use AdventureWorksDW2014;

**-- Find the top products of 2013**

**-- using ROW\_NUMBER() as a Rank function**

**-- fragile solution**

SELECT **Row\_number**()   
         OVER (   
           ORDER BY *Sum*(s.salesamount) DESC) AS 'Rank',   
       *Count*(DISTINCT s.salesordernumber)    'OrderCount'   
       *-- use 1 instead of a field for faster performance*   
       ,   
       *Sum*(s.salesamount)                    'Sales',   
       cat.englishproductcategoryname        'Category',   
       sub.englishproductsubcategoryname     'SubCategory'   
FROM   factinternetsales s   
       INNER JOIN dimproduct p   
               ON s.productkey = p.productkey   
       INNER JOIN dimproductsubcategory sub   
               ON p.productsubcategorykey = sub.productsubcategorykey   
       INNER JOIN dimproductcategory cat   
               ON sub.productcategorykey = cat.productcategorykey   
*-- filter*   
WHERE  *Year*(s.orderdate) = 2013 *--use date function to parse year*   
*-- must use group by in order for aggregation to work properly*   
GROUP  BY cat.englishproductcategoryname *-- column aliases aren't allowed*   
          ,   
          sub.englishproductsubcategoryname   
ORDER  BY 3 DESC;

**-- use RANK() function instead**

**-- when RANK() and ROW\_NUMBER() have the same order by the results are the same**

SELECT **Row\_number**()   
         OVER (   
           ORDER BY *Sum*(s.salesamount) DESC) AS 'Rank',   
       *Count*(DISTINCT s.salesordernumber)    'OrderCount'   
       *-- use 1 instead of a field for faster performance*   
       ,   
       **Rank**()   
         OVER (   
           ORDER BY *Sum*(s.salesamount) DESC) 'SalesRank',   
       *Sum*(s.salesamount)                    'TotalSales',   
       cat.englishproductcategoryname        'Category',   
       sub.englishproductsubcategoryname     'SubCategory'   
FROM   factinternetsales s   
       INNER JOIN dimproduct p   
               ON s.productkey = p.productkey   
       INNER JOIN dimproductsubcategory sub   
               ON p.productsubcategorykey = sub.productsubcategorykey   
       INNER JOIN dimproductcategory cat   
               ON sub.productcategorykey = cat.productcategorykey   
*-- filter*   
WHERE  *Year*(s.orderdate) = 2013 *--use date function to parse year*   
*-- must use group by in order for aggregation to work properly*   
GROUP  BY cat.englishproductcategoryname *-- column aliases aren't allowed*   
          ,   
          sub.englishproductsubcategoryname   
ORDER  BY cat.englishproductcategoryname,   
          sub.englishproductsubcategoryname;

**-- Show the top product Sub Categories for each year**

SELECT *Count*(DISTINCT s.salesordernumber)    'OrderCount'   
       *-- use 1 instead of a field for faster performance*   
       ,   
       **Rank**()   
         OVER (   
           partition BY *Year*(s.orderdate)   
           ORDER BY *Sum*(s.salesamount) DESC) 'SalesRank',   
       *Sum*(s.salesamount)                    'TotalSales',   
       cat.englishproductcategoryname        'Category',   
       sub.englishproductsubcategoryname     'SubCategory',   
       *Year*(s.orderdate)                     'Year'   
FROM   factinternetsales s   
       INNER JOIN dimproduct p   
               ON s.productkey = p.productkey   
       INNER JOIN dimproductsubcategory sub   
               ON p.productsubcategorykey = sub.productsubcategorykey   
       INNER JOIN dimproductcategory cat   
               ON sub.productcategorykey = cat.productcategorykey   
*-- must use group by in order for aggregation to work properly*   
GROUP  BY cat.englishproductcategoryname *-- column aliases aren't allowed*   
          ,   
          sub.englishproductsubcategoryname,   
          *Year*(s.orderdate)   
ORDER  BY *Year*(s.orderdate),   
          *Sum*(s.salesamount) DESC;

**-- Year over Year Calculations**

USE AdventureWorksDW2014;

GO

**-- Get Previous Year Sales**

WITH monthlysales (yearnum, monthnum, sales)   
     AS (SELECT d.calendaryear,   
                d.monthnumberofyear,   
                *Sum*(s.salesamount)   
         FROM   dimdate d   
                JOIN factinternetsales s   
                  ON d.datekey = s.orderdatekey   
         GROUP  BY d.calendaryear,   
                   d.monthnumberofyear)   
*-- Get Current Year and join to CTE for previous year*   
SELECT d.calendaryear,   
       d.monthnumberofyear,   
       ms.sales           PrevSales,   
       Sum(s.salesamount) CurrentSales   
FROM   dimdate d   
       JOIN factinternetsales s   
         ON d.datekey = s.orderdatekey   
       JOIN monthlysales ms   
         ON d.calendaryear - 1 = ms.yearnum   
            AND d.monthnumberofyear = ms.monthnum   
GROUP  BY d.calendaryear,   
          d.monthnumberofyear,   
          ms.sales   
ORDER  BY 1 DESC,   
          2 DESC

**-- Now calculate the % change Year over Year**

WITH monthlysales (yearnum, monthnum, sales)   
     AS (SELECT d.calendaryear,   
                d.monthnumberofyear,   
                *Sum*(s.salesamount)   
         FROM   dimdate d   
                JOIN factinternetsales s   
                  ON d.datekey = s.orderdatekey   
         GROUP  BY d.calendaryear,   
                   d.monthnumberofyear)   
*-- Get Current Year and join to CTE for previous year*   
SELECT d.calendaryear,   
       d.monthnumberofyear,   
       ms.sales as PrevSales,   
       *Sum*(s.salesamount)  CurrentSales,   
       ( *Sum*(s.salesamount) - ms.sales ) / *Sum*(s.salesamount) 'PctGrowth'   
FROM   dimdate d   
       JOIN factinternetsales s   
         ON d.datekey = s.orderdatekey   
       JOIN monthlysales ms   
         ON d.calendaryear - 1 = ms.yearnum   
            AND d.monthnumberofyear = ms.monthnum   
GROUP  BY d.calendaryear,   
          d.monthnumberofyear,   
          ms.sales   
ORDER  BY 1 DESC,   
          2 DESC

**-- Common Table Expressions (CTEs)**

USE AdventureWorksDW2014

GO

**-- use a CTE to get an aggregate of an aggregate**

**-- Show number of profitable weeks**

WITH sales\_cte (yr, weeknum, weeklysales)   
     AS (SELECT *Year*(orderdate)         AS Yr,   
                *Datepart*(wk, orderdate) AS WeekNum,   
                *Sum*(salesamount)        AS WeeklySales   
         FROM   factinternetsales   
         GROUP  BY *Year*(orderdate),   
                   *Datepart*(wk, orderdate))   
SELECT \*,   
       CASE   
         WHEN weeklysales > 140000 THEN 1   
         ELSE 0   
       END AS 'Profitable'   
FROM   sales\_cte   
ORDER  BY 1,   
          2   
  
go

**-- Summarize by Year**

WITH sales\_cte (yr, weeknum, weeklysales)   
     AS (SELECT *Year*(orderdate)         AS Yr,   
                *Datepart*(wk, orderdate) AS WeekNum,   
                *Sum*(salesamount)        AS WeeklySales   
         FROM   factinternetsales   
         GROUP  BY *Year*(orderdate),   
                   *Datepart*(wk, orderdate))   
SELECT yr,   
       *Sum*(CASE   
             WHEN weeklysales > 140000 THEN 1   
             ELSE 0   
           END) AS 'Profitable'   
FROM   sales\_cte   
GROUP  BY yr   
ORDER  BY 1   
  
go

-------------------------------------------------------------------------

-- **Use CTE to navigate employee hierarchy**

WITH directreports (managerid, employeeid, title, deptid, level)   
     AS (   
        *-- Anchor member definition*   
        SELECT e.parentemployeekey,   
               e.employeekey,   
               e.title,   
               e.departmentname,   
               0 AS Level   
        FROM   dimemployee AS e   
        WHERE  e.parentemployeekey IS NULL   
         UNION ALL   
         *-- Recursive member definition*   
         SELECT e.parentemployeekey,   
                e.employeekey,   
                e.title,   
                e.departmentname,   
                level + 1   
         FROM   dimemployee AS e   
                INNER JOIN directreports AS d   
                        ON e.parentemployeekey = d.employeeid)   
*-- Statement that executes the CTE*   
SELECT managerid,   
       employeeid,   
       title,   
       deptid,   
       level   
FROM   directreports   
WHERE  deptid = 'Information Services'   
        OR level = 0

**-- Calculate the customer acquisition funnel**

SELECT c.firstname,   
       c.lastname,   
       c.datefirstpurchase,   
       *Datediff*(d, c.datefirstpurchase, *Getdate*()) AS 'DaysSinceFirstPurchase'   
*-- How long have they been a customer?*   
FROM   dimcustomer c   
ORDER  BY 3 DESC

**-- Calculate a Monthly average of customer tenure**

SELECT **Eomonth**(c.datefirstpurchase)                         AS   
       'MonthOfFirstPurchase'   
       *-- What month did they become a customer?*   
       ,   
       *Datediff*(d, **Eomonth**(c.datefirstpurchase), *Getdate*()) AS   
       'DaysSinceFirstPurchase' *-- How long have they been a customer?*   
       ,   
       *Count*(1)                                             AS 'CustomerCount'   
*-- How manY customers are there for this month?*   
FROM   dimcustomer c   
GROUP  BY **Eomonth**(c.datefirstpurchase)   
ORDER  BY 2 DESC

**-- The data might not always be updated, so let’s find the latest monthly sales amount**

*-- Get the most recent month*   
SELECT d.calendaryear,   
       d.monthnumberofyear,   
       mdt.ismaxdate,   
       *Sum*(s.salesamount) AS 'TotalSales'   
FROM   dimdate d   
       JOIN factinternetsales s   
         ON d.datekey = s.orderdatekey   
       LEFT JOIN (SELECT 1              AS 'IsMaxDate',   
                         *Max*(orderdate) AS 'MaxDate'   
                  FROM   factinternetsales) mdt   
              ON d.calendaryear = *Year*(mdt.maxdate)   
                 AND d.monthnumberofyear = *Month*(mdt.maxdate)   
GROUP  BY d.calendaryear,   
          d.monthnumberofyear,   
          mdt.ismaxdate   
ORDER  BY 1 DESC,   
          2 DESC

**-- These counts are cumulative, so for monthly totals take the last day of the month**

SELECT dt.fulldatealternatekey AS 'Date',   
       *Count*(1)                AS ActiveCount   
FROM   dimdate dt   
       LEFT JOIN (SELECT 'Active' AS 'EmpStatus',   
                         \*   
                  FROM   dimemployee) emp   
              *-- regular active employees*   
              ON ( dt.fulldatealternatekey BETWEEN emp.startdate AND   
       **Isnull**(emp.enddate, '9999-12-31') )   
WHERE  dt.fulldatealternatekey = **Eomonth**(dt.fulldatealternatekey)   
GROUP  BY dt.fulldatealternatekey   
ORDER  BY 1

use AdventureWorksDW2014;

**-- Show a 6 week rolling average of Weekly Sales for 2013**

**-- first create weekly sales totals**

SELECT   *Sum*(s.salesamount) 'WeeklySales' ,   
         *Datepart*(ww, s.orderdate) AS 'WeekNum'   
FROM     factinternetsales s   
WHERE    *Year*(s.orderdate) = 2013   
GROUP BY *Datepart*(ww, s.orderdate)   
ORDER BY *Datepart*(ww, s.orderdate) ASC   
*-- use that subquery as our source and calculate the moving average*

SELECT   avg(weeklysales) OVER (ORDER BY weeknum rows BETWEEN 6 PRECEDING AND CURRENT row) AS avgsales ,  
         weeklysales AS 'TotalSales' ,  
         weeknum   
FROM     (   
                  SELECT   sum(s.salesamount) 'WeeklySales' ,   
                           datepart(ww, s.orderdate) AS 'WeekNum'   
                  FROM     factinternetsales s   
                  WHERE    year(s.orderdate) = 2013   
                  GROUP BY datepart(ww, s.orderdate) ) AS s   
GROUP BY weeknum,   
         weeklysales   
ORDER BY weeknum ASC

**-- Running Total**

SELECT *Sum*(monthlysales)   
         OVER ( partition BY salesyear ORDER BY salesmonth rows UNBOUNDED PRECEDING) AS YTDSales,   
       monthlysales AS 'MonthlySales',   
       salesyear,   
       salesmonth   
FROM   (SELECT *Sum*(s.salesamount) 'MonthlySales',   
               *Month*(s.orderdate) AS 'SalesMonth',   
               *Year*(s.orderdate)  AS 'SalesYear'   
        FROM   factinternetsales s   
        GROUP  BY *Month*(s.orderdate),   
                  *Year*(s.orderdate)) AS s   
GROUP  BY salesmonth,   
          salesyear,   
          monthlysales   
ORDER  BY salesyear,   
          salesmonth ASC

**-- Window Functions**

**/\***

**OVER()**

**-- executes an aggregation over a given partition and sort order**

**-- works with Ranking, Aggregate and Analytics functions**

**\*/**

USE AdventureWorksDW2014;

GO

**-- Show each sales average for Group, Country, and Region all in one query**

SELECT DISTINCT t.salesterritorygroup,   
                t.salesterritorycountry,   
                t.salesterritoryregion,   
                *Avg*(s.salesamount)   
                  OVER(   
                    partition BY t.salesterritorygroup )   AS 'GroupAvgSales',   
                *Avg*(s.salesamount)   
                  OVER(   
                    partition BY t.salesterritorycountry ) AS 'CountryAvgSales',   
                *Avg*(s.salesamount)   
                  OVER(   
                    partition BY t.salesterritoryregion )  AS 'RegionAvgSales'   
FROM   factinternetsales s   
       JOIN dimsalesterritory t   
         ON s.salesterritorykey = t.salesterritorykey   
WHERE  *Year*(s.orderdate) = 2013   
ORDER  BY 1,   
          2,   
          3