2, 'Mary', 60000),
        (3, 'Mark', 70000),
        (4, 'Joe', 90000),
        (5, 'Chris', 30000),
        (6, 'Paul', 30000),
        (7, 'Eric', 65000),
        (8, 'Steve', 65000),
        (9, 'Bruce', 101000),
        (10, 'Jennifer', 95000),
        (11, 'Mike', 82000);
```

```
-- 1. List the 10th highest Employee's Annual Salary.

 WITH Highest_Rank_Salary AS
      (
        SELECT EmployeeName
               ,AnnualSalary, ROW_NUMBER () OVER (ORDER BY AnnualSalary DESC) AS Ranking
       FROM Employee
        )

 SELECT EmployeeName, AnnualSalary, Ranking
 FROM Highest_Rank_Salary
 WHERE Ranking = 10

-- 2. What is the difference between Rank() and Dense_Rank().
/*The one and only difference between the DENSE_RANK() and RANK() functions is the fact
that RANK() will assign non-consecutive ranks to the values in a set in the case of a tie,
which means that with RANK() there will be gaps between the integer values when there is a tie.
But the DENSE_RANK() will assign consecutive ranks to the values in the case of a tie,
so there will be no gaps between the integer values in the case of a tie.*/

 SELECT EmployeeName
        ,AnnualSalary
        ,RANK() OVER(ORDER BY AnnualSalary DESC) AS Ranking
 FROM Employee

 EmployeeName    AnnualSalary    Ranking
Bruce             101000           1
Jennifer           95000           2
Joe               90000            3
Mike              82000            4
Mark              70000            5
Eric              65000            6
Steve             65000            6
Mary              60000            8
John              35000            9
Chris             30000           10
Paul              30000           10
```

```
 SELECT EmployeeName
        ,AnnualSalary
        ,DENSE_RANK() OVER(ORDER BY AnnualSalary DESC) AS Dense_Ranking
 FROM Employee

 EmployeeName    AnnualSalary     Dense_Ranking
Bruce             101000                 1
Jennifer           95000                 2
Joe               90000                  3
Mike              82000                  4
Mark              70000                  5
Eric              65000                  6
Steve             65000                  6
Mary              60000                  7
John              35000                  8
Chris             30000                  9
Paul              30000                  9

--3. List the running total of the annual salary.
SELECT EmployeeName
      ,AnnualSalary
      ,SUM(AnnualSalary) OVER(ORDER BY EmployeeName) AS Running_Total
FROM Employee
--Checking
SELECT SUM(AnnualSalary) AS total
FROM Employee
EmployeeName     AnnualSalary     Running_Total
Bruce              101000           101000
Chris               30000           131000
Eric                65000           196000
Jennifer             95000           291000
Joe                 90000            381000
John                35000           416000
Mark                70000           486000
Mary                60000           546000
Mike                82000           628000
Paul                30000           658000
Steve               65000           723000
```