Back to Resume

```
/*
                         Understanding GROUP BY and how ROLLUP/CUBE functions work
*/
  USE master;
  GO

  IF EXISTS (SELECT name FROM sys.databases WHERE name = N'Rollup_Cube_Example')
  BEGIN
      ALTER DATABASE [Rollup_Cube_Example] SET SINGLE_USER WITH ROLLBACK IMMEDIATE
--======================= Drop This Database and End Function =======================--
      DROP DATABASE [Rollup_Cube_Example] --If it already exists so we can start fresh.
     /*Print out that the database was dropped */
     /*Convert to sysdatetime and then cast to varchar. */
     PRINT 'Rollup_Cube_Example Database: Dropped Database Successfully.'
            + CAST(CONVERT(varchar, SYSDATETIME(), 121) AS varchar (20))
  END
--=================== Create the Database with Settings =======================--
  --Its always a good idea to plan the size of the database for growth. --Plenty of size...
  CREATE DATABASE [Rollup_Cube_Example] ON PRIMARY
          (NAME = N'Rollup_Cube_Example', FILENAME = N'C:\Rollup_Cube_Example.mdf'
          , SIZE = 10MB, MAXSIZE = 1GB, FILEGROWTH = 10MB)
  LOG ON
          --Store Database Here
          (NAME = N'Rollup_Cube_Example_log'
          --Store Log File Here
          , FILENAME = N'C:\Rollup_Cube_Example_log.LDF', SIZE = 1MB, MAXSIZE = 1GB
          , FILEGROWTH = 10MB)
  GO

  /*Log In Owner Database Name = SA Note: SA Means "System Administrator"*/
  EXEC [Rollup_Cube_Example].dbo.sp_changedbowner @loginame = N'SA', @map=false
  USE Rollup_Cube_Example
```

```sql
--Note: Employees' ManagerID is their Manager's EmployeeID
--Drop the table first if it exists.
IF OBJECT_ID('Employee', 'U') IS NOT NULL
DROP TABLE [dbo].[Employee];
GO

CREATE TABLE Employee
                    (EmployeeID INT PRIMARY KEY NOT NULL
                    ,EmployeeName VARCHAR(25)
                    ,Gender VARCHAR(8)
                    ,Title VARCHAR(25)
                    ,ManagerID INT
                    ,HireDate DATETIME
                    ,Salary INT
                    ,DepartmentID INT
                    ,DepartmentName VARCHAR(25)
                    )
/*Print out that the Employee table was created */
/*Convert to sysdatetime and then cast to varchar. */
PRINT 'Rollup_Cube_Example: Employee table was created Successfully.'
         + CAST(CONVERT(varchar, SYSDATETIME(), 121) AS varchar (20))

INSERT INTO Employee
    VALUES (2801, 'Ryan Anderson', 'Male', 'President', NULL, '05/10/2015', 200000, 10, 'Chief Executive Officer'),
           (2632, 'John Hancock', 'Male', 'IT Manager', 2801, '05/01/2016', 145000, 20, 'Information Technology'),
           (2755, 'Jane Potter', 'Female', 'Finance Manager', 2801, '12/01/2015', 115000, 30, 'Accounting'),
           (2600, 'David Singleton', 'Male', 'Sales Manager', 2801, '07/08/2015', 110000, 40, 'Sales'),
           (2933, 'Allen Jensing', 'Male', 'BI Developer', 2632, '09/02/2017', 125000, 20, 'Information Technology'),
           (2818, 'Lisa Jenkins', 'Female', 'Data Analyst', 2632, '02/25/2016', 70000, 20, 'Information Technology'),
           (2511, 'James Cassidy', 'Male', 'Accountant', 2755, '02/01/2015', 55000, 30, 'Accounting'),
           (2786, 'Clark Duran', 'Male', 'Accounting Assistant', 2755, '09/28/2016', 35000, 30, 'Accounting'),
           (2811, 'Bruce Hendrix', 'Male', 'Salesman', 2600, '05/03/2015', 40000, 40, 'Sales'),
           (2683, 'Paul Fisher', 'Male', 'Salesman', 2600, '06/03/2015', 38000, 40, 'Sales');

/*Print out that the Employee table was populated with data */
/*Convert to sysdatetime and then cast to varchar. */
PRINT 'Rollup_Cube_Example: Employee table was populated with data Successfully.'
        + CAST(CONVERT(varchar, SYSDATETIME(), 121) AS varchar (20))
```

```
--Checking: SELECT * FROM Employee ORDER BY DepartmentID

/*
  Our Employee table's data set:
EmployeeID   EmployeeName    Gender  Title                 ManagerID         HireDate          Salary      DepartmentID  DepartmentName
-------------------------------------------------------------------------------------------------------------------------------------------
  2801        Ryan Anderson     Male     President              NULL    2015-05-10 00:00:00.000   200000        10         Chief Executive Officer
  2632        John Hancock      Male     IT Manager             2801    2016-05-01 00:00:00.000   145000        20         Information Technology
  2818        Lisa Jenkins      Female   Data Analyst           2632    2016-02-25 00:00:00.000    70000        20         Information Technology
  2933        Allen Jensing     Male     BI Developer           2632    2017-09-02 00:00:00.000   125000        20         Information Technology
  2511        James Cassidy     Male     Accountant             2755    2015-02-01 00:00:00.000    55000        30         Accounting
  2755        Jane Potter       Female   Finance Manager        2801    2015-12-01 00:00:00.000   115000        30         Accounting
  2786        Clark Duran       Male     Accounting Assistant  2755    2016-09-28 00:00:00.000    35000        30         Accounting
  2811        Bruce Hendrix     Male     Salesman               2600    2015-05-03 00:00:00.000    40000        40         Sales
  2600        David Singleton   Male     Sales Manager          2801    2015-07-08 00:00:00.000   110000        40         Sales
  2683        Paul Fisher       Male     Salesman               2600    2015-06-03 00:00:00.000    38000        40         Sales
*/




--Get all employees and their manager's name.
--Logic approach: employee's manager id =(join) manager's employee id

  SELECT e.EmployeeID
          ,e.EmployeeName AS Employee
          ,e.Title
          ,e.ManagerID
          ,m.EmployeeName AS Manager
  FROM Employee AS e
           LEFT JOIN Employee AS m
           ON e.ManagerID = m.EmployeeID
  ORDER BY m.EmployeeName  --Note: We are really ordering by Manager Ascending.
```

```
/* Output:
    All employee's with their manager's name.
EmployeeID    Employee                Title                       ManagerID      Manager
------------------------------------------------------------------------------------------------------------
   2801           Ryan Anderson         President                   NULL              NULL
   2811           Bruce Hendrix          Salesman                   2600              David Singleton
   2683           Paul Fisher              Salesman                   2600              David Singleton
   2511           James Cassidy         Accountant                 2755              Jane Potter
   2786           Clark Duran             Accounting Assistant   2755              Jane Potter
   2818           Lisa Jenkins            Data Analyst               2632              John Hancock
   2933           Allen Jensing           BI Developer               2632              John Hancock
   2600           David Singleton       Sales Manager             2801              Ryan Anderson
   2632           John Hancock          IT Manager                  2801              Ryan Anderson
   2755           Jane Potter             Finance Manager          2801              Ryan Anderson
*/


--Get all employees who joined the company before their managers.
  SELECT e.EmployeeID
            ,e.EmployeeName AS Employee
            ,CONVERT(VARCHAR(10), e.HireDate, 110) AS EMP_HireDate
            ,e.Title
            ,e.ManagerID
            ,m.EmployeeName AS Manager
            ,CONVERT(VARCHAR(10), m.HireDate, 110) AS MGR_HireDate
  FROM Employee AS e
          LEFT JOIN Employee AS m
          ON e.ManagerID = m.EmployeeID
  WHERE e.HireDate < m.HireDate
  ORDER BY m.HireDate
/*
OutPut:
EmployeeID     Employee              Title               ManagerID        Manager                HireDate
-------------------------------------------------------------------------------------------------------------------------
   2683            Paul Fisher            Salesman             2600             David Singleton      07-08-2015
   2811            Bruce Hendrix        Salesman             2600             David Singleton      07-08-2015
   2511            James Cassidy       Accountant           2755             Jane Potter             12-01-2015
   2818            Lisa Jenkins          Data Analyst         2632             John Hancock          05-01-2016
```

```
*/
          /* Understanding how GROUP BY works with filters */
/*
  What is the total salary in each department?
  This is done by using a simple GROUP BY.
  We are only trying to get the total salary by department.
  Here, we are grouping by DepartmentID and calculating the annual Salary.
  Note: Everything before the aggregate is included in the GROUP BY clause.
*/

  SELECT DepartmentID
          ,SUM(Salary) AS Salary
  FROM Employee
  GROUP BY DepartmentID
  ORDER BY Salary DESC
/*
Output:
DepartmentID     Salary
--------------------------------
   20               340000
   30               205000
   10               200000
   40               188000
*/

/*
  We can filter our grouped rows by using the HAVING clause after the GROUP BY.
  Please note that HAVING is only used with GROUP BY and is always used
  after the GROUP BY. If we wanted to filter before grouping, we would use our
  WHERE clause.
*/
  SELECT DepartmentID
          ,SUM(Salary) AS Salary
  FROM Employee
  GROUP BY DepartmentID
  HAVING DepartmentID = 20 OR DepartmentID = 30
  ORDER BY Salary DESC
```

```
/*
Output:
DepartmentID     Salary
---------------------------------
   20                    340000
   30                    205000
*/




/*
  Here we are using the WHERE clause and filtering by the HireDate before the GROUP BY.
*/
--SELECT * FROM Employee

/*
  Please give me the employees total salary where employees were hired between
  the hire dates of 05/03/2015 and 06/02/2016. When we grouped, we only grouped
  those DepartmentID columns that were between the dates.

  Note: It's important to understand that the numbers changed on Salary because of
  filtering our employee hire dates using the WHERE clause before the GROUP BY.
  The WHERE clause can have a profound effect on how your groupings data are aggregated.
*/

  SELECT DepartmentID
          ,SUM(Salary) AS Salary
  FROM Employee
  WHERE HireDate BETWEEN '05/03/2015' AND '06/02/2016'
  GROUP BY DepartmentID
  HAVING DepartmentID = 20 OR DepartmentID = 30 OR DepartmentID = 40
  ORDER BY Salary DESC
/*
Output:
DepartmentID    Salary
--------------------------------
   30                  115000
   40                  188000
   20                  215000
*/
```

```
/*********************************************************/
/*********************** Using ROLLUP *******************/
/*********************************************************/

/*
  Total all the Salaries of each employee’s department and give me the total
  of everyone's salary by using one SELECT statement.
*/
 SELECT COALESCE(DepartmentName, 'All Departments') AS ALL_Departments
       ,SUM(Salary) AS Salary_Total
 FROM Employee
 GROUP BY ROLLUP (DepartmentName)

/*
Output:
ALL_Departments                    Salary_Total
----------------------------------------------------------
  Accounting                          205000
  Chief Executive Officer             200000
  Information Technology              340000
  Sales                               188000
  All Departments                     933000

*/
/*
  Let's calculate department other than the CEO's salary, then total everything.
*/
 SELECT COALESCE(DepartmentName, 'All Departments') AS ALL_Departments
       ,SUM(Salary) AS Salary_Total
 FROM Employee
 WHERE DepartmentName != 'Cheif Executive Officer'
 GROUP BY ROLLUP (DepartmentName)

/*
  Another way... using the NOT IN operator (preferred method).
*/
 SELECT COALESCE(DepartmentName, 'All Departments') AS ALL_Departments
       ,SUM(Salary) AS Salary_Total
 FROM Employee
 WHERE DepartmentName NOT IN ('Cheif Executive Officer')
 GROUP BY ROLLUP (DepartmentName)
```

```
/*
Output:
ALL_Departments                    Salary_Total
-------------------------------------------------------
   Accounting                          205000
   Information Technology              340000
   Sales                               188000
   All Departments                     733000
*/




/*******************************************************/
/******************** CUBE VS ROLLUP ********************/
/*******************************************************/
```

Note: For this demonstration we will not be using ORDER BY, so we can show the difference.

```
  SELECT COALESCE(DepartmentName, 'All Departments') AS ALL_Departments
            ,SUM(Salary) AS Salary_Total
  FROM Employee
  GROUP BY CUBE (DepartmentName)

/*
   Note that there is not difference as with ROLLUP; both come to the same conclusion.
   This is because we are using GROUP BY with only one column.
Output:
ALL_Departments                    Salary_Total
-------------------------------------------------------
   Accounting                          205000
   Chief Executive Officer             200000
   Information Technology              340000
   Sales                               188000
   All Departments                     933000
*/
```

```
/*
  Now let's add another column to the table called "Gender", use it in our GROUP BY
  and see what happens.
*/



 SELECT COALESCE (DepartmentName, 'All Departments') AS Department
            ,COALESCE (Gender,'All Genders') AS Gender,
       SUM(Salary) as Salary_Total
 FROM Employee
 GROUP BY ROLLUP (DepartmentName, Gender)

/*
  Please pay special attention to how the data is formatted.

  Note: Each ROLLUP is by department showing the total for each Gender
  in a hierarchical format with the total for everything at the bottom.
*/

/*
Department                          Gender        Salary_Total
---------------------------------------------------------------------------
  Accounting                        Female           115000
  Accounting                        Male              90000
  Accounting                      All Genders        205000
  Chief Executive Officer           Male             200000
  Chief Executive Officer         All Genders        200000
  Information Technology            Female            70000
  Information Technology            Male             270000
  Information Technology          All Genders        340000
  Sales                             Male             188000
  Sales                           All Genders        188000
  All Departments                 All Genders        933000   --ROLLUP does a nice job (Hierarchy Structure)
*/
```

```sql
  SELECT COALESCE (DepartmentName, 'All Departments') AS Department
            ,COALESCE (Gender,'All Genders') AS Gender
            ,SUM(Salary) as Salary_Total
  FROM Employee
  GROUP BY CUBE (DepartmentName, Gender)
/*
   Please note: That here we have our totals of male and female by each department in a different order.
   We total the Females first of each department, then total all departments that have
   Females. Then we total each department that has males and total all the males of each department.
   Next, we total all the departments together (Male and Female) as "All Genders", then strangely enough
   under that we total every department as "All Genders" and then finally each department under “All Genders”.

   CUBE doesn't display the information in a hierarchal type format, but in an aggregated format of all
   possible combinations. Looks like we are going down the line of each Gender first and then All Genders.
   Note that we calculate all departments on the first “All Genders” set.
*/
/*
Department                     Gender         Salary_Total
------------------------------------------------------------------------------
   Accounting                   Female            115000
   Information Technology       Female             70000
   All Departments              Female            185000
   Accounting                   Male               90000
   Chief Executive Officer      Male              200000
   Information Technology       Male              270000
   Sales                        Male              188000
   All Departments              Male              748000
   All Departments           All Genders          933000   --Note where the total of everything is (interesting).
   Accounting                All Genders          205000
   Chief Executive Officer   All Genders          200000
   Information Technology    All Genders          340000
   Sales                     All Genders          188000
*/
/*
   The big difference to remember is: You should use ROLLUP if you want your data presented
   in a hierarchal aggregated format and CUBE if you want all possible combinations to be shown
   with aggregation.
*/
```