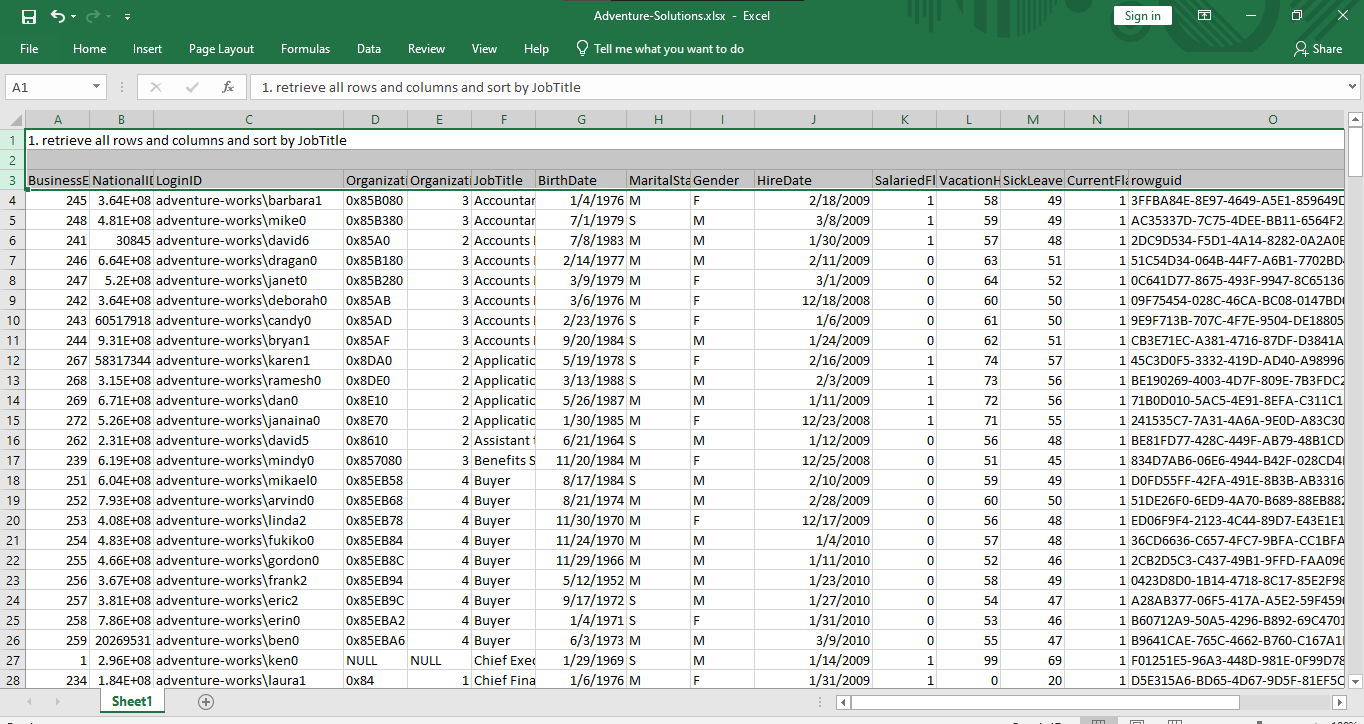
1. From the following table write a query in SQL to retrieve all rows and columns from the employee table in the Adventureworks database. Sort the result set in ascending order on jobtitle.

select \* from HumanResources.Employee order by JobTitle



=====================================================================================

2. From the following table write a query in SQL to retrieve all rows and columns from the employee table using table aliasing in the Adventureworks database. Sort the output in ascending order on lastname.

select E.BusinessEntityID,

ISNULL(P.PersonType,'') as PersonType,

ISNULL(nullif(P.NameStyle,0),'false') as NameStyle,

isnull(P.Title,'') as Title,

ISNULL(P.FirstName,'')as FirstName,

ISNULL(P.MiddleName,'') as MiddleName,

ISNULL(P.LastName,'') as LastName,

ISNULL(P.Suffix,'') as Suffix,

ISNULL(P.EmailPromotion,'') as EmailPromotion,

ISNULL(P.AdditionalContactInfo,'') as AdditionalContactInfo,

ISNULL(P.Demographics,'') as Demographics,

ISNULL(P.rowguid,'') as rowguid ,

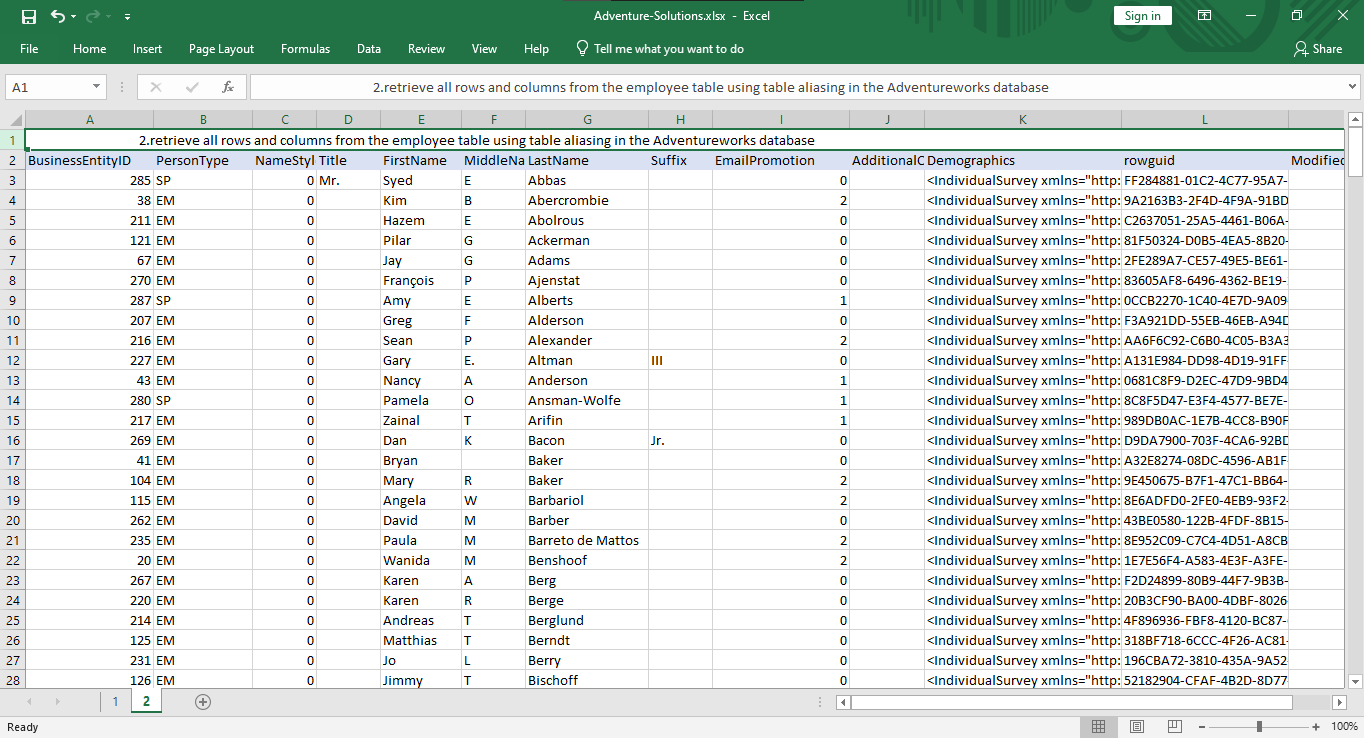
ISNULL(P.ModifiedDate,'')as ModifiedDate

from HumanResources.Employee E

join Person.Person P

on E.BusinessEntityID= P.BusinessEntityID

order by P.LastName asc



=======================================================================================

3. From the following table write a query in SQL to return all rows and a subset of the columns (FirstName, LastName, businessentityid) from the person table in the AdventureWorks database. The third column heading is renamed to Employee\_id. Arranged the output in ascending order by lastname.

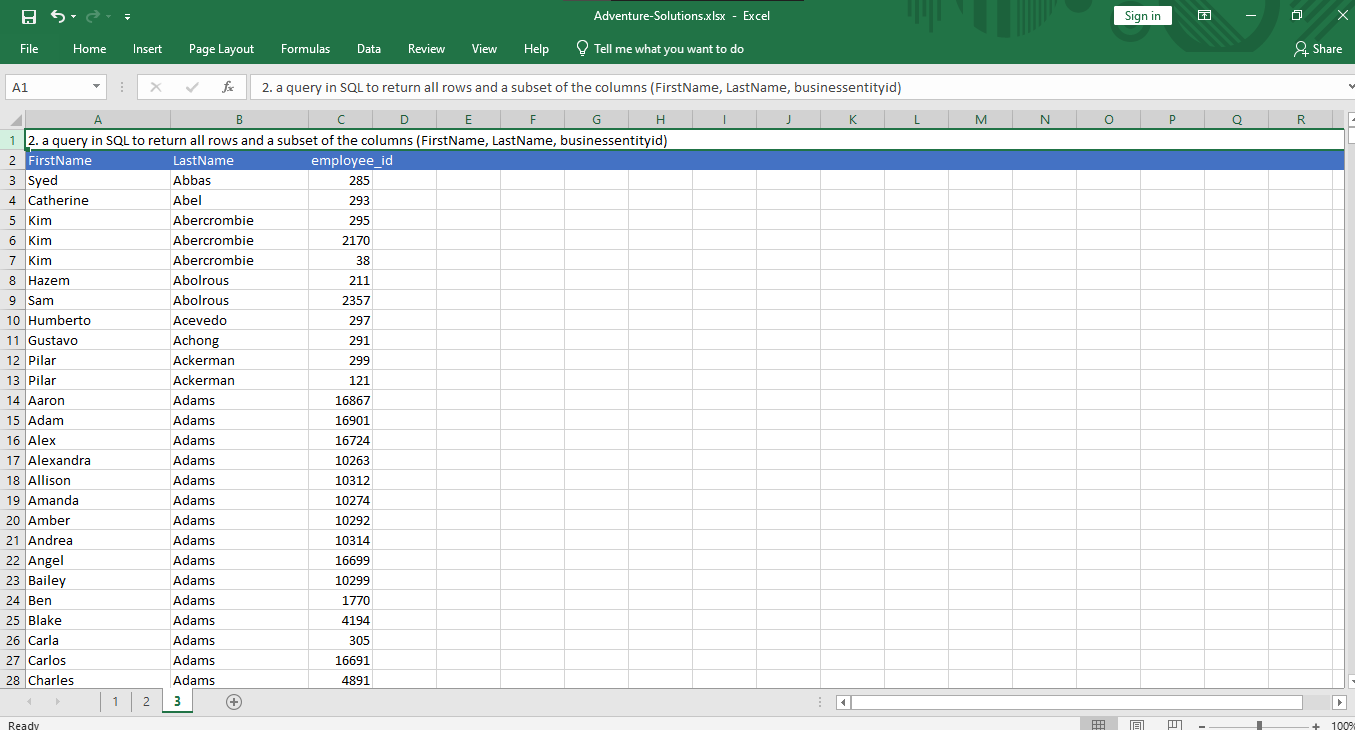
select P.FirstName,

P.LastName,

P.BusinessEntityID as employee\_id

from Person.Person P

order by P.LastName asc



=======================================================================================

4. From the following table write a query in SQL to return only the rows for product that have a sellstartdate that is not NULL and a productline of 'T'. Return productid, productnumber, and name. Arranged the output in ascending order on name.

select P.ProductID,

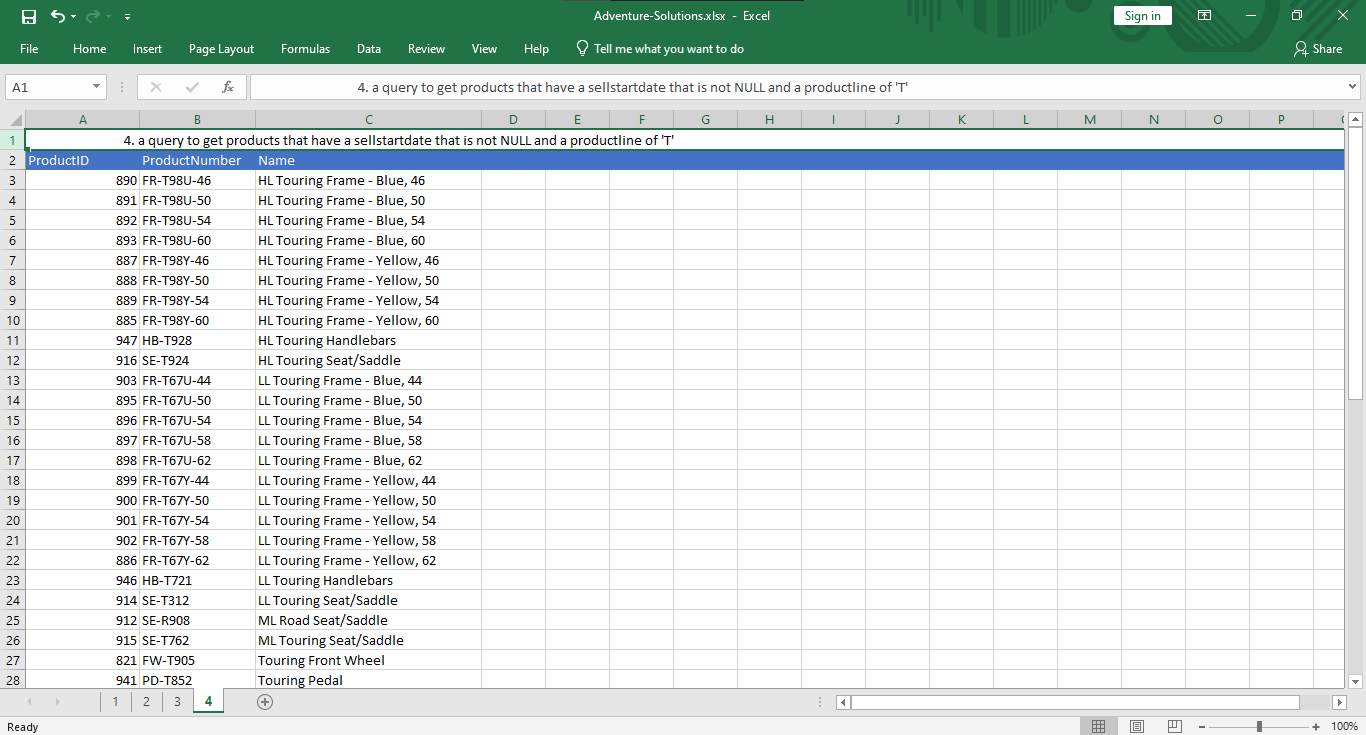
P.ProductNumber,

P.Name,

from production.Product P

where P.SellStartDate is not null and P.ProductLine='T'

order by P.Name asc



=====================================================================================

5. From the following table write a query in SQL to return all rows from the salesorderheader table in Adventureworks database and calculate the percentage of tax on the subtotal have decided. Return salesorderid, customerid, orderdate, subtotal, percentage of tax column. Arranged the result set in ascending order on subtotal.

select salesorderid,

customerid,

orderdate,

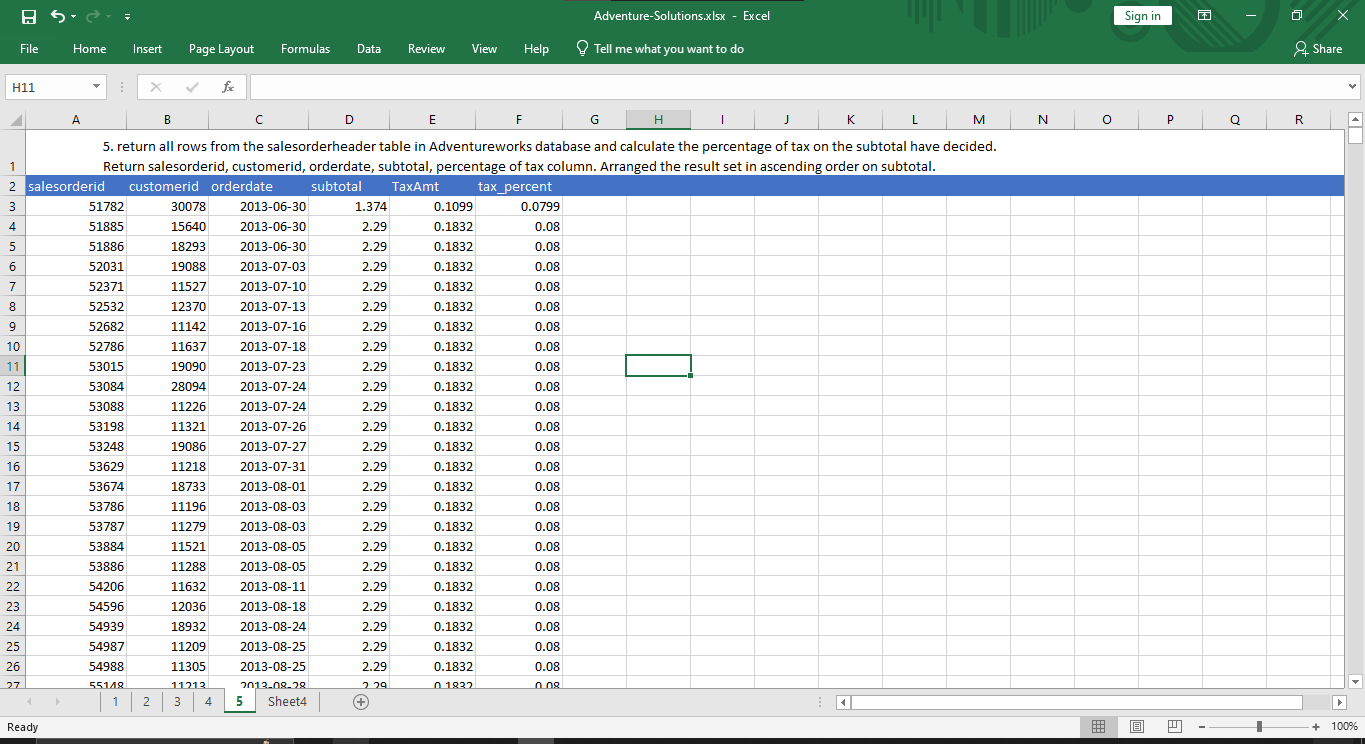
subtotal,

TaxAmt,

(TaxAmt/subtotal) as tax\_percent

from sales.salesorderheader S

order by subtotal asc



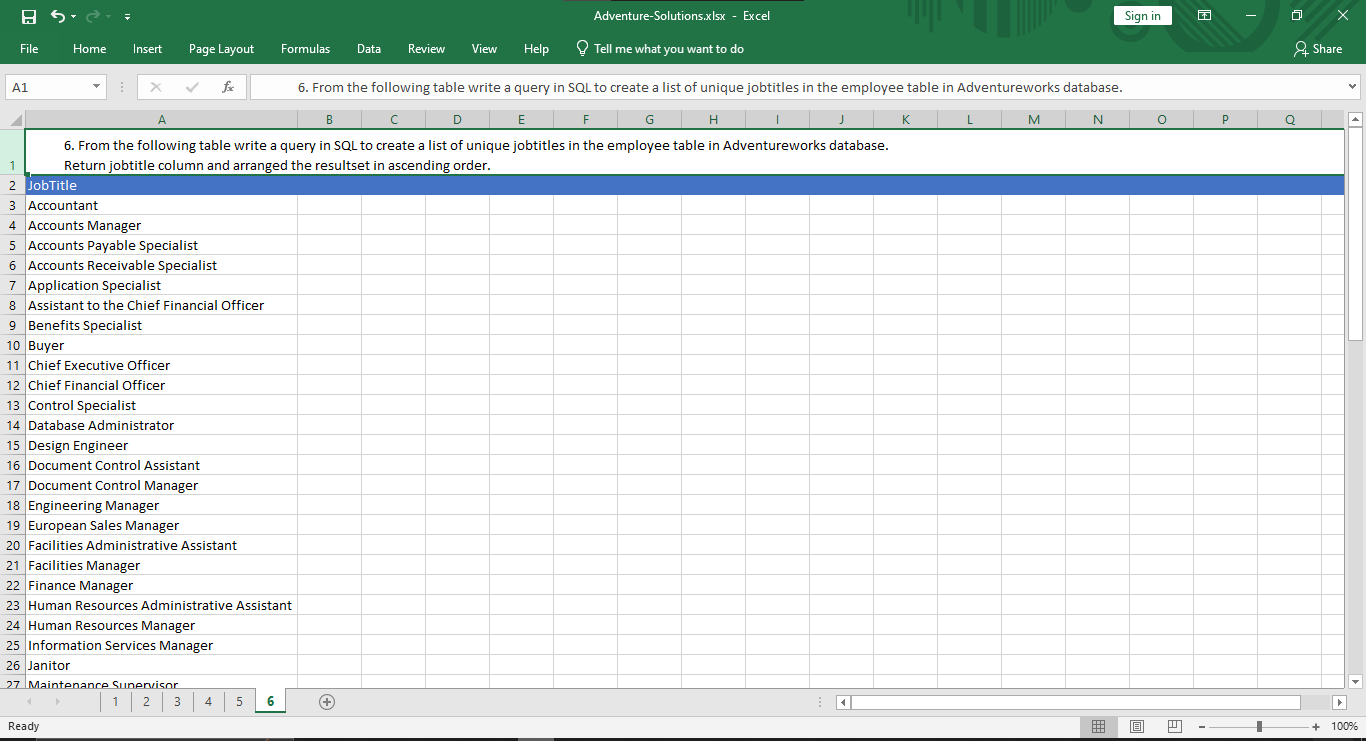
===========================================================================

6. From the following table write a query in SQL to create a list of unique jobtitles in the employee table in Adventureworks database. Return jobtitle column and arranged the resultset in ascending order.

select distinct JobTitle

from HumanResources.Employee

order by JobTitle asc



========================================================================================

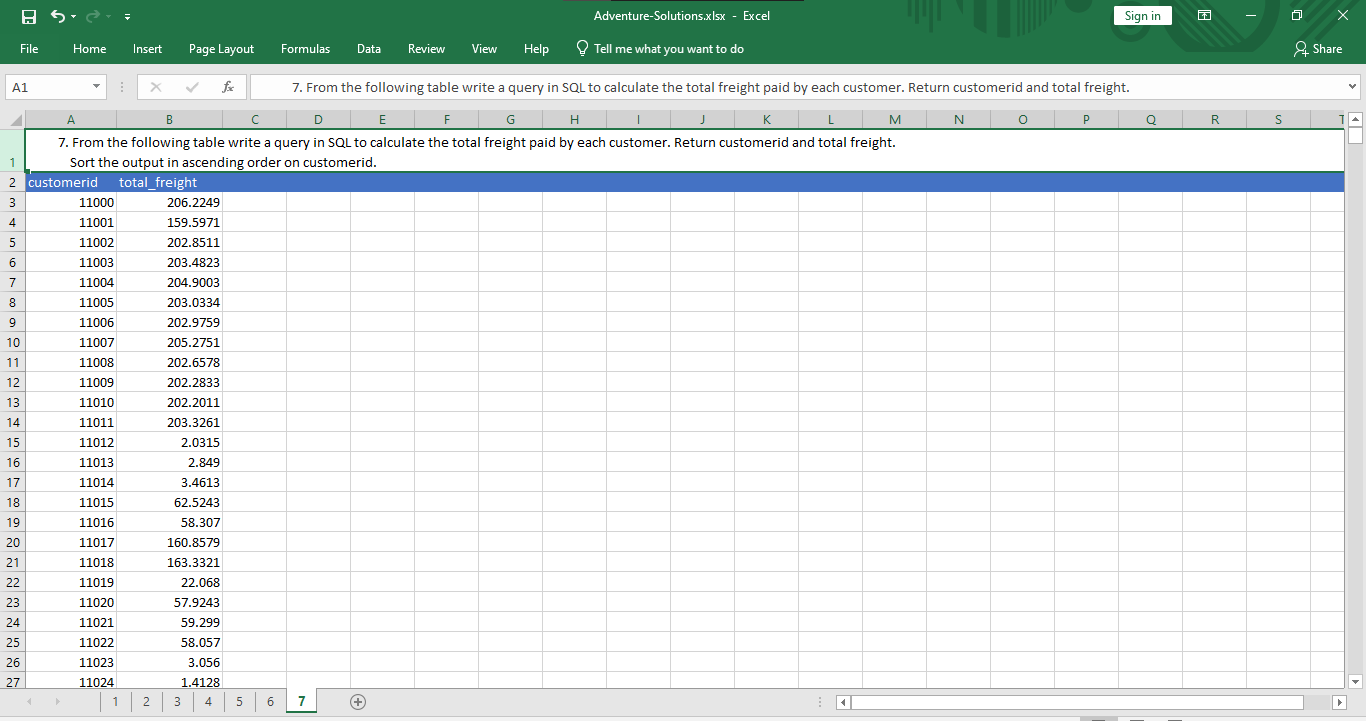
7. From the following table write a query in SQL to calculate the total freight paid by each customer. Return customerid and total freight. Sort the output in ascending order on customerid.

select customerid,sum(Freight) as total\_freight

from sales.salesorderheader

group by CustomerID

order by customerid asc



=========================================================================

8. From the following table write a query in SQL to find the average and the sum of the subtotal for every customer. Return customerid, average and sum of the subtotal. Grouped the result on customerid and salespersonid. Sort the result on customerid column in descending order.

select customerid,

salespersonid,

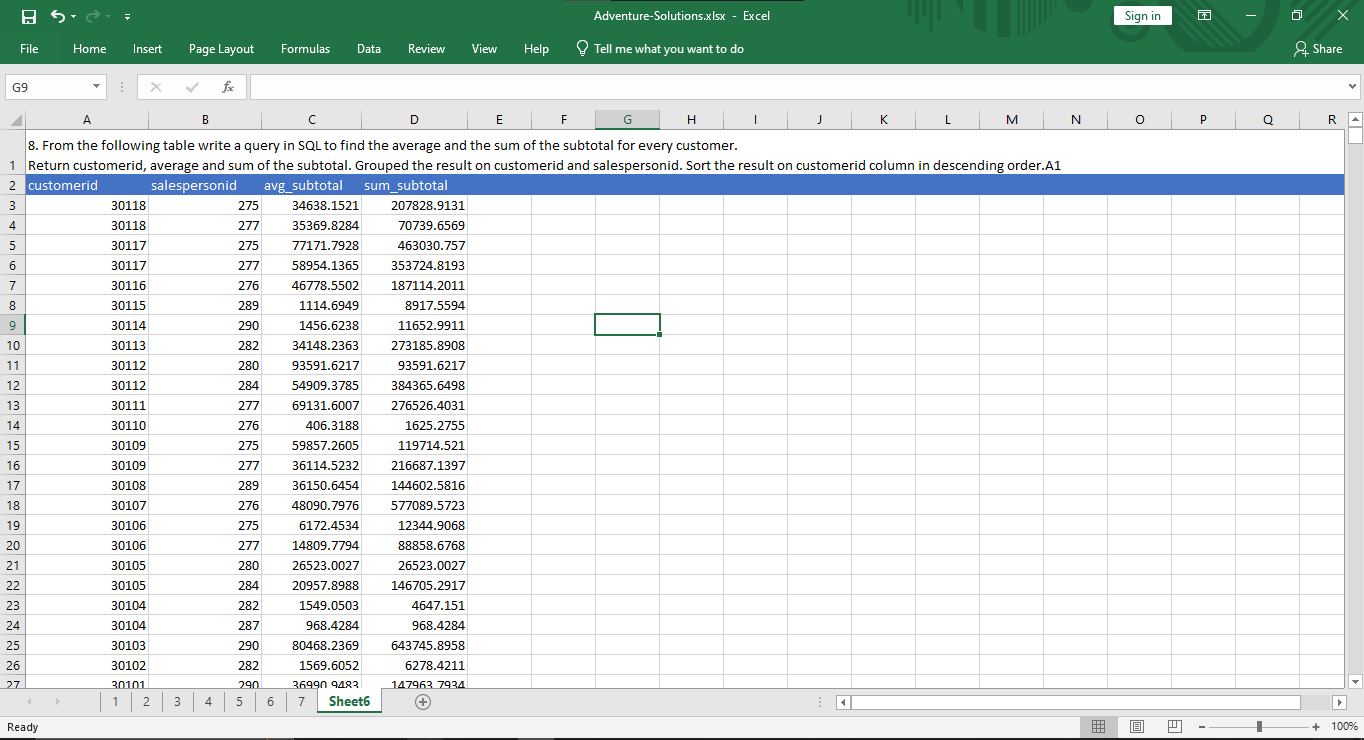
(sum(subtotal)/count(\*)) as avg\_subtotal,

sum(subtotal) as sum\_subtotal

from sales.salesorderheader

group by customerid, salespersonid

order by customerid desc



==========================================================================

9. From the following table write a query in SQL to retrieve total quantity of each productid which are in shelf of 'A' or 'C' or 'H'. Filter the results for sum quantity is more than 500. Return productid and sum of the quantity. Sort the results according to the productid in ascending order.

select ProductID,

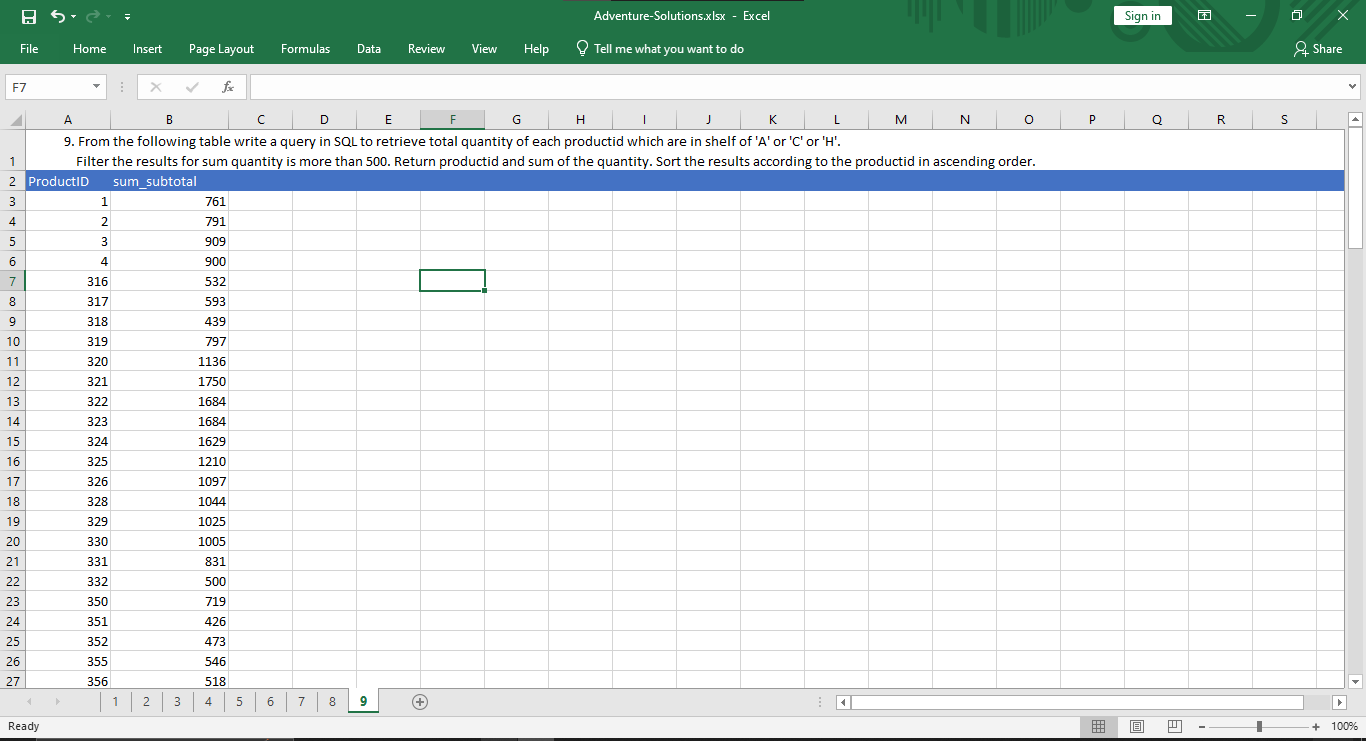
sum(Quantity) as sum\_subtotal

from production.productinventory

where shelf in ('A','C','H')

group by ProductID

order by productid asc



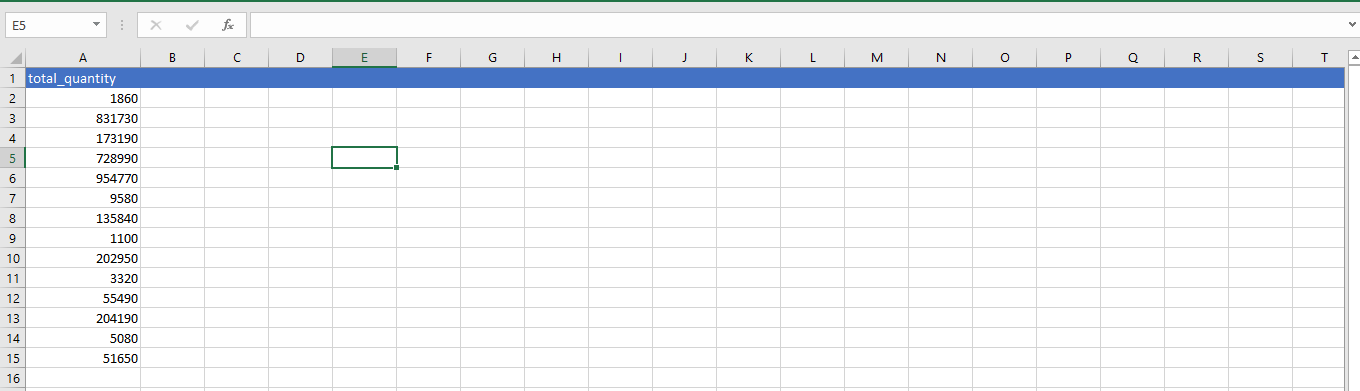
=========================================================================

10. From the following table write a query in SQL to find the total quentity for a group of locationid multiplied by 10.

select sum (Quantity) as sum\_subtotal

from production.productinventory

group by LocationID



===========================================================================

11. From the following tables write a query in SQL to find the persons whose last name starts with letter 'L'.Return BusinessEntityID, FirstName, LastName, and PhoneNumber. Sort the result on lastname and firstname.

select P.BusinessEntityID,

FirstName,

LastName,

F.PhoneNumber as person\_phone

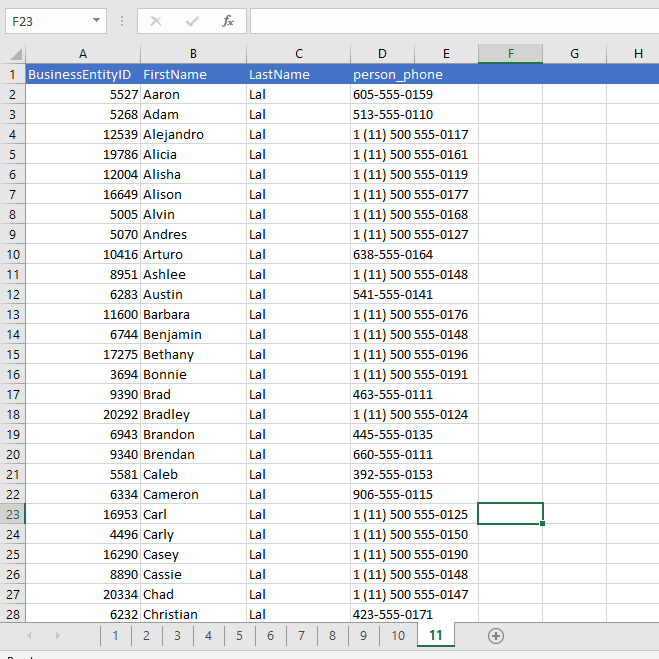
from Person.Person P

join Person.PersonPhone F

on P.BusinessEntityID=F.BusinessEntityID

where LastName like 'L%'

order by LastName,FirstName



==========================================================================

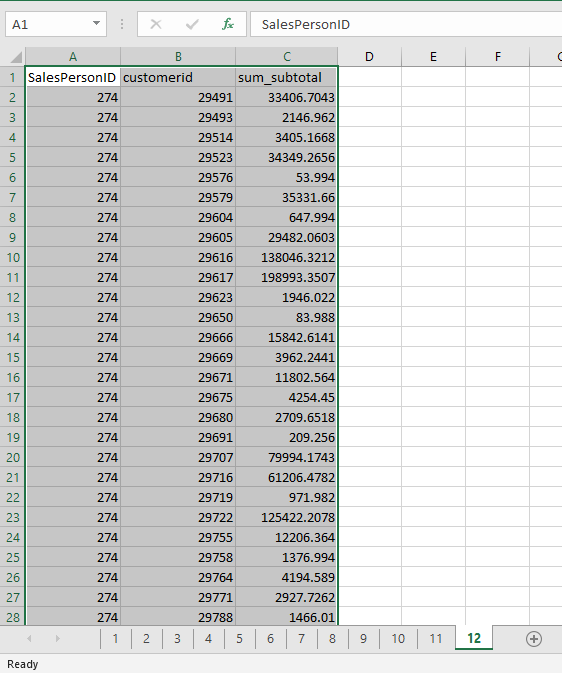
12. From the following table write a query in SQL to find the sum of subtotal column. Group the sum on distinct salespersonid and customerid. Rolls up the results into subtotal and running total. Return salespersonid, customerid and sum of subtotal column i.e. sum\_subtotal.

SELECT salespersonid,customerid,sum(subtotal) AS sum\_subtotal

FROM sales.salesorderheader s

where SalesPersonID is not null

GROUP BY ROLLUP (salespersonid, customerid);



===========================================================================

13. From the following table write a query in SQL to find the sum of the quantity of all combination of group of distinct locationid and shelf column. Return locationid, shelf and sum of quantity as TotalQuantity.

SELECT locationid, shelf, SUM(quantity) AS TotalQuantity

FROM production.productinventory

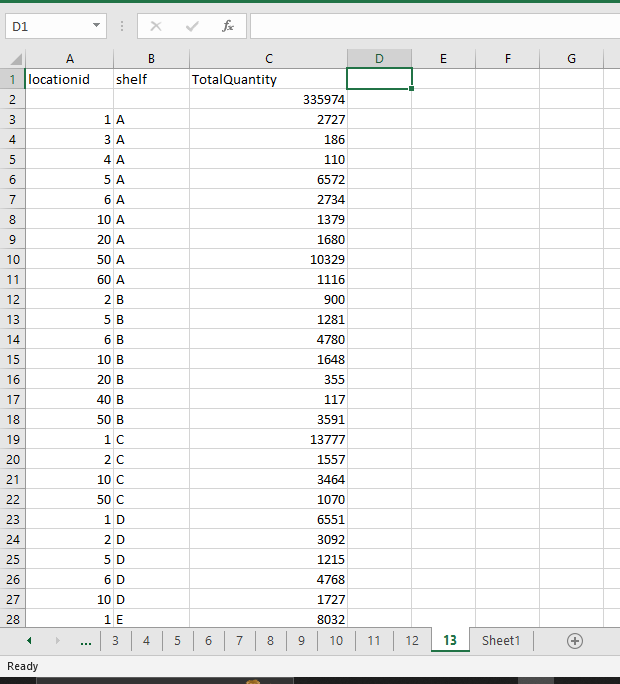
GROUP BY GROUPING SETS

(

(locationid, shelf),

()

)



===========================================================================

14. From the following table write a query in SQL to find the sum of the quantity with subtotal for each locationid. Group the results for all combination of distinct locationid and shelf column. Rolls up the results into subtotal and running total. Return locationid, shelf and sum of quantity as TotalQuantity.

SELECT locationid, shelf, SUM(quantity) AS TotalQuantity

FROM production.productinventory

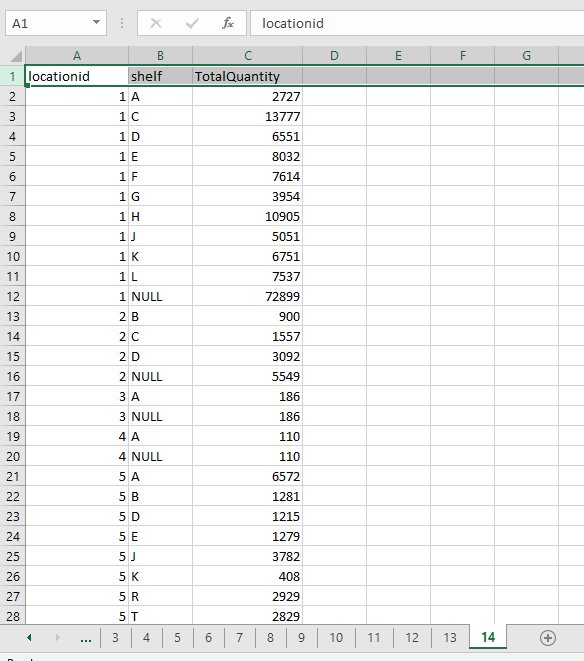
GROUP BY CUBE (locationid, shelf);

-------------------------------------OR----------------------------------------

SELECT locationid, shelf, SUM (quantity) AS TotalQuantity

FROM production.productinventory

GROUP BY ROLLUP (locationid, Shelf)



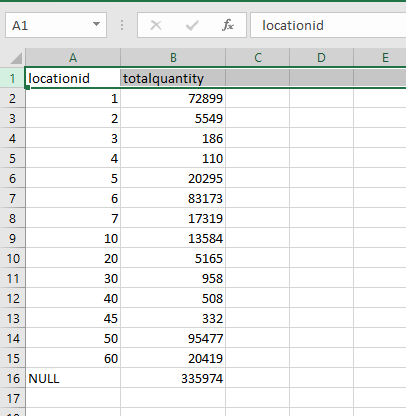
===========================================================================

15. From the following table write a query in SQL to find the total quantity for each locationid and calculate the grand-total for all locations. Return locationid and total quantity. Group the results on locationid.

select locationid,sum(Quantity) totalquantity

from production.productinventory

group by grouping sets (locationid,( ))



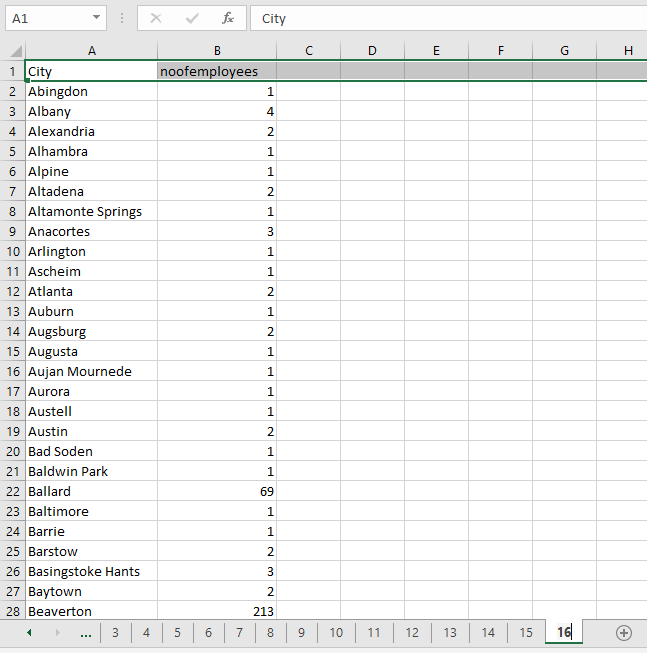
==========================================================================

16. From the following table write a query in SQL to retrieve the number of employees for each City. Return city and number of employees. Sort the result in ascending order on city.

select City, COUNT(City)

from Person.Address

group by City order by City



==========================================================================

17. From the following table write a query in SQL to retrieve the total sales for each year.

Return the year part of order date and total due amount. Sort the result in ascending order on year part of order date.

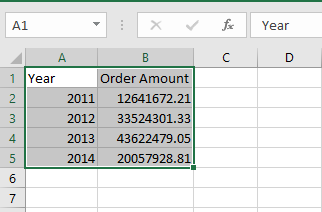
select YEAR(OrderDate) as [Year],

SUM(SubTotal) as [Order Amount]

from Sales.SalesOrderHeader

group by YEAR(OrderDate)

order by YEAR(OrderDate)



==========================================================================

18. From the following table write a query in SQL to retrieve the total sales for each year. Filter the result set for those orders where order year is on or before 2016. Return the year part of orderdate and total due amount. Sort the result in ascending order on year part of order date.

select YEAR(OrderDate) as [Year],

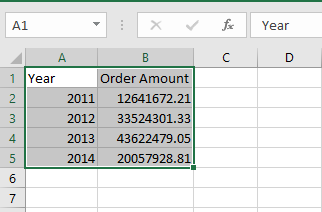
SUM(SubTotal) as [Order Amount]

from Sales.SalesOrderHeader

where YEAR(OrderDate) < '2016'

group by YEAR(OrderDate)

order by YEAR(OrderDate)



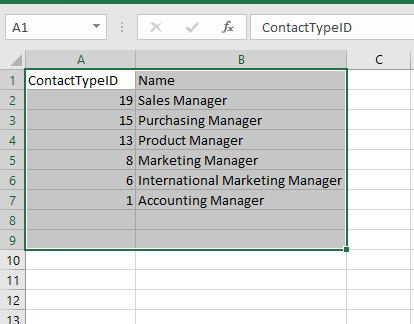
==========================================================================

19. From the following table write a query in SQL to find the contacts who are designated as a manager in various departments.Returns ContactTypeID, name. Sort the result set in descending order.

select ContacttypeId, Name from Person.ContactType

where Name like '%manager'

order by ContactTypeID desc



===========================================================================

20. From the following tables write a query in SQL to make a list of contacts who are designated as 'Purchasing Manager'. Return BusinessEntityID, LastName, and FirstName columns. Sort the result set in ascending order of LastName, and FirstName.

SELECT pp.BusinessEntityID, LastName, FirstName

FROM Person.BusinessEntityContact AS pb

JOIN Person.ContactType AS pc

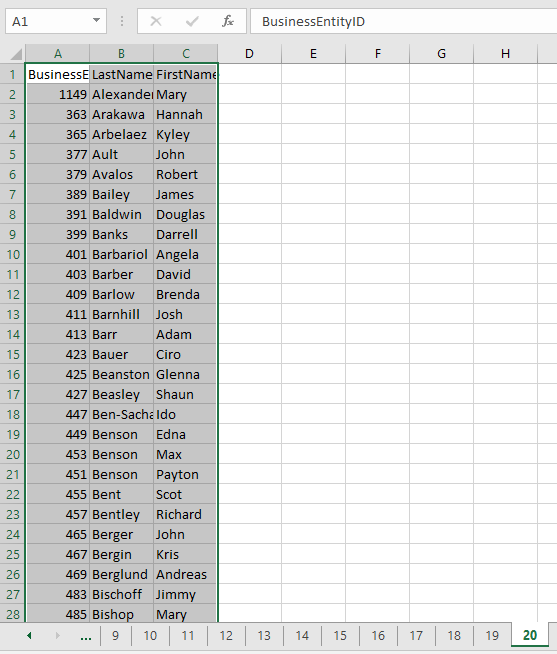
ON pc.ContactTypeID = pb.ContactTypeID

JOIN Person.Person AS pp

ON pp.BusinessEntityID = pb.PersonID

WHERE pc.Name = 'Purchasing Manager'

ORDER BY LastName, FirstName;



21. From the following tables write a query in SQL to retrieve the salesperson for each PostalCode who belongs to a territory and SalesYTD is not zero.Return row numbers of each group of PostalCode, last name, salesytd, postalcode column. Sort the salesytd of each postalcode group in descending order. Sorts the postalcode in ascending order.

select ROWNUMBER () OVER (PARTITION BY pa.PostalCode ORDER BY SalesYTD DESC) AS [Row Number],

pp.LastName,

ss.SalesYTD,

pa.PostalCode

from Sales.SalesPerson ss

join Person.Person pp

on ss.BusinessEntityID=pp.BusinessEntityID

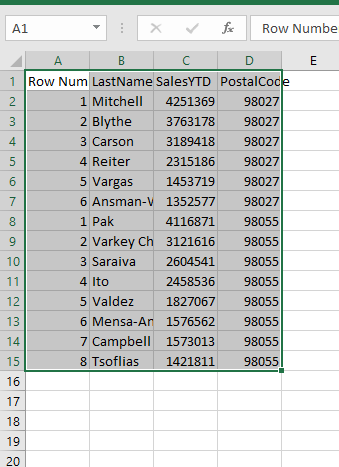
join person.Address pa

on pa.AddressID=ss.BusinessEntityID

WHERE TerritoryID IS NOT NULL

AND SalesYTD <> 0

ORDER BY PostalCode;



--==========================================================================

--22. From the following table write a query in SQL to count the number of contacts for combination of each type and name.

--Filter the output for those who have 100 or more contacts. Return ContactTypeID and ContactTypeName and BusinessEntityContact.

--Sort the result set in descending order on number of contacts.

select pb.contacttypeid,pc.Name ctypename,count(\*) nocontacts

from Person.BusinessEntityContact pb

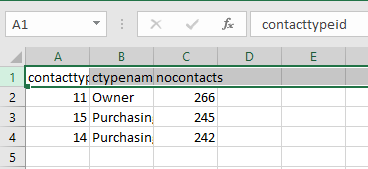
join Person.ContactType pc

on pb.ContactTypeID=pc.ContactTypeID

group by pb.ContactTypeID, pc.Name

having( count(\*) >=100)

order by COUNT(\*) desc



==========================================================================

23. From the following table write a query in SQL to retrieve the RateChangeDate, full name (first name, middle name and last name) and weekly salary (40 hours in a week) of employees. In the output the RateChangeDate should appears in date format. Sort the output in ascending order on NameInFull.

select CAST(HE.RateChangeDate as VARCHAR(10) ) AS FromDate,

CONCAT(PP.LastName, ', ', PP.FirstName, ' ', PP.MiddleName) AS NameInFull,

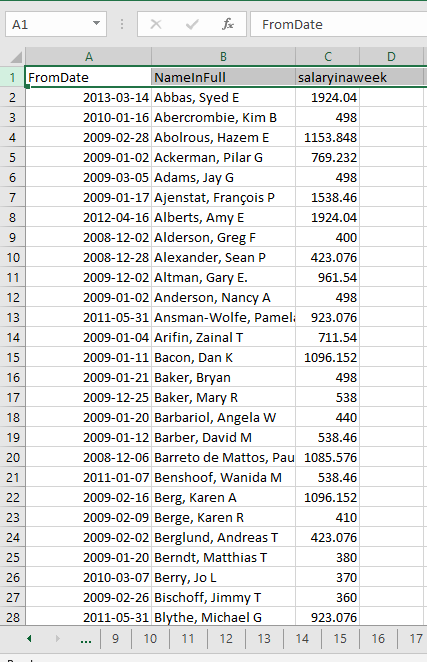
(HE.Rate\*40) salaryinaweek

from HumanResources.EmployeePayHistory HE

join Person.Person PP

on HE.BusinessEntityID=pp.BusinessEntityID

order by nameinfull



==========================================================================

24. From the following tables write a query in SQL to calculate and display the latest weekly salary of each employee.Return RateChangeDate, full name (first name, middle name and last name) and weekly salary (40 hours in a week) of employees Sort the output in ascending order on NameInFull.

select CAST(HE.RateChangeDate as date ) AS FromDate,

CONCAT(PP.LastName, ', ', PP.FirstName, ' ', PP.MiddleName) AS NameInFull,

(HE.Rate\*40) salaryinaweek

from HumanResources.EmployeePayHistory HE

join Person.Person PP

on HE.BusinessEntityID=pp.BusinessEntityID

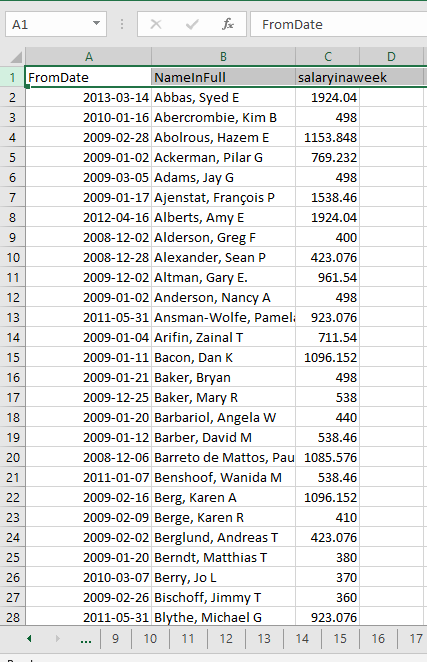
WHERE HE.RateChangeDate = (SELECT MAX(RateChangeDate)

FROM HumanResources.EmployeePayHistory

WHERE BusinessEntityID = HE.BusinessEntityID

)

order by nameinfull



==========================================================================

25. From the following table write a query in SQL to find the sum, average, count, minimum, and maximum order quentity for those orders

whose id are 43659 and 43664. Return SalesOrderID, ProductID, OrderQty, sum, average, count, max, and min order quantity.

select \* from Sales.SalesOrderDetail

select SalesOrderId,

productid,

orderqty,

sum(orderqty) OVER (PARTITION BY ss.SalesOrderId ) [Total Quantity],

avg(orderqty) OVER (PARTITION BY ss.SalesOrderId ) [Avg Quantity],

count(\*) OVER (PARTITION BY ss.SalesOrderId ) [No of Orders],

min(OrderQty) OVER (PARTITION BY ss.SalesOrderId ) [Min Quantity],

max(OrderQty) OVER (PARTITION BY ss.SalesOrderId ) [Max Quantity]

from Sales.SalesOrderDetail ss

where ss.SalesOrderId in (43659 , 43664)



==========================================================================

26. From the following table write a query in SQL to find the sum, average, and number of order quantity for those orders whose ids are 43659 and 43664 and product id starting with '71'. Return SalesOrderID, OrderNumber,ProductID, OrderQty, sum, average, and number of order quantity.

SELECT SalesOrderID AS OrderNumber, ProductID,

OrderQty AS Quantity,

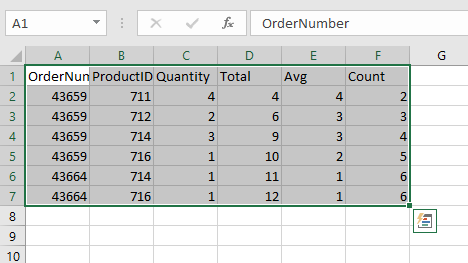
SUM(OrderQty) OVER (ORDER BY SalesOrderID, ProductID) AS Total,

AVG(OrderQty) OVER(PARTITION BY SalesOrderID ORDER BY SalesOrderID, ProductID) AS Avg,

COUNT(OrderQty) OVER(ORDER BY SalesOrderID, ProductID ROWS BETWEEN UNBOUNDED PRECEDING AND 1 FOLLOWING) AS Count

FROM Sales.SalesOrderDetail SS

WHERE SalesOrderID IN(43659,43664) and CAST(ProductID AS varchar(10)) LIKE '71%';



==========================================================================

27. From the following table write a query in SQL to retrieve the total cost of each salesorderID that exceeds 100000. Return SalesOrderID, total cost.

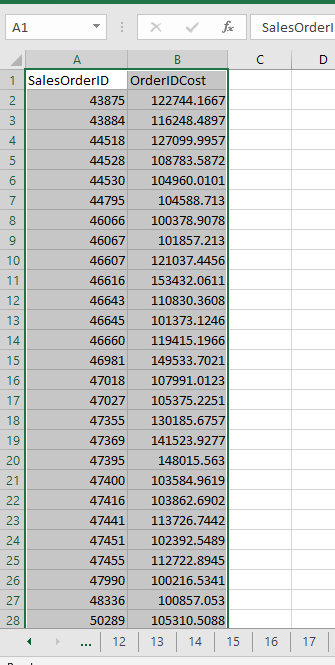
SELECT SalesOrderID, SUM (orderqty\*unitprice) AS OrderIDCost

FROM Sales.SalesOrderDetail

GROUP BY SalesOrderID

HAVING SUM(orderqty\*unitprice) > 100000.00

ORDER BY SalesOrderID;



==========================================================================

28. From the following table write a query in SQL to retrieve products whose names start with 'Lock Washer'. Return product ID, and name and order the result set in ascending order on product ID column.

select productid,name

from Production.Product

where Name like 'Lock Washer%'

order by ProductID



==========================================================================

29. Write a query in SQL to fetch rows from product table and order the result set on an unspecified column listprice.Return product ID, name, and color of the product.

SELECT ProductID, Name, isnull(Color,'')

FROM Production.Product

ORDER BY ListPrice;



==========================================================================

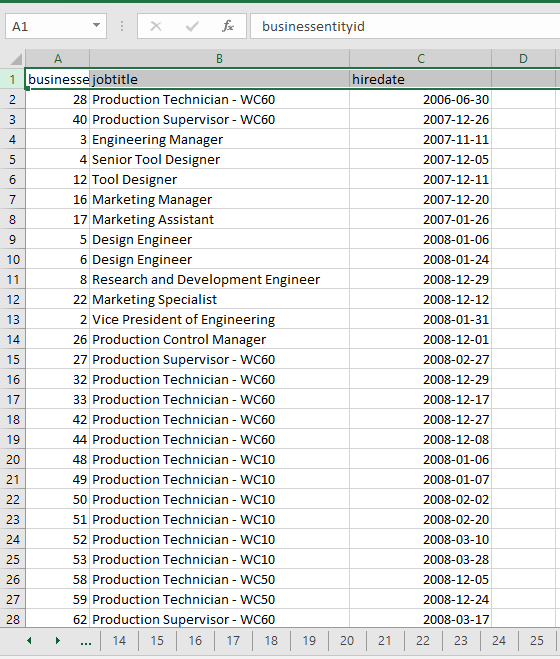
30. From the following table write a query in SQL to retrieve records of employees.

Order the output on year (default ascending order) of hiredate. Return BusinessEntityID, JobTitle, and HireDate.

select businessentityid,jobtitle ,hiredate

from HumanResources.Employee

order by YEAR(hiredate)



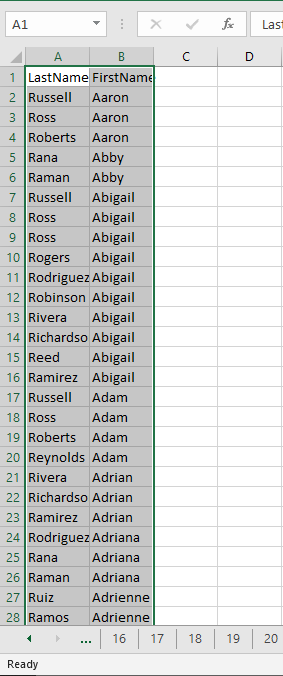
31. From the following table write a query in SQL to retrieve those persons whose last name begins with letter 'R'. Return lastname,and firstname and display the result in ascending order on firstname and descending order on lastname columns.

SELECT LastName, FirstName

FROM Person.Person

WHERE LastName LIKE 'R%'

ORDER BY FirstName ASC, LastName DESC;



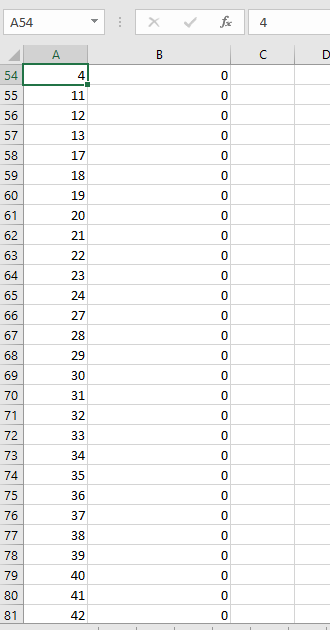
32. From the following table write a query in SQL to ordered the BusinessEntityID column descendingly when SalariedFlag set to 'true' and BusinessEntityID in ascending order when SalariedFlag set to 'false'. Return BusinessEntityID, SalariedFlag columns.

SELECT BusinessEntityID, SalariedFlag

FROM HumanResources.Employee

ORDER BY CASE SalariedFlag WHEN 'true' THEN BusinessEntityID END DESC

,CASE WHEN SalariedFlag ='false' THEN BusinessEntityID END ASC;



==========================================================================

33. From the following table write a query in SQL to set the result in order by the column TerritoryName when the column CountryRegionName is equal to 'United States' and by CountryRegionName for all other rows.

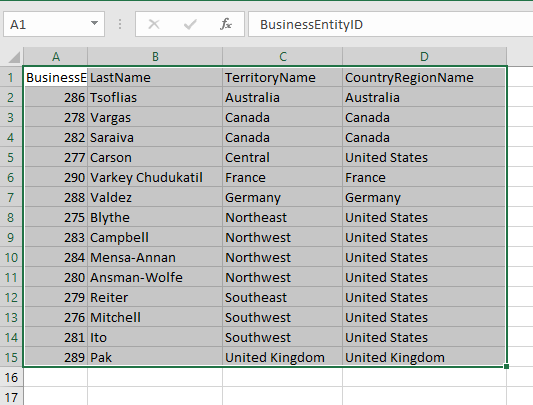
SELECT BusinessEntityID, LastName, TerritoryName, CountryRegionName

FROM Sales.vSalesPerson

WHERE TerritoryName IS NOT NULL

ORDER BY CASE CountryRegionName WHEN 'United States' THEN TerritoryName

ELSE CountryRegionName END;



==========================================================================

34. From the following table write a query in SQL to find those persons who lives in a territory and the value of salesytd except 0. Return first name, last name,row number as 'Row Number', 'Rank', 'Dense Rank' and NTILE as 'Quartile', salesytd and postalcode. Order the output on postalcode column.

SELECT p.FirstName, p.LastName

,ROW\_NUMBER() OVER (ORDER BY a.PostalCode) AS "Row Number"

,RANK() OVER (ORDER BY a.PostalCode) AS "Rank"

,DENSE\_RANK() OVER (ORDER BY a.PostalCode) AS "Dense Rank"

,NTILE(4) OVER (ORDER BY a.PostalCode) AS "Quartile"

,s.SalesYTD, a.PostalCode

FROM Sales.SalesPerson AS s

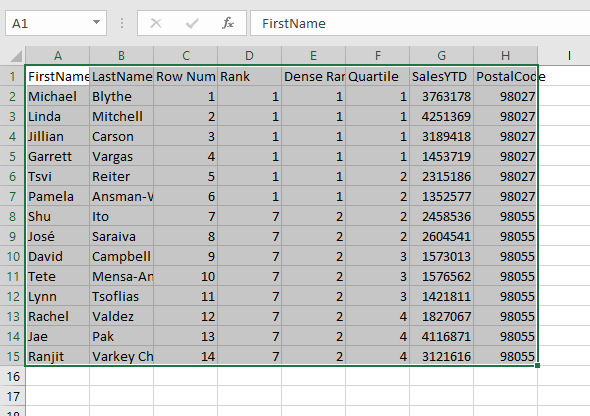
INNER JOIN Person.Person AS p

ON s.BusinessEntityID = p.BusinessEntityID

INNER JOIN Person.Address AS a

ON a.AddressID = p.BusinessEntityID

WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0;



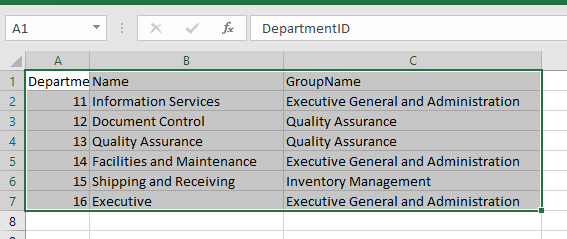
==========================================================================

35From the following table write a query in SQL to skip the first 10 rows from the sorted result set and return all remaining rows.

SELECT DepartmentID, Name, GroupName

FROM HumanResources.Department

ORDER BY DepartmentID OFFSET 10 ROWS;



==========================================================================

36. From the following table write a query in SQL to skip the first 5 rows and return the next 5 rows from the sorted result set.

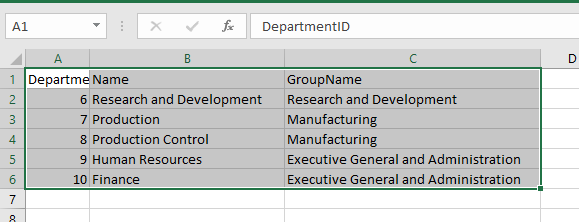
SELECT DepartmentID, Name, GroupName

FROM HumanResources.Department

ORDER BY DepartmentID

OFFSET 5 ROWS

FETCH NEXT 5 ROWS ONLY;



==========================================================================

37. From the following table write a query in SQL to list all the products that are Red or Blue in color. Return name, color and listprice.Sorts this result by the column listprice.

SELECT Name, Color, ListPrice

FROM Production.Product

WHERE Color = 'Blue'

UNION ALL

SELECT Name, Color, ListPrice

FROM Production.Product

WHERE Color = 'Red'

ORDER BY ListPrice ASC;



==========================================================================

38. Create a SQL query from the SalesOrderDetail table to retrieve the product name and any associated sales orders.Additionally, it returns any sales orders that don't have any items mentioned in the Product table as well as any products that have sales orders other than those that are listed there. Return product name, salesorderid. Sort the result set on product name column.

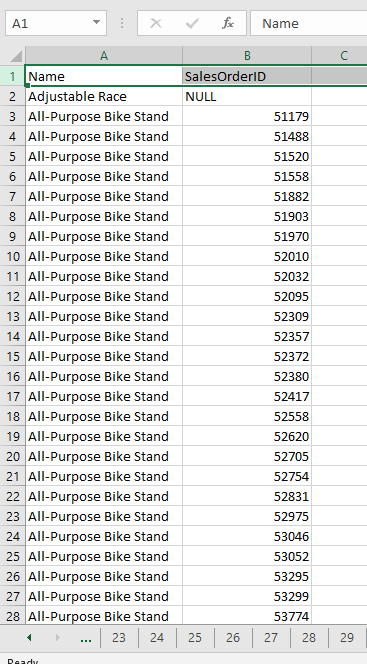
SELECT pp.Name, SS.SalesOrderID

FROM Production.Product AS pp

FULL JOIN Sales.SalesOrderDetail AS SS

ON pp.ProductID = SS.ProductID

ORDER BY pp.Name ;



==========================================================================

39. From the following table write a SQL query to retrieve the product name and salesorderid. Both ordered and unordered products are included in the result set.

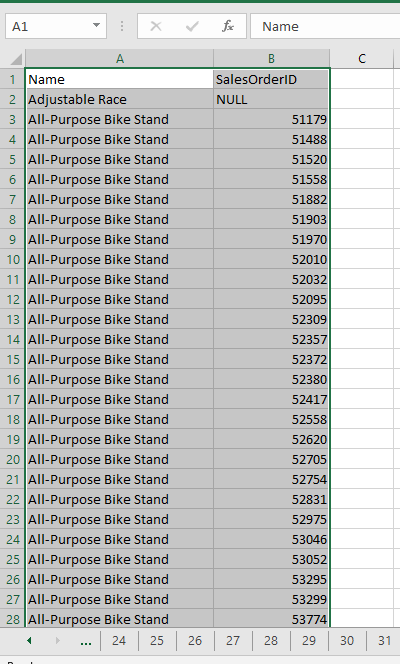
SELECT pp.Name, ss.SalesOrderID

FROM Production.Product AS pp

LEFT JOIN Sales.SalesOrderDetail AS ss

ON pp.ProductID = ss.ProductID

ORDER BY pp.Name ;



==========================================================================

40. From the following tables write a SQL query to get all product names and sales order IDs. Order the result set on product name column.

SELECT pp.Name, ss.SalesOrderID

FROM Production.Product AS pp

JOIN Sales.SalesOrderDetail AS ss

ON pp.ProductID = ss.ProductID

ORDER BY pp.Name ;



==========================================================================

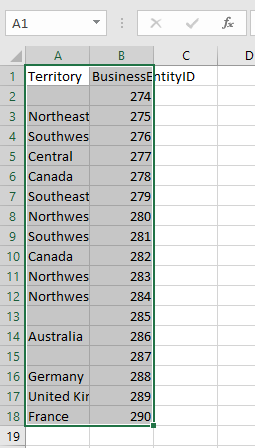
41. From the following tables write a SQL query to retrieve the territory name and BusinessEntityID. The result set includes all salespeople, regardless of whether or not they are assigned a territory.

SELECT isnull(st.Name,'') AS Territory, sp.BusinessEntityID

FROM Sales.SalesTerritory AS st

RIGHT JOIN Sales.SalesPerson AS sp

ON st.TerritoryID = sp.TerritoryID ;



==========================================================================

42. Write a query in SQL to find the employee's full name (firstname and lastname) and city from the following tables.Order the result set on lastname then by firstname.

SELECT concat(pp.FirstName,' ', pp.LastName) AS Name, sub.City

FROM Person.Person AS pp

JOIN HumanResources.Employee HE

ON pp.BusinessEntityID = HE.BusinessEntityID

JOIN (SELECT pb.BusinessEntityID, pa.City

FROM Person.Address AS pa

JOIN Person.BusinessEntityAddress AS pb

ON pa.AddressID = pb.AddressID

) AS sub

ON pp.BusinessEntityID = sub.BusinessEntityID

ORDER BY pp.LastName, pp.FirstName;



==========================================================================

43. Write a SQL query to return the businessentityid,firstname and lastname columns of all persons in the person table (derived table) with persontype is 'IN' and the last name is 'Adams'. Sort the result set in ascending order on firstname. A SELECT statement after the FROM clause is a derived table.

select businessentityid,firstname,lastname

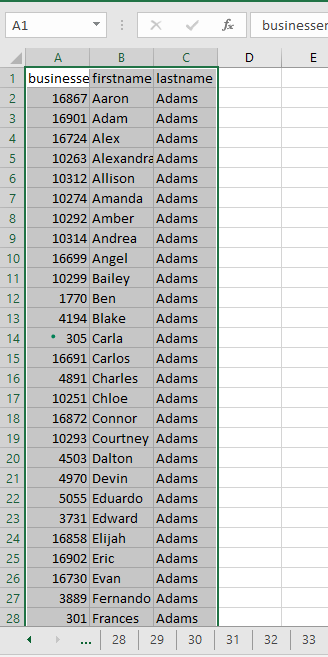
from ( select businessentityid,firstname,lastname

from Person.Person

where lastname in('Adams')

) as derived

order by FirstName



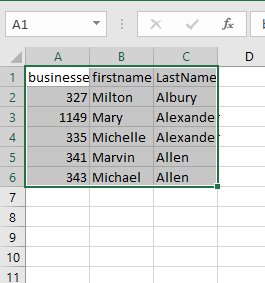
==========================================================================

44. Create a SQL query to retrieve individuals from the following table with a businessentityid inside 1500, a lastname starting with 'Al', and a firstname starting with 'M'.

SELECT businessentityid, firstname,LastName

FROM person.person

WHERE businessentityid <= 1500 AND LastName LIKE 'Al%' AND FirstName LIKE 'M%';



==========================================================================

45. Write a SQL query to find the productid, name, and colour of the items 'Blade', 'Crown Race' and 'AWC Logo Cap' using a derived table with multiple values.

select productid, name, isnull(Color,'')

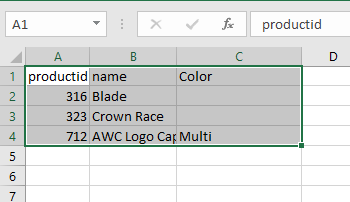
from (select productid, name, Color

from Production.Product

where name in ('Blade', 'Crown Race' , 'AWC Logo Cap' )

) as derv

order by ProductID



==========================================================================

46. Create a SQL query to display the total number of sales orders each sales representative receives annually. Sort the result set by SalesPersonID and then by the date component of the orderdate in ascending order. Return the year component of the OrderDate, SalesPersonID, and SalesOrderID.

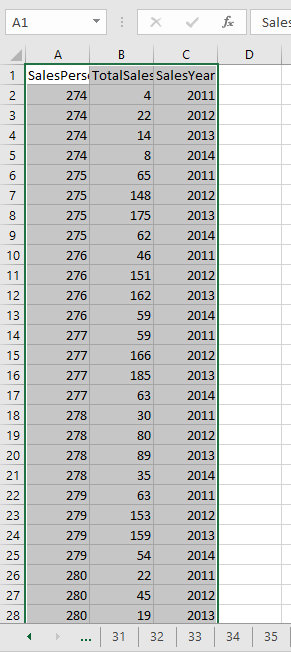
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, year(OrderDate) AS SalesYear

FROM Sales.SalesOrderHeader

WHERE SalesPersonID IS NOT NULL

GROUP BY year(OrderDate), SalesPersonID

ORDER BY SalesPersonID, SalesYear



==========================================================================

47. From the following table write a query in SQL to find the average number of sales orders for all the years of the sales representatives.

WITH Sales\_CTE (SalesPersonID, NumberOfOrders)

AS

(

SELECT SalesPersonID, COUNT(\*)

FROM Sales.SalesOrderHeader

WHERE SalesPersonID IS NOT NULL

GROUP BY SalesPersonID

)

SELECT AVG(NumberOfOrders) AS "Average Sales Per Person"

FROM Sales\_CTE;

----------------------------OR-------------------------------

SELECT AVG(NumberOfOrders) AS "Average Sales Per Person"

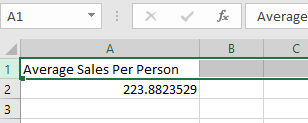
FROM (SELECT SalesPersonID, COUNT(\*) AS NumberOfOrders

FROM Sales.SalesOrderHeader

WHERE SalesPersonID IS NOT NULL

GROUP BY SalesPersonID

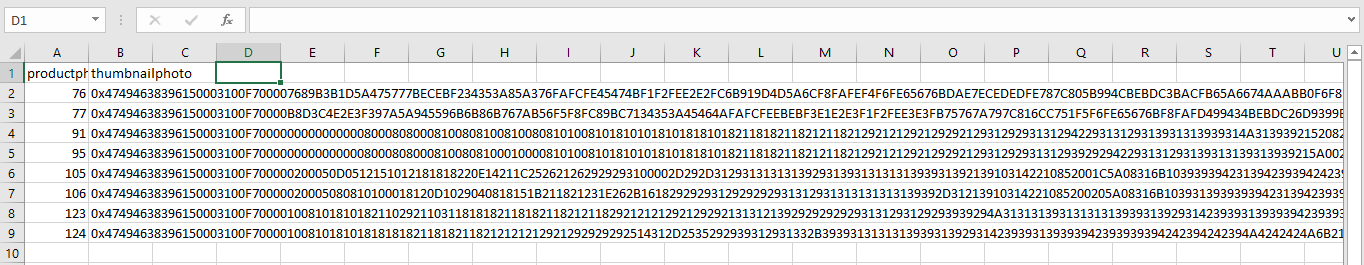
) derived



--==========================================================================

48. Write a SQL query on the following table to retrieve records with the characters green\_ in the LargePhotoFileName field. The following table's columns must all be returned.

select productphotoid,thumbnailphoto from Production.ProductPhoto where LargePhotoFileName like '%green\_%'



==========================================================================

49. Write a SQL query to retrieve the mailing address for any company that is outside the United States (US) and in a city whose name starts with Pa. Return Addressline1, Addressline2, city, postalcode, countryregioncode columns.

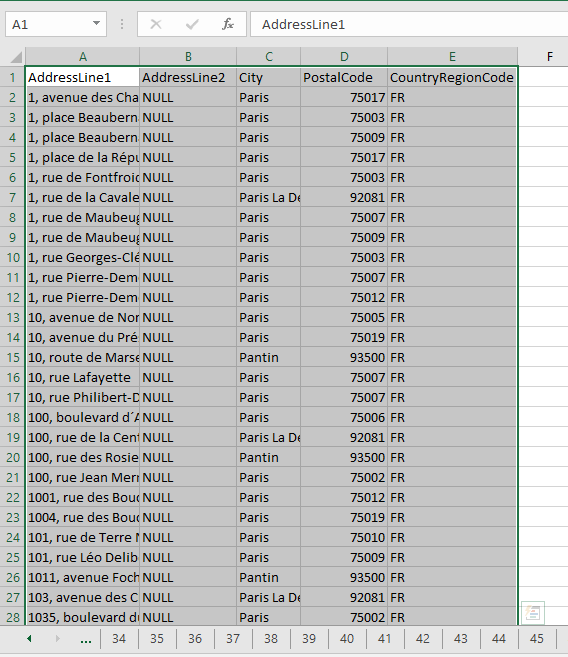
SELECT AddressLine1, AddressLine2, City, PostalCode, CountryRegionCode

FROM Person.Address AS a

JOIN Person.StateProvince AS s ON a.StateProvinceID = s.StateProvinceID

WHERE CountryRegionCode NOT IN ('US')

AND City LIKE 'Pa%';



==========================================================================

50. From the following table write a query in SQL to fetch first twenty rows. Return jobtitle, hiredate.

Order the result set on hiredate column in descending order.

SELECT HE.JobTitle, HE.HireDate

FROM HumanResources.Employee HE

ORDER BY HE.HireDate desc

OFFSET 0 ROWS

FETCH FIRST 20 ROWS ONLY;



==========================================================================

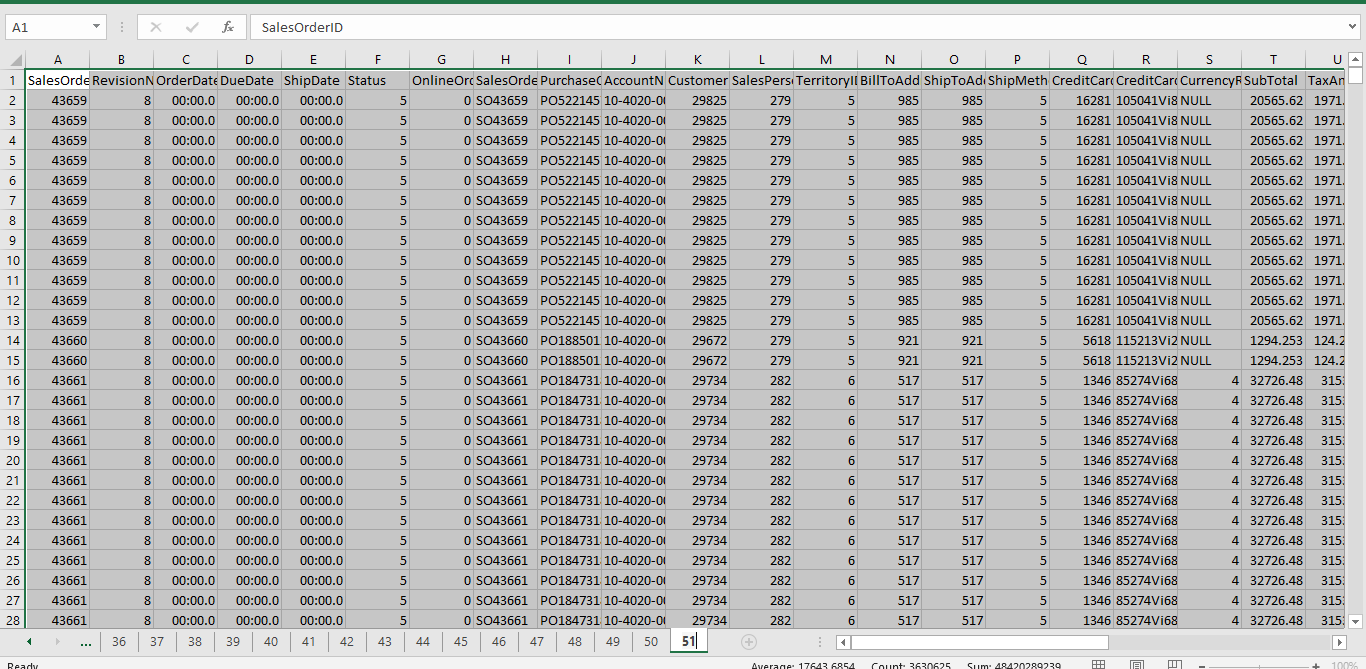
51. From the following tables write a SQL query to retrieve the orders with orderqtys greater than 5 or unitpricediscount less than 1000,and totaldues greater than 100. Return all the columns from the tables.

select \* from Sales.SalesOrderHeader SSOH

join Sales.SalesOrderDetail SSOD

on SSOH.SalesOrderID= SSOD.SalesOrderID

where (SSOD.OrderQty>5 or SSOD.UnitPriceDiscount<1000) and SSOH.TotalDue>100

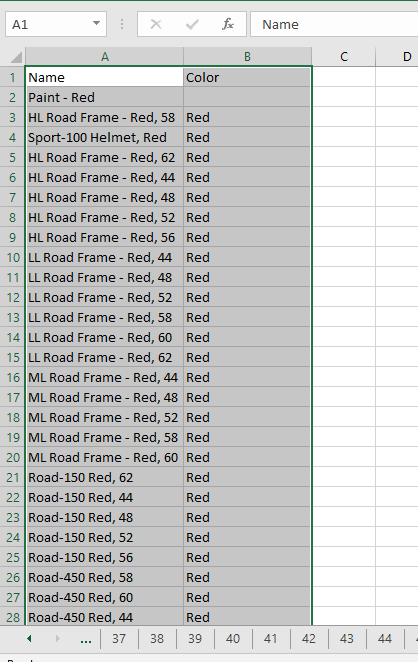


52. From the following table write a query in SQL that searches for the word 'red' in the name column. Return name, and color columns from the table.

SELECT Name, isnull(Color,'') Color

FROM Production.Product

HAVING name like ('%red%');



==========================================================================

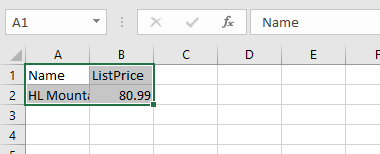
53. From the following table write a query in SQL to find all the products with a price of $80.99 that contain the word Mountain. Return name, and listprice columns from the table.

SELECT Name, ListPrice

FROM Production.Product

HAVING ListPrice = 80.99

and name like ('%Mountain%');



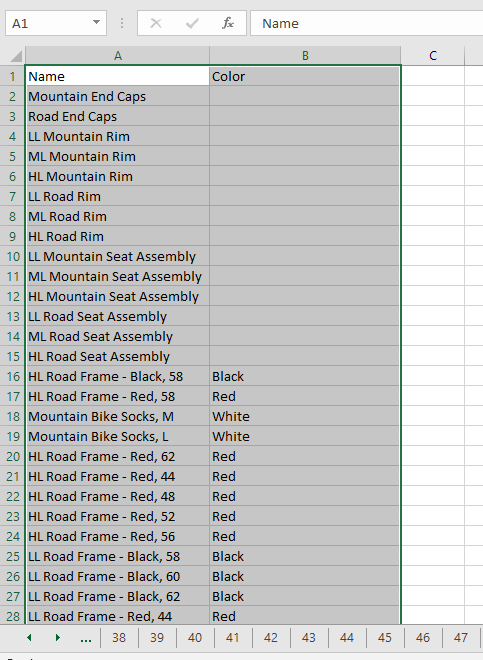
==========================================================================

54. From the following table write a query in SQL to retrieve all the products that contain either the phrase Mountain or Road.Return name, and color columns.

SELECT Name, isnull(Color,'') Color

FROM Production.Product

HAVING name like ('%Mountain%') or name like ('%Road%');



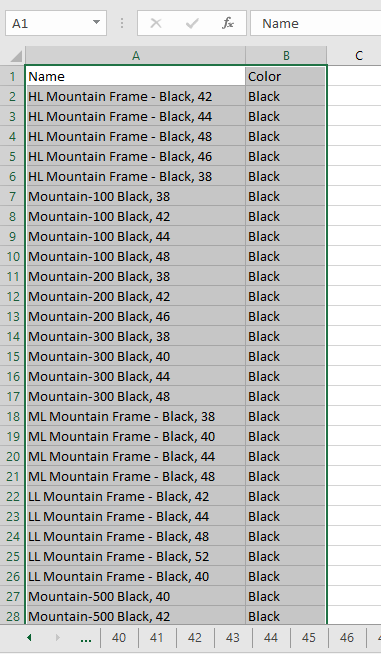
==========================================================================

55. From the following table write a query in SQL to search for name which contains both the word 'Mountain' and the word 'Black'. Return Name and color.

SELECT Name, isnull(Color,'') Color

FROM Production.Product

WHERE name like ('%Mountain%') and name like ('%Black%');



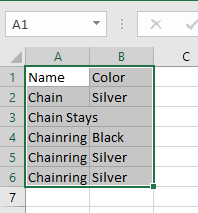
==========================================================================

56. From the following table write a query in SQL to return all the product names with at least one word starting with the prefix chain in the Name column.

SELECT Name, isnull(Color,'') Color

FROM Production.Product

WHERE name like ('%chain%');



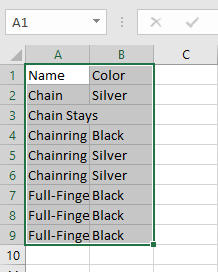
==========================================================================

57. From the following table write a query in SQL to return all category descriptions containing strings with prefixes of either chain or full.

SELECT Name, isnull(Color,'') Color

FROM Production.Product

WHERE name like ('%chain%') or name like ('%full%');



==========================================================================

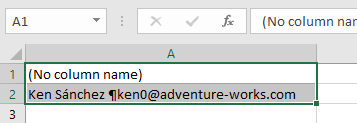
58. From the following table write a SQL query to output an employee's name and email address, separated by a new line character.

SELECT concat(pp.FirstName,' ', pp.LastName) + ' '+'¶'+ pe.EmailAddress

FROM Person.Person pp

INNER JOIN Person.EmailAddress pe ON pp.BusinessEntityID = pe.BusinessEntityID

where pp.BusinessEntityID = 1;



==========================================================================

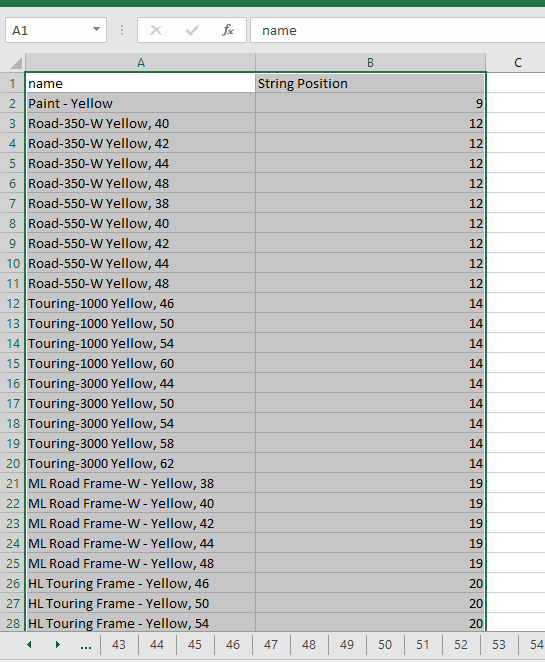
59. From the following table write a SQL query to locate the position of the string "yellow" where it appears in the product name.

SELECT name, CHARINDEX('yellow', name) as "String Position"

from production.product

where CHARINDEX('yellow', name)>0

order by CHARINDEX('yellow', name)

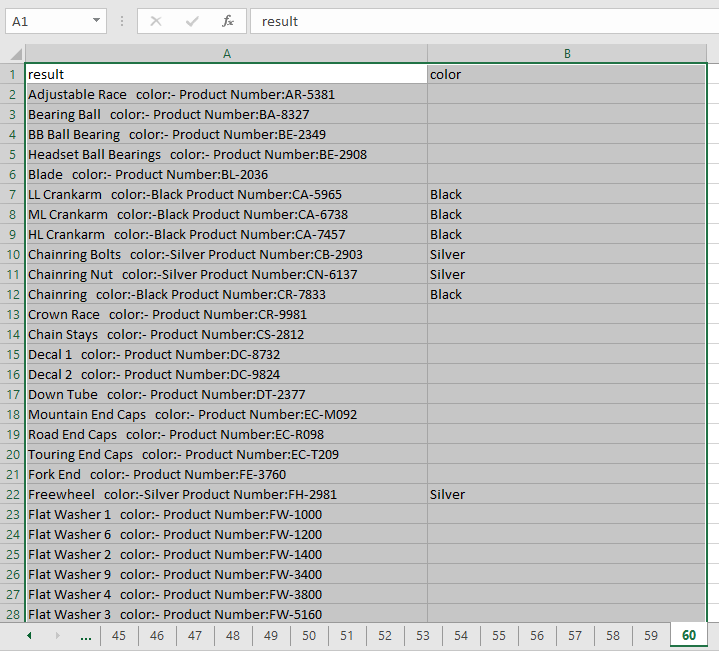


==========================================================================

60 From the following table write a query in SQL to concatenate the name, color, and productnumber columns.

SELECT CONCAT(name, ' color:-',color,' Product Number:', productnumber ) AS result, isnull(color) color

FROM production.product;



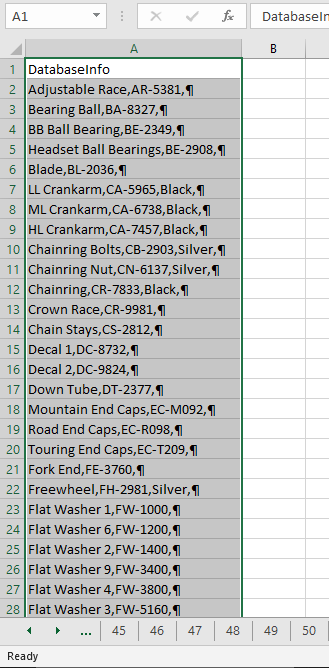
==========================================================================

61 Write a SQL query that concatenate the columns name, productnumber, colour, and a new line character from

--the following table, each separated by a specified character.

SELECT CONCAT (',', name, productnumber, color,'¶') AS DatabaseInfo

FROM production.product;



==========================================================================

62 From the following table write a query in SQL to return the five leftmost characters of each product name.

SELECT LEFT(Name, 5) as [LEFT]

FROM Production.Product

ORDER BY ProductID;

-----------OR------------------

SELECT SUBSTRING(Name,1,5)

FROM Production.Product

ORDER BY ProductID;



==========================================================================

63 From the following table write a query in SQL to select the number of characters and the data in FirstName for people located in Australia.

SELECT len(FirstName) AS [Length], FirstName, LastName

FROM Sales.vIndividualCustomer

WHERE CountryRegionName = 'Australia';



64 From the following tables write a query in SQL to return the number of characters in the column FirstName and the first and last name of contacts located in Australia.

SELECT distinct len(FirstName) AS [FNameLength], FirstName, LastName

FROM Sales.vstorewithcontacts AS SVC

JOIN Sales.vstorewithaddresses AS SVA

ON SVC.businessentityid = SVA.businessentityid

WHERE CountryRegionName = 'Australia'



==========================================================================

65 From the following table write a query in SQL to select product names that have prices between $1000.00 and $1220.00. Return product name as Lower, Upper, and also LowerUpper.

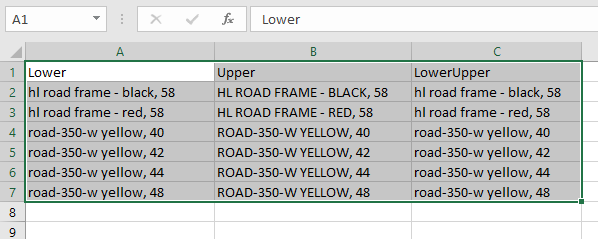
SELECT LOWER(Name) AS [Lower],

UPPER(Name) AS Upper,

LOWER(UPPER(Name)) As LowerUpper

FROM production.Product

WHERE standardcost between 1000.00 and 1220.00;

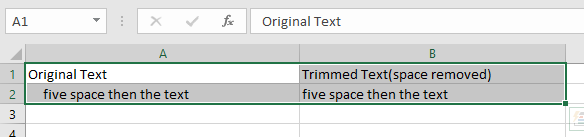


==========================================================================

66 Write a query in SQL to remove the spaces from the beginning of a string.

select ' five space then the text' as [Original Text],

LTRIM (' five space then the text') as [Trimmed Text(space removed)]

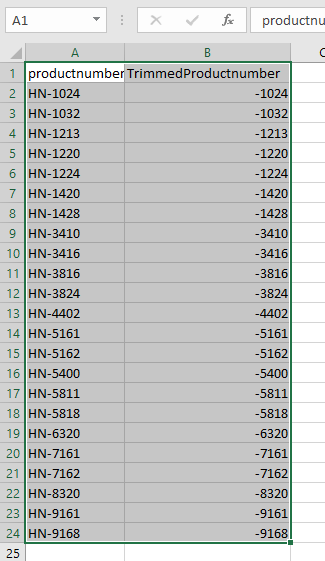


67 From the following table write a query in SQL to remove the substring 'HN' from the start of the column productnumber. Filter the results to only show those productnumbers that start with "HN". Return original productnumber column and 'TrimmedProductnumber'.

SELECT productnumber,substring(productnumber,3,len(productnumber)) as [TrimmedProductnumber]

from production.product

where productnumber like 'HN%';



==========================================================================

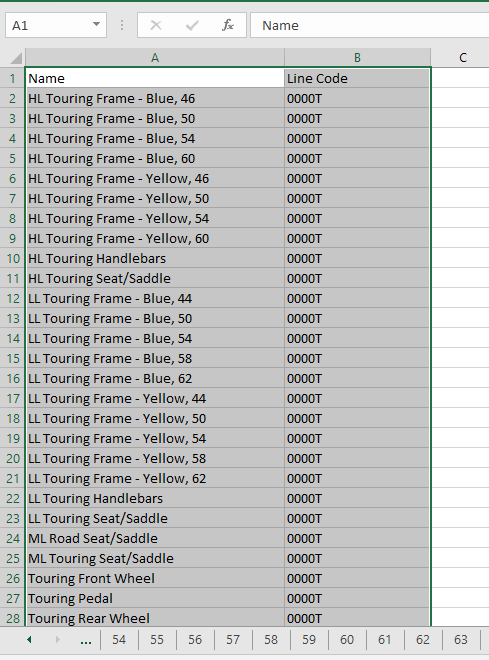
68 From the following table write a query in SQL to repeat a 0 character four times in front of a production line for production line 'T'.

SELECT Name, concat(REPLICATE('0', 4) , ProductLine) AS [Line Code]

FROM Production.Product

WHERE ProductLine like 'T'

ORDER BY Name;



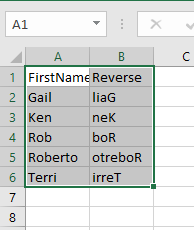
==========================================================================

69 From the following table write a SQL query to retrieve all contact first names with the characters inverted for people whose businessentityid is less than 6.

SELECT FirstName, REVERSE(FirstName) AS Reverse

FROM Person.Person

ORDER BY FirstName;



==========================================================================

70 From the following table write a query in SQL to return the eight rightmost characters of each name of the product. Also return name,productnumber column. Sort the result set in ascending order on productnumber.

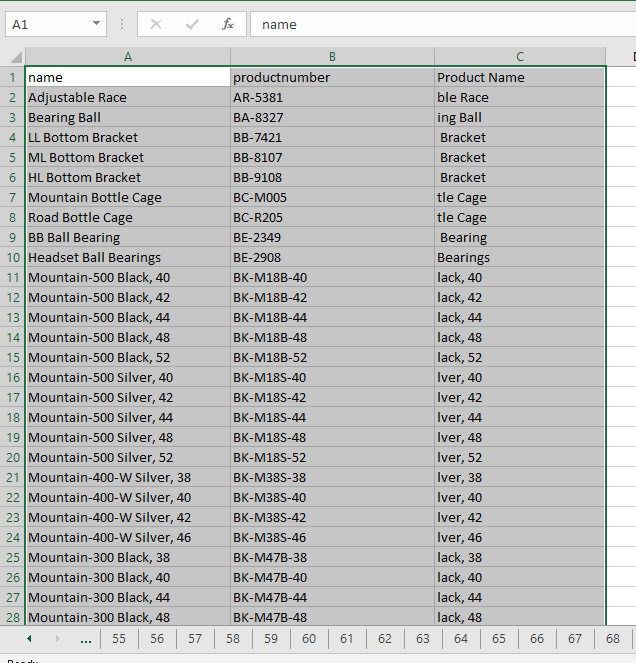
SELECT name,

productnumber,

RIGHT (name, 8) AS [Product Name]

FROM production.product

ORDER BY productnumber;

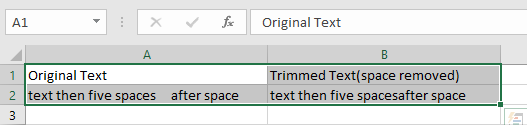


==========================================================================

71 Write a query in SQL to remove the spaces at the end of a string.

SELECT CONCAT ('text then five spaces ','after space') as [Original Text],

CONCAT (RTRIM ('text then five spaces '),'after space') as [Trimmed Text(space removed)];



==========================================================================

72 From the following table write a query in SQL to fetch the rows for the product name ends with the letter 'S' or 'M' or 'L'. Return productnumber and name.

SELECT productnumber, name

FROM production.product

WHERE RIGHT (name,1) in ('S','M','L');

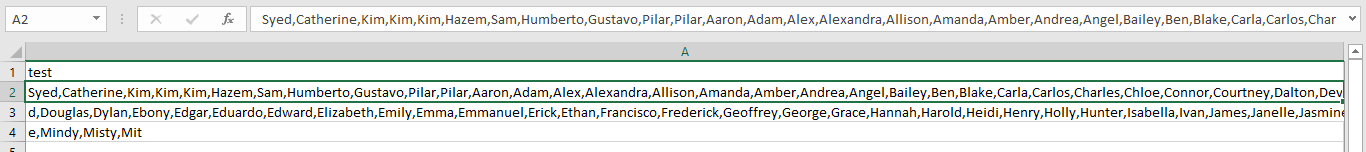


==========================================================================

73 From the following table write a query in SQL to replace null values with 'N/A' and return the names separated by commas in a single row.

SELECT STRING\_AGG(CONVERT( NVARCHAR(max),coalesce(firstname, ' N/A')),',' ) as test

FROM Person.Person;



==========================================================================

74 From the following table write a query in SQL to return the names and modified date separated by commas in a single row.

SELECT STRING\_AGG(

CONVERT(

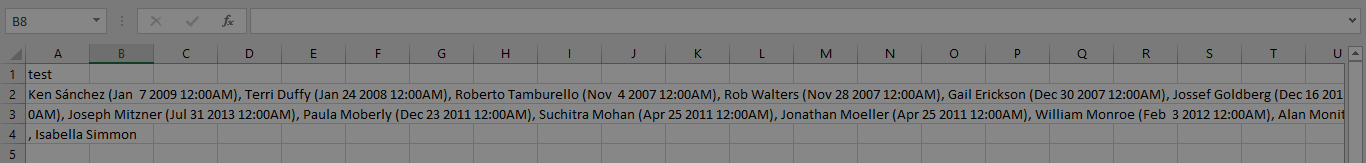
NVARCHAR(max),

CONCAT(FirstName, ' ', LastName, ' (', ModifiedDate, ')'

)),', '

) AS test

FROM Person.Person;



75 From the following table write a query in SQL to find the email addresses of employees and groups them by city. Return top ten rows.

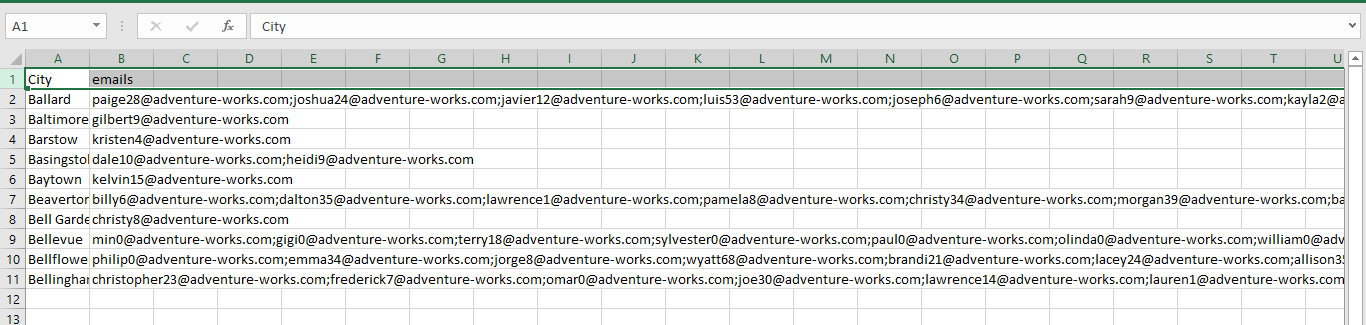
SELECT TOP 10 City, STRING\_AGG(cast(EmailAddress as varchar(max)), ';') AS emails

FROM Person.BusinessEntityAddress AS PB

JOIN Person.Address AS PA ON PB.AddressID = PA.AddressID

JOIN Person.EmailAddress AS PE ON PB.BusinessEntityID = PE.BusinessEntityID

GROUP BY City



76 From the following table write a query in SQL to create a new job title called "Production Assistant" in place of "Production Supervisor".

SELECT jobtitle ,REPLACE(jobtitle, 'Production Supervisor', 'Production Assistant') as "New Jobtitle"

FROM humanresources.employee e

WHERE jobtitle like '%Production Supervisor%'



==========================================================================

77 From the following table write a SQL query to retrieve all the employees whose job titles begin with "Sales". Return firstname, middlename, lastname and jobtitle column.

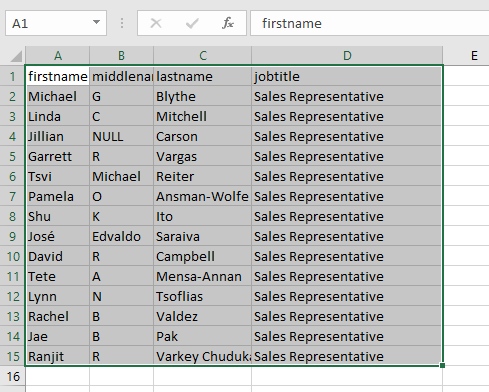
SELECT PP.firstname, PP.middlename, PP.lastname, HE.jobtitle

FROM person.person PP

JOIN humanresources.employee HE

ON PP.businessentityid=HE.businessentityid

WHERE SUBSTRING(HE.jobtitle,1,5) like 'Sales';



==========================================================================

78 From the following table write a query in SQL to return the last name of people so that it is in uppercase, trimmed, and concatenated with the first name.

select concat(UPPER(TRIM(LastName)),',',FirstName) as name

from Person.Person



==========================================================================

79 From the following table write a query in SQL to show a resulting expression that is too small to display. Return FirstName, LastName, Title, and SickLeaveHours. The SickLeaveHours will be shown as a small expression in text format.

SELECT PP.FirstName,

PP.LastName,

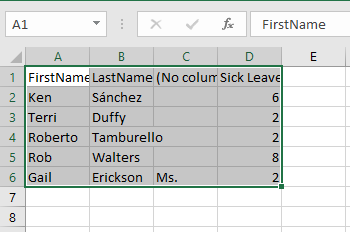
isnull(PP.Title,''),

SUBSTRING (CAST (HE.SickLeaveHours AS varchar(10)),1,1) AS [Sick Leave]

FROM HumanResources.Employee HE JOIN Person.Person PP

ON HE.BusinessEntityID = PP.BusinessEntityID

WHERE HE.BusinessEntityID <= 5;



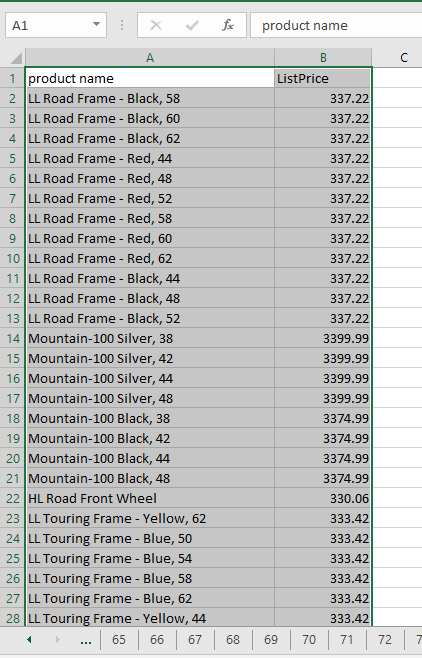
==========================================================================

80 From the following table write a query in SQL to retrieve the name of the products. Product, that have 33 as the first two digits of listprice.

SELECT Name as [product name], ListPrice

FROM production.Product

where SUBSTRING (CAST (ListPrice AS varchar (10)),1,2) ='33'



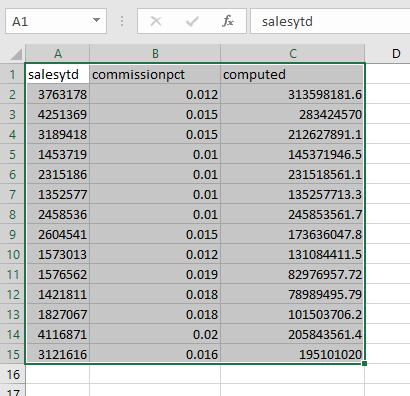
==========================================================================

81 From the following table write a query in SQL to calculate by dividing the total year-to-date sales (SalesYTD) by the commission percentage (CommissionPCT).Return SalesYTD, CommissionPCT, and the value rounded to the nearest whole number.

select salesytd,commissionpct,salesytd/commissionpct as [computed]

from Sales.SalesPerson

where commissionpct !=0



==========================================================================

82 From the following table write a query in SQL to find those persons that have a 2 in the first digit of their SalesYTD. Convert the SalesYTD column to an int type, and then to a char(20) type. Return FirstName, LastName, SalesYTD, and BusinessEntityID.

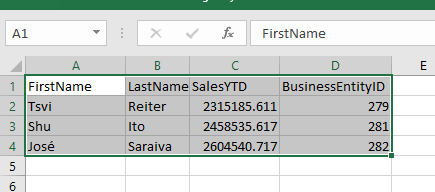
SELECT pp.FirstName, pp.LastName, ss.SalesYTD, ss.BusinessEntityID

FROM Person.Person AS pp

JOIN Sales.SalesPerson AS ss

ON pp.BusinessEntityID = ss.BusinessEntityID

WHERE CAST (CAST (ss.SalesYTD AS INT) AS char(20)) LIKE '2%';

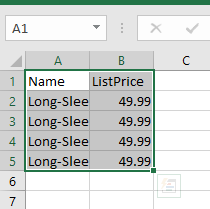


83 From the following table write a query in SQL to convert the Name column to a char(16) column. Convert those rows if the name starts with 'Long-Sleeve Logo Jersey'. Return name of the product and listprice.

SELECT CAST (Name AS CHAR (16)) AS Name, ListPrice

FROM production.Product

WHERE Name LIKE 'Long-Sleeve Logo Jersey%';



==========================================================================

84 From the following table write a SQL query to determine the discount price for the salesorderid 46672.Calculate only those orders with discounts of more than.02 percent. Return productid, UnitPrice, UnitPriceDiscount,

--and DiscountPrice (UnitPrice\*UnitPriceDiscount ).

SELECT productid,

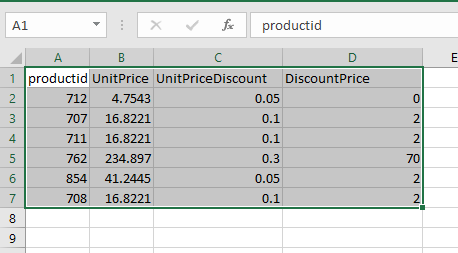
UnitPrice,

UnitPriceDiscount,

CAST (UnitPrice\*UnitPriceDiscount AS int) AS DiscountPrice

FROM sales.salesorderdetail

WHERE SalesOrderid = 46672 AND UnitPriceDiscount > .02;



=====================================================================================

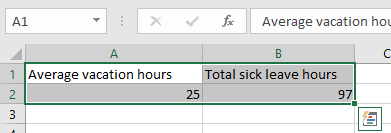
85 From the following table write a query in SQL to calculate the average vacation hours, and the sum of sick leave hours, that the vice presidents have used.

SELECT cast(AVG(VacationHours) as float)AS "Average vacation hours",

SUM(SickLeaveHours) AS "Total sick leave hours"

FROM HumanResources.Employee

WHERE JobTitle LIKE 'Vice President%';



==========================================================================

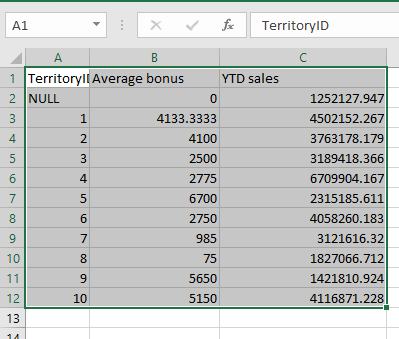
86 From the following table write a query in SQL to calculate the average bonus received and the sum of year-to-date sales for each territory. Return territoryid, Average bonus, and YTD sales.

SELECT TerritoryID, AVG(Bonus)as [Average bonus],

SUM(SalesYTD) as [YTD sales]

FROM Sales.SalesPerson

GROUP BY TerritoryID;



==========================================================================

88 From the following table write a query in SQL to return a moving average of yearly sales for each territory. Return BusinessEntityID, TerritoryID, SalesYear, SalesYTD, average SalesYTD as MovingAvg, and total SalesYTD as CumulativeTotal.

SELECT BusinessEntityID, TerritoryID

,year(ModifiedDate) AS SalesYear

,SalesYTD AS SalesYTD

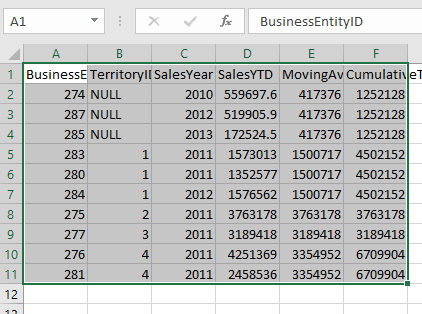
,AVG(SalesYTD) OVER (PARTITION BY TerritoryID ) AS MovingAvg

,SUM(SalesYTD) OVER (PARTITION BY TerritoryID ) AS CumulativeTotal

FROM Sales.SalesPerson

WHERE TerritoryID IS NULL OR TerritoryID < 5

ORDER BY TerritoryID,SalesYear;



==========================================================================

89 From the following table write a query in SQL to return a moving average of sales, by year, for all sales territories.

Return BusinessEntityID, TerritoryID, SalesYear, SalesYTD, average SalesYTD as MovingAvg, and total SalesYTD as CumulativeTotal.

SELECT BusinessEntityID, TerritoryID

Year (ModifiedDate) AS SalesYear ,

SalesYTD AS SalesYTD,

AVG(SalesYTD) OVER win AS MovingAvg ,

SUM (SalesYTD) OVER win AS CumulativeTotal

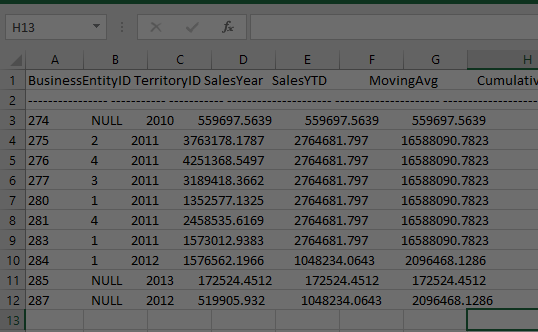
FROM Sales.SalesPerson

WHERE TerritoryID IS NULL OR TerritoryID < 5

WINDOW win AS (PARTITION BY year (ModifiedDate))

ORDER BY BusinessEntityID

SELECT \* FROM Sales.SalesPerson

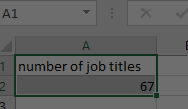


==========================================================================

90 From the following table write a query in SQL to return the number of different titles that employees can hold.

select count (distinct JobTitle) as [number of job titles]

from HumanResources.Employee

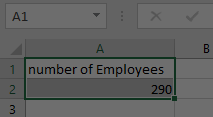
****

==========================================================================

91 From the following table write a query in SQL to find the total number of employees.

select count (\*) as [number of Employees]

from HumanResources.Employee

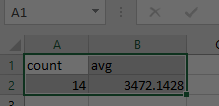
****

==========================================================================

91 From the following table write a query in SQL to find the total number of employees.

select count (\*) as [number of Employees]

from HumanResources.Employee

****

==========================================================================

93 From the following tables wirte a query in SQL to return aggregated values for each department.Return name, minimum salary, maximum salary, average salary, and number of employees in each department.

SELECT DISTINCT Name

, MIN(Rate) OVER win AS MinSalary

, MAX(Rate) OVER win AS MaxSalary

, AVG(Rate) OVER win AS AvgSalary

,COUNT(HED.BusinessEntityID) OVER win AS EmployeesPerDept

FROM HumanResources.EmployeePayHistory AS HE

JOIN HumanResources.EmployeeDepartmentHistory AS HED

ON HE.BusinessEntityID = HED.BusinessEntityID

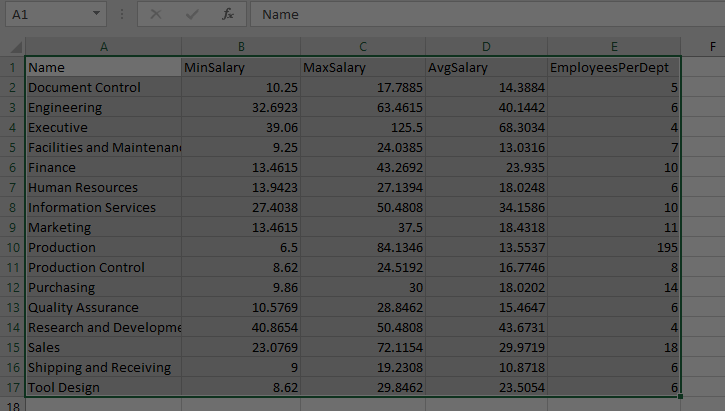
JOIN HumanResources.Department AS HD

ON HD.DepartmentID = HED.DepartmentID

WHERE HED.EndDate IS NULL

WINDOW win AS (PARTITION BY HED.DepartmentID)

ORDER BY Name;

****

==========================================================================

94 From the following tables write a SQL query to return the departments of a company that each have more than 15 employees.

select \* from humanresources.employee

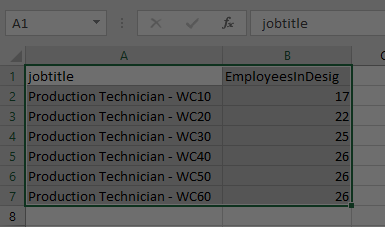
SELECT jobtitle,

COUNT(businessentityid) AS EmployeesInDesig

FROM humanresources.employee

GROUP BY jobtitle

HAVING COUNT(businessentityid) > 15;

****

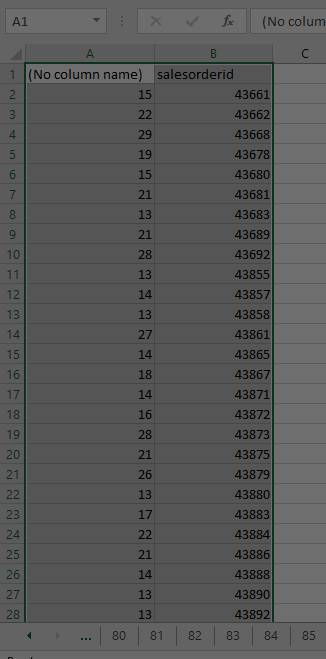
=========================================================================

95 From the following table write a query in SQL to find the number of products that ordered in each of the specified sales orders.

select COUNT(\*),salesorderid from Sales.SalesOrderDetail

group by SalesOrderID

having COUNT(\*)>12

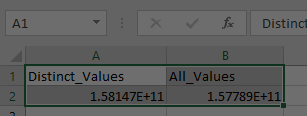
****

==========================================================================

97 SELECT var\_pop(DISTINCT SalesQuota) AS Distinct\_Values, var\_pop(SalesQuota) AS All\_Values FROM sales.salespersonquotahistory;

SELECT varp(DISTINCT SalesQuota) AS Distinct\_Values, varp(SalesQuota) AS All\_Values

FROM sales.salespersonquotahistory;

****

==========================================================================

--98 From the following table write a query in SQL to return the total ListPrice and StandardCost of products for each color.

--Products that name starts with 'Mountain' and ListPrice is more than zero. Return Color, total list price, total standardcode.

--Sort the result set on color in ascending order.

SELECT Color, SUM(ListPrice) as sum, SUM(StandardCost) as sum

FROM Production.Product

WHERE Color IS NOT NULL

AND ListPrice >0

AND Name LIKE 'Mountain%'

GROUP BY Color

ORDER BY Color;



==========================================================================

99 From the following table write a query in SQL to find the TotalSalesYTD of each SalesQuota.Show the summary of the TotalSalesYTD amounts for all SalesQuota groups. Return SalesQuota and TotalSalesYTD.

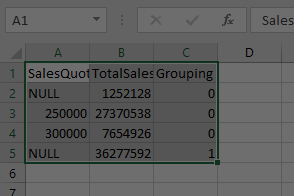
SELECT SalesQuota, SUM(SalesYTD) as "TotalSalesYTD" ,

GROUPING(SalesQuota) as "Grouping"

FROM Sales.SalesPerson

GROUP BY rollup(SalesQuota)

order by GROUPING(SalesQuota)

****

==========================================================================

100 From the following table write a query in SQL to calculate the sum of the ListPrice and StandardCost for each color.Return color, sum of ListPrice.

SELECT ISNULL (Color, '') AS Color,

SUM(ListPrice) AS TotalList,

SUM(StandardCost) AS TotalCost

FROM production.product

GROUP BY Color

ORDER BY CASE Color

WHEN not null THEN Color END asc



==========================================================================

101. From the following table write a query in SQL to calculate the salary percentile for each employee within a given department. Return department, last name, rate, cumulative distribution and percent rank of rate. Order the result set by ascending on department and descending on rate.

select HD.Name,

LastName,

Rate,

CUME\_DIST () OVER (PARTITION BY HD.DepartmentID ORDER BY Rate) AS CumeDist,

PERCENT\_RANK () OVER (PARTITION BY HD.DepartmentID ORDER BY Rate ) AS PctRank

from HumanResources.EmployeeDepartmentHistory HED

join HumanResources.Department HD

on HD.DepartmentID=HED.DepartmentID

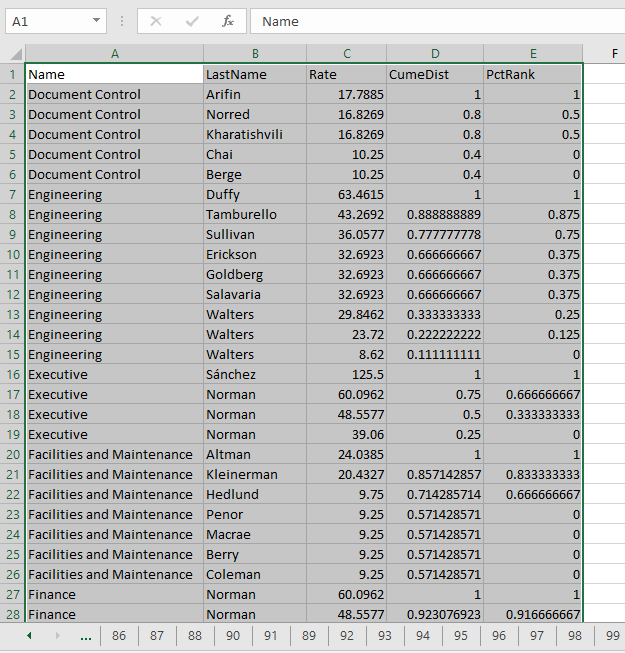
join Person.Person PP

on PP.BusinessEntityID=HED.BusinessEntityID

join HumanResources.EmployeePayHistory HEP

on PP.BusinessEntityID=HEP.BusinessEntityID

ORDER BY HD.Name, Rate DESC;



==========================================================================

102. From the following table write a query in SQL to return the name of the product that is the least expensive in a given product category. Return name, list price and the first value i.e. LeastExpensive of the product.

create proc listPricedProduct

@ProductSubcategoryID int

as

begin

SELECT Name,

ListPrice,

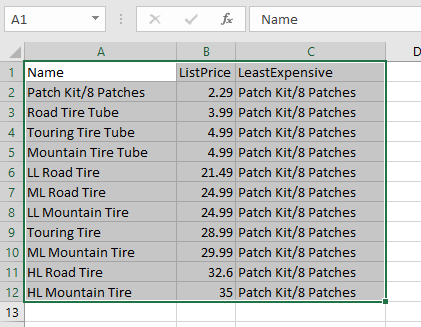
FIRST\_VALUE(Name) OVER (ORDER BY ListPrice ASC) AS LeastExpensive

FROM Production.Product

WHERE ProductSubcategoryID = @ProductSubcategoryID;

end

execute listPricedProduct @ProductSubcategoryID=37



==========================================================================

103. From the following table write a query in SQL to return the employee with the fewest number of vacation hours compared to other employees with the same job title. Partitions the employees by job title and apply the first value to each partition independently.

create view leastVacationHour

as

SELECT HE.jobTitle,

PP.lastName,

HE.vacationhours,

FIRST\_VALUE(PP.lastName) OVER ( partition by jobTitle ORDER BY VacationHours ASC) AS fewestvacationhours

from HumanResources.Employee HE

join Person.Person PP

on PP.BusinessEntityID=HE.BusinessEntityID

select \* from leastVacationHour



==========================================================================

104. From the following table write a query in SQL to return the difference in sales quotas for a specific employee overprevious years. Returun BusinessEntityID, sales year, current quota, and previous quota.

create proc diffQuotas

@BusinessEntityID int,

@year1 int,

@year2 int

as

begin

SELECT BusinessEntityID,

year(QuotaDate) AS SalesYear,

SalesQuota AS CurrentQuota,

LAG(SalesQuota, 1,0) OVER (ORDER BY year(QuotaDate)) AS PreviousQuota

FROM Sales.SalesPersonQuotaHistory

WHERE BusinessEntityID = @BusinessEntityID AND year(QuotaDate) in (@year1,@year2)

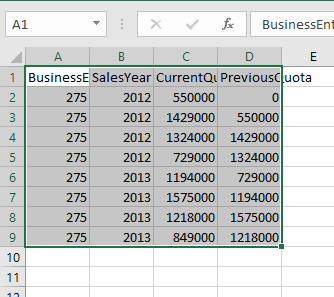
end

exec diffQuotas

@BusinessEntityID =275,

@year1 =2012,

@year2 =2013



==========================================================================

105. From the following table write a query in SQL to compare year-to-date sales between employees. Return TerritoryName, BusinessEntityID, SalesYTD, and sales of previous year i.e.PrevRepSales. Sort the result set in ascending order on territory name.

ALTER proc yearToDateDiff

as

begin

select territoryname,

businessentityid,

salesytd,

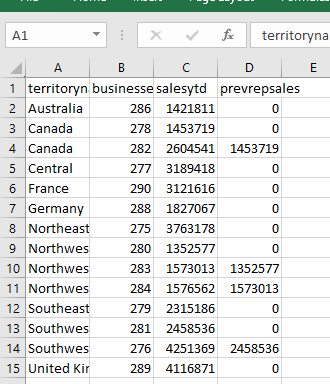
LAG(salesytd, 1,0) OVER (partition by TerritoryName ORDER BY salesytd) AS prevrepsales

from Sales.vSalesPerson

WHERE territoryname IS NOT NULL

end

EXEC yearToDateDiff



--==========================================================================

--106. From the following tables write a query in SQL to return the hire date of the last employee in each

--department for the given salary (Rate). Return department, lastname, rate, hiredate, and the last value of hiredate.

create proc lastEmployeHireDate

@department varchar(30),

@department2 varchar(30)

as

begin

select HVED.department,

HVED.lastname ,

HEP.rate ,

hiredate,

LAST\_VALUE(HireDate) OVER ( PARTITION BY Department ORDER BY Rate) AS lastvalue

from HumanResources.vEmployeeDepartmentHistory HVED

join HumanResources.EmployeePayHistory HEP

on HEP.BusinessEntityID=HVED.BusinessEntityID

join HumanResources.Employee HE

on HE.BusinessEntityID=HVED.BusinessEntityID

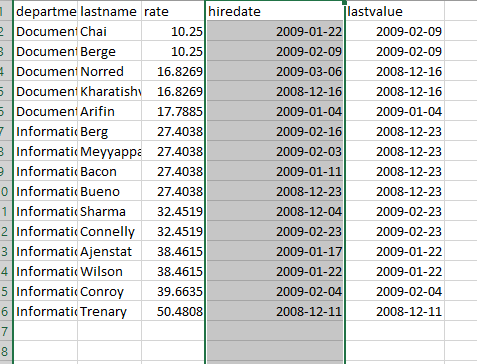
where HVED.department in (@department, @department2)

end

exec lastEmployeHireDate

@department ='Document Control',

@department2 ='Information Services'



==========================================================================

107. From the following table write a query in SQL to compute the difference between the sales quota value for the current quarter and the first and last quarter of the year respectively for a given number of employees. Return BusinessEntityID, quarter, year, differences between current quarter and first and last quarter. Sort the result set on BusinessEntityID, SalesYear, and Quarter in ascending order.

create proc quarterDifference

as

begin

SELECT BusinessEntityID

, DATEPART(QUARTER, QuotaDate) AS Quarter

,QuotaDate

, year(QuotaDate) AS SalesYear

, SalesQuota AS QuotaThisQuarter

, SalesQuota - FIRST\_VALUE(SalesQuota)

OVER (PARTITION BY BusinessEntityID, year(QuotaDate)

ORDER BY DATEPART(QUARTER, QuotaDate)

) AS DifferenceFromFirstQuarter

, SalesQuota - LAST\_VALUE(SalesQuota)

OVER (PARTITION BY BusinessEntityID, year(QuotaDate)

ORDER BY DATEPART(QUARTER, QuotaDate)

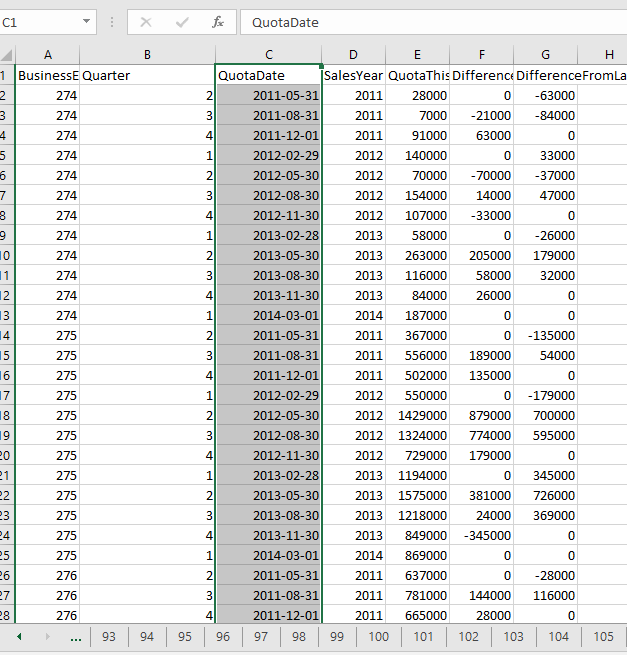
RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS DifferenceFromLastQuarter

FROM Sales.SalesPersonQuotaHistory

ORDER BY BusinessEntityID, SalesYear, Quarter;

end

exec quarterDifference



--==========================================================================

--108. From the following table write a query in SQL to return the statistical variance of the sales quota values for a salesperson

--for each quarter in a calendar year. Return quotadate, quarter, SalesQuota, and statistical variance.

--Order the result set in ascending order on quarter.

SELECT quotadate AS Year,

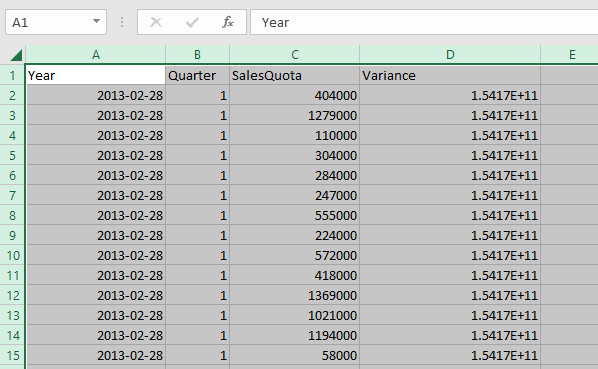
DATEPART (QUARTER, QuotaDate) AS Quarter,

SalesQuota AS SalesQuota,

varp(SalesQuota) OVER (ORDER BY year(quotadate), DATEPART(QUARTER, QuotaDate)) AS Variance

FROM sales.salespersonquotahistory

ORDER BY DATEPART (QUARTER, QuotaDate);



==========================================================================

109. From the following table write a query in SQL to return the difference in sales quotas for a specific employee over subsequent year. Return BusinessEntityID, year, SalesQuota, and the salesquota coming in next row.

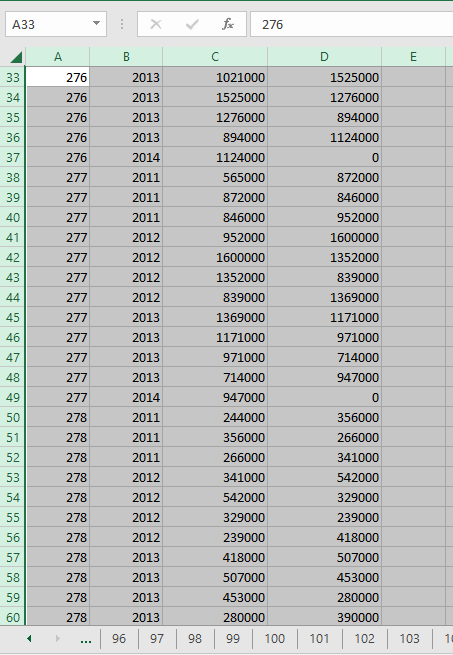
SELECT BusinessEntityID,

year(quotadate) AS SalesYear,

SalesQuota AS CurrentQuota,

LEAD( SalesQuota, 1,0) OVER ( partition by BusinessEntityID ORDER BY year(quotadate)) AS NextQuota

FROM Sales.SalesPersonQuotaHistory



--==========================================================================

--110. From the following query write a query in SQL to compare year-to-date sales between employees for specific terrotery. Return TerritoryName, BusinessEntityID, SalesYTD, and the salesquota coming in next row.

create proc compareWithTerritory

@TerritoryName1 varchar (20),

@TerritoryName2 varchar (20)

as

begin

SELECT TerritoryName,

BusinessEntityID,

SalesYTD,

LEAD (SalesYTD, 1, 0) OVER (PARTITION BY TerritoryName ORDER BY SalesYTD DESC) AS NextRepSales

FROM Sales.vSalesPerson

WHERE TerritoryName IN (@TerritoryName1, @TerritoryName2)

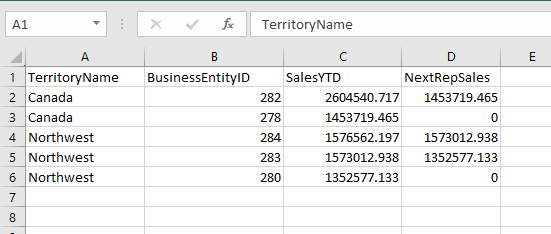
ORDER BY TerritoryName;

end;

exec compareWithTerritory

@TerritoryName1= 'Canada',

@TerritoryName2= 'Northwest'



==========================================================================

111. From the following table write a query in SQL to obtain the difference in sales quota values for a specified employee over subsequent calendar quarter. Return year, quarter, sales quota, next sales quota, and the difference in sales quota.Sort the result set on year and then by quarter, both in ascending order.

alter proc calendarDifference

as

begin

SELECT year(QuotaDate) AS Year,

year(QuotaDate) AS Quarter,

SalesQuota AS SalesQuota,

LEAD(SalesQuota,1,0) OVER (PARTITION BY BussinessEntityID ORDER BY year(QuotaDate), DATEPART(QUARTER, QuotaDate)) AS NextQuota,

SalesQuota - LEAD(SalesQuota,1,0) OVER (ORDER BY year(QuotaDate), DATEPART(QUARTER, QuotaDate)) AS Diff

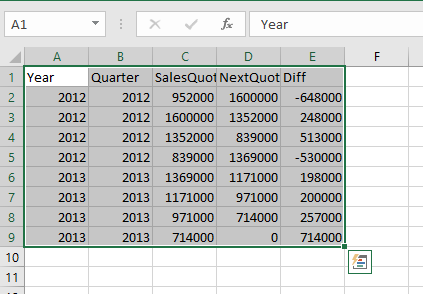
FROM sales.salespersonquotahistory

WHERE businessentityid = 277 AND year(QuotaDate) IN (2012,2013)

ORDER BY year(QuotaDate), DATEPART(QUARTER, QuotaDate);

end

exec calendarDifference



112. From the following table write a query in SQL to compute the salary percentile for each employee within a given department. Return Department, LastName, Rate, CumeDist, and percentile rank. Sort the result set in ascending order on department and descending order on rate.

--N.B.: The cumulative distribution calculates the relative position of a specified value in a group of values.

CREATE proc pr\_percentileCumdist

@Department1 varchar(30),

@Department2 varchar(30)

AS

begin

SELECT Department, LastName, Rate,

CUME\_DIST () OVER (PARTITION BY Department ORDER BY Rate) AS CumeDist,

PERCENT\_RANK() OVER (PARTITION BY Department ORDER BY Rate ) AS PctRank

FROM HumanResources.vEmployeeDepartmentHistory AS HVED

JOIN HumanResources.EmployeePayHistory AS HEP

ON HEP.BusinessEntityID = HVED.BusinessEntityID

WHERE Department IN (@Department1, @Department2)

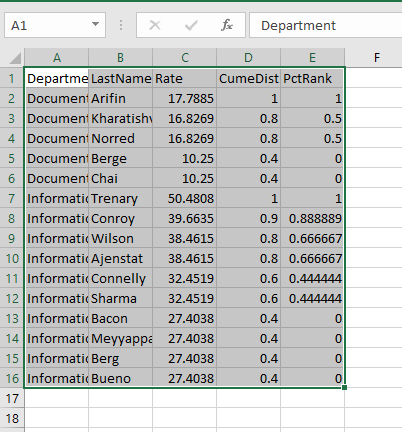
ORDER BY Department, Rate DESC

end

exec pr\_percentileCumdist

@Department1 ='Information Services',

@Department2 ='Document Control'



==========================================================================

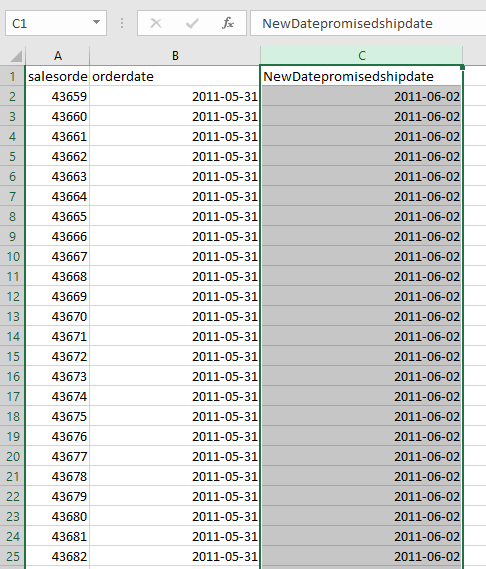
113. From the following table write a query in SQL to add two days to each value in the OrderDate column,to derive a new column named PromisedShipDate. Return salesorderid, orderdate, and promisedshipdate column.

select salesorderid,

orderdate ,

DATEADD (DAY, 2, orderdate) AS NewDatepromisedshipdate

from sales.salesorderheader



==========================================================================

114. From the following table write a query in SQL to obtain a newdate by adding two days with current date for each salespersons.Filter the result set for those salespersons whose sales value is more than zero.

select firstname,lastname ,DATEADD(DAY, 2, GETDATE()) AS [New Date]

from Sales.SalesPerson SS

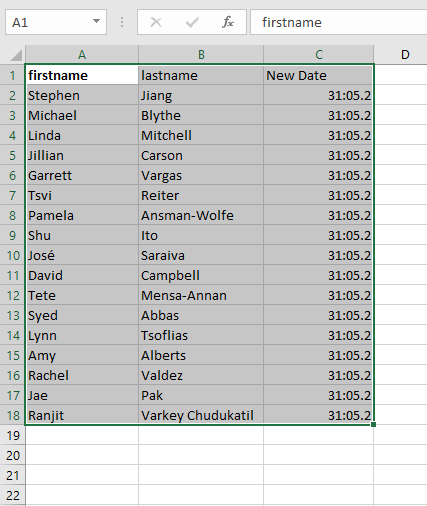
join Person.Person PP

on SS.BusinessEntityID = PP.BusinessEntityID

JOIN Person.Address AS A

ON A.AddressID = PP.BusinessEntityID

WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0;

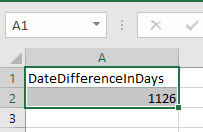


115. From the following table write a query in SQL to find the differences between the maximum and minimum orderdate.

select \* from Sales.SalesOrderHeader

SELECT DATEDIFF (DAY, MIN(OrderDate), MAX(OrderDate)) AS DateDifferenceInDays

FROM Sales.SalesOrderHeader;



==========================================================================

116. From the following table write a query in SQL to rank the products in inventory, by the specified inventory locations,

--according to their quantities. Divide the result set by LocationID and sort the result set on Quantity in descending order.

select PP.productid,

PP.name,

locationid,

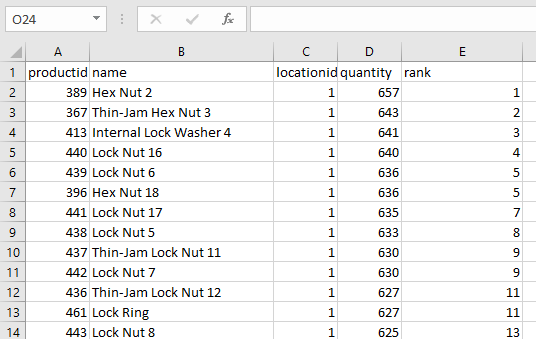
quantity,

DENSE\_RANK () OVER (partition by LocationID order by Quantity desc) as [rank]

from Production.ProductInventory PPI

join Production.Product PP

on PP.ProductID=PPI.productID



==========================================================================

117. From the following table write a query in SQL to return the top ten employees ranked by their salary.

select \* from HumanResources.EmployeePayHistory

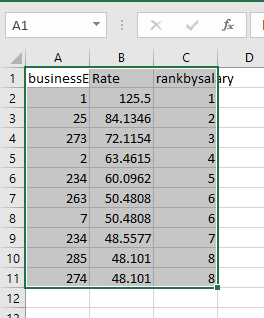
select top 10 businessEntityID,

Rate,

DENSE\_RANK () OVER (order by rate desc) as rankbysalary

from HumanResources.EmployeePayHistory

order by rate desc



==========================================================================

118. From the following table write a query in SQL to divide rows into four groups of employees based on their year-to-date sales. Return first name, last name, group as quartile, year-to-date sales, and postal code.

select firstname,

lastname,

NTILE(4) OVER ( order by salesYTD ) AS quartile,

salesytd,

postalcode

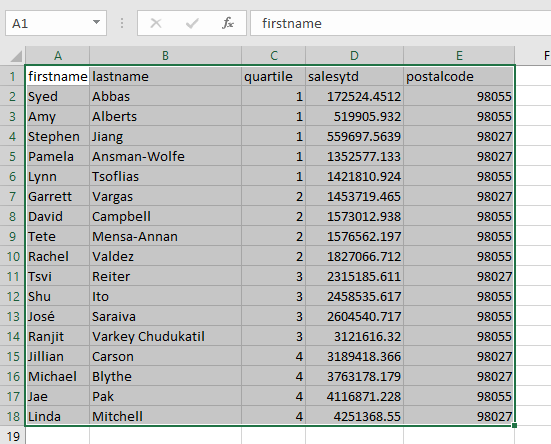
from Sales.SalesPerson SS

join Person.Person PP

on SS.BusinessEntityID = PP.BusinessEntityID

JOIN Person.Address AS A

ON A.AddressID = PP.BusinessEntityID



119. From the following tables write a query in SQL to rank the products in inventory the specified inventory locations according to their quantities. The result set is partitioned by LocationID and logically ordered by Quantity.

--Return productid, name, locationid, quantity, and rank.

select PP.productid,

PP.Name,

PPI.locationid,

PPI.quantity,

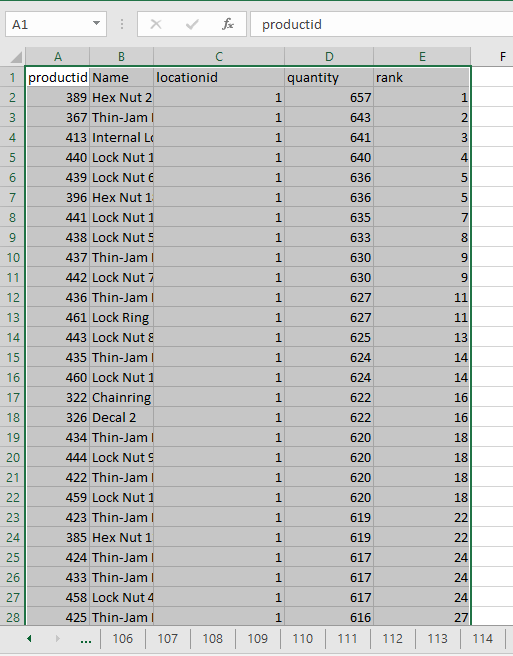
RANK() over(partition by locationID order by Quantity desc) as [rank]

from production.productinventory PPI

join Production.Product PP

on PP.ProductID=PPI.productID

order by locationID



120. From the following table write a query in SQL to find the salary of top ten employees. Return BusinessEntityID, Rate, and rank of employees by salary.

select \* from HumanResources.EmployeePayHistory

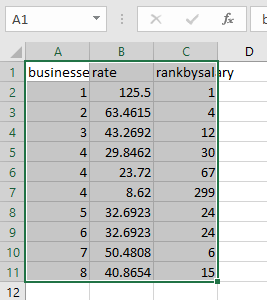
select top 10 businessentityid,

rate,

RANK() over( order by rate desc) as rankbysalary

from HumanResources.EmployeePayHistory

ORDER BY BusinessEntityID



==========================================================================

121. From the following table write a query in SQL to calculate a row number for the salespeople based on their year-to-date sales ranking. Return row number, first name, last name, and year-to-date sales.

SELECT ROW\_NUMBER () OVER (ORDER BY SalesYTD DESC) AS [Row],

FirstName,

LastName,

ROUND(SalesYTD,2) AS [Sales YTD]

FROM Sales.vSalesPerson

WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0;



122. From the following table write a query in SQL to calculate row numbers for all rows between 50 to 60 inclusive. Sort the result set on orderdate.

select salesorderid,orderdate , rownumber

from (select salesorderid,

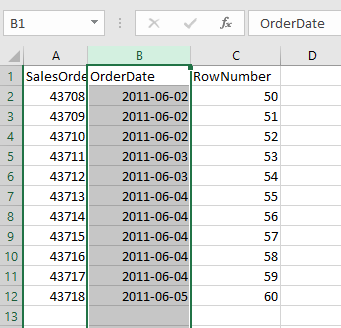
orderdate,

ROW\_NUMBER () OVER (ORDER BY orderDate) AS rownumber

from Sales.SalesOrderHeader

) derivedTable

WHERE rownumber BETWEEN 50 AND 60;



--==========================================================================

123. From the following table write a query in SQL to return first name, last name, territoryname, salesytd, and row number. Partition the query result set by the TerritoryName. Orders the rows in each partition by SalesYTD. Sort the result set on territoryname in ascending order.

SELECT FirstName,

LastName,

TerritoryName,

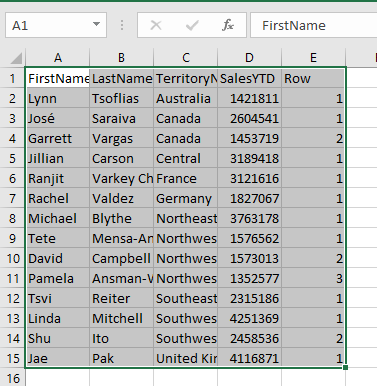
ROUND(SalesYTD,2) AS SalesYTD,

ROW\_NUMBER() OVER(PARTITION BY TerritoryName ORDER BY SalesYTD DESC) AS Row

FROM Sales.vSalesPerson

WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0

ORDER BY TerritoryName;



==========================================================================

124. From the following table write a query in SQL to order the result set by the column TerritoryName when the column CountryRegionName is equal to 'United States' and by CountryRegionName for all other rows. Return BusinessEntityID, LastName, TerritoryName, CountryRegionName.

select \* from Sales.vSalesPerson

SELECT BusinessEntityID, LastName, TerritoryName, CountryRegionName

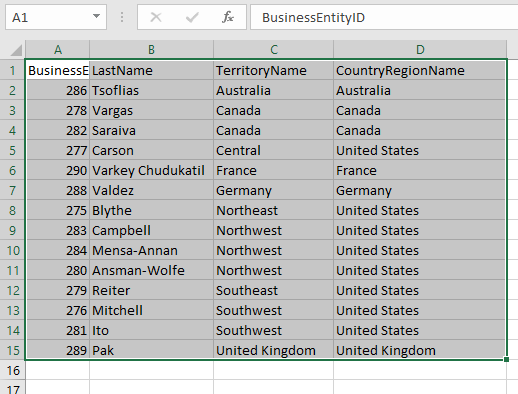
FROM Sales.vSalesPerson

WHERE TerritoryName IS NOT NULL

ORDER BY CASE CountryRegionName

WHEN 'United States' THEN TerritoryName

ELSE CountryRegionName END;



125. From the following tables write a query in SQL to return the highest hourly wage for each job title. Restricts the titles to those that are held by men with a maximum pay rate greater than 40 dollars or women with a maximum pay rate greater than 42 dollars.

SELECT jobtitle, MAX (HEP.Rate) AS maximumrate

FROM HumanResources.Employee AS HE

JOIN HumanResources.EmployeePayHistory AS HEP

ON HE.BusinessEntityID = HEP.BusinessEntityID

GROUP BY JobTitle

HAVING (

MAX(

CASE WHEN Gender = 'M' THEN HEP.Rate

ELSE NULL END

) > 40.00

OR

MAX(

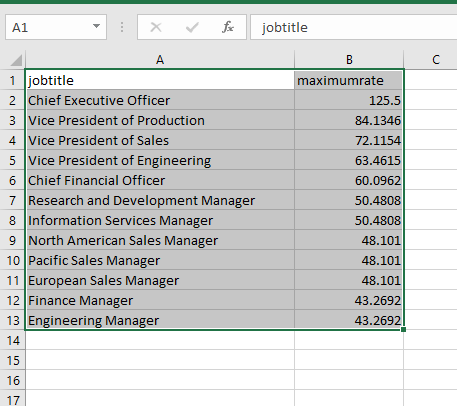
CASE WHEN Gender = 'F' HEP.Rate

ELSE NULL END

) > 42.00

)

ORDER BY MaximumRate DESC;



126. From the following table write a query in SQL to sort the BusinessEntityID in descending order for those employees that have the SalariedFlag set to 'true' and in ascending order that have the SalariedFlag set to 'false'.

--Return BusinessEntityID, and SalariedFlag.

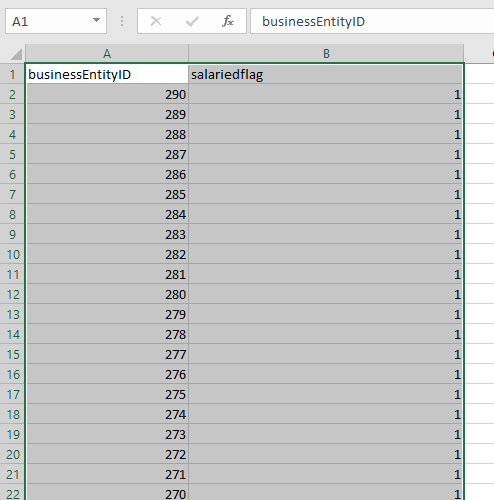
select businessEntityID,salariedflag from HumanResources.Employee

order by CASE when SalariedFlag = 'true'

THEN BusinessEntityID END DESC,

CASE WHEN SalariedFlag = 'false'

THEN BusinessEntityID END;



==========================================================================

127. From the following table write a query in SQL to display the list price as a text comment based on the price range for a product.Return ProductNumber, Name, and listprice. Sort the result set on ProductNumber in ascending order.

SELECT ProductNumber, Name, listprice,

CASE

WHEN ListPrice = 0 THEN 'Mfg item - not for resale'

WHEN ListPrice < 50 THEN 'Under $50'

WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'

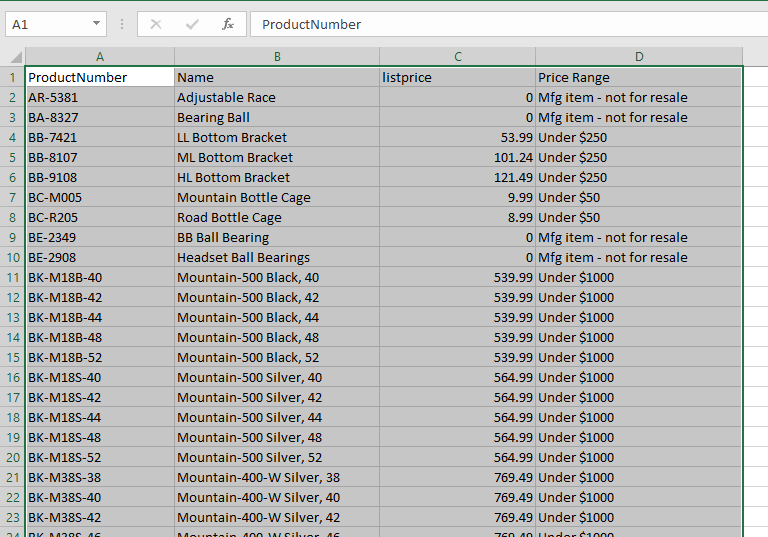
WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'

ELSE 'Over $1000'

END "Price Range"

FROM Production.Product

ORDER BY ProductNumber;



128. From the following table write a query in SQL to change the display of product line categories to make them more understandable. Return ProductNumber, category, and name of the product. Sort the result set in ascending order on ProductNumber.

SELECT ProductNumber,

CASE

WHEN ProductLine ='R' THEN 'Road'

WHEN ProductLine ='M' THEN 'Mountain'

WHEN ProductLine ='T' THEN 'Touring'

WHEN ProductLine ='S' THEN 'Other sale items'

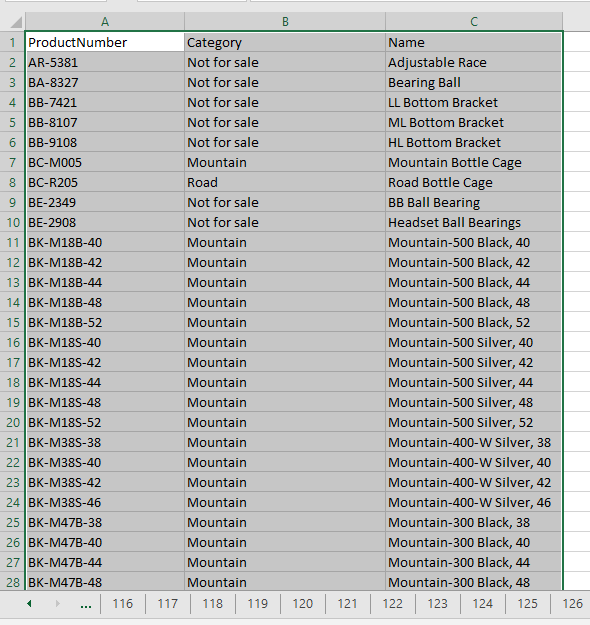
ELSE 'Not for sale'

end [Category],

Name

FROM Production.Product

ORDER BY ProductNumber;



129. From the following table write a query in SQL to evaluate whether the values in the MakeFlag and FinishedGoodsFlag columns are the same.

select productid,

makeflag,

finishedgoodsflag,

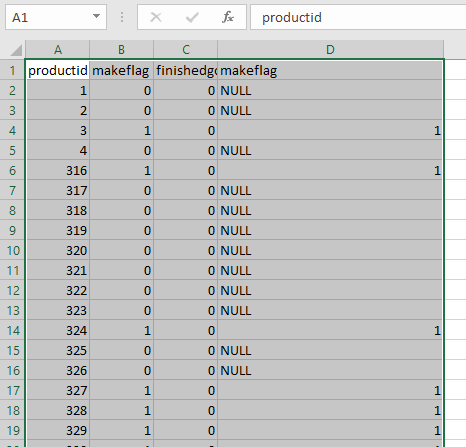
CASE

WHEN MakeFlag = FinishedGoodsFlag THEN isnull( CONVERT(VARCHAR(10), null) ,'')

ELSE MakeFlag

END as makeflag

from production.Product



130. From the following table write a query in SQL to select the data from the first column that has a nonnull value. Retrun Name, Class, Color, ProductNumber, and FirstNotNull.

SELECT Name,

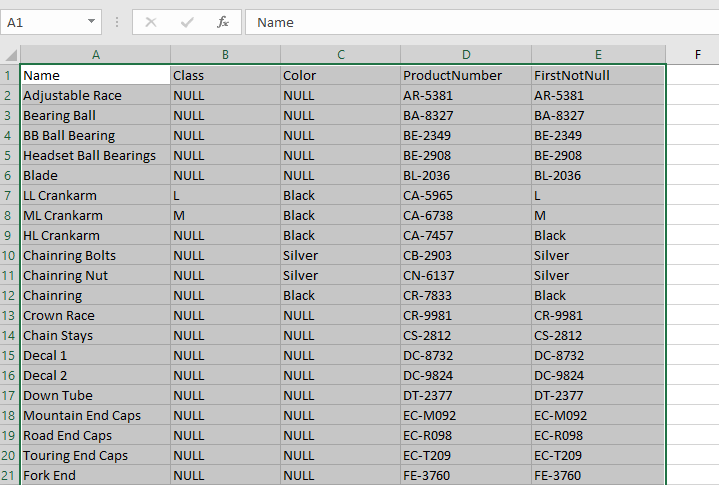
Class,

Color,

ProductNumber,

COALESCE (Class, Color, ProductNumber) AS FirstNotNull

FROM Production.Product;



131. From the following table write a query in SQL to check the values of MakeFlag and FinishedGoodsFlag columns and return whether they are same or not. Return ProductID, MakeFlag, FinishedGoodsFlag, and the column that are null or not null.

SELECT ProductID, MakeFlag, FinishedGoodsFlag,

CASE

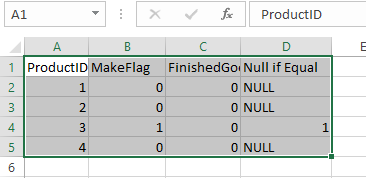
WHEN MakeFlag = FinishedGoodsFlag THEN NULL

ELSE MakeFlag

END AS "Null if Equal"

FROM Production.Product

WHERE ProductID < 10;

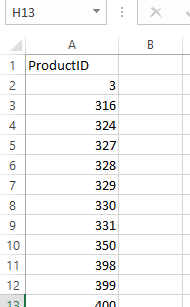


--132. From the following tables write a query in SQL to return any distinct values that are returned by both the query.

SELECT ProductID

FROM Production.Product

where ProductID in (SELECT ProductID FROM Production.WorkOrder )

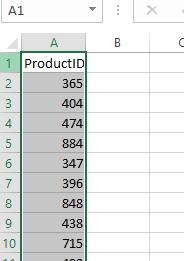


133. From the following tables write a query in SQL to return any distinct values from first query that aren't also found on the 2nd query.

SELECT ProductID

FROM Production.Product

where ProductID not in (SELECT ProductID FROM Production.WorkOrder)



134. From the following tables write a query in SQL to fetch any distinct values from the left query that aren't also present in the query to the right.

SELECT ProductID

FROM Production.WorkOrder

EXCEPT

SELECT ProductID

FROM Production.Product;

ProductID

No results is displayed

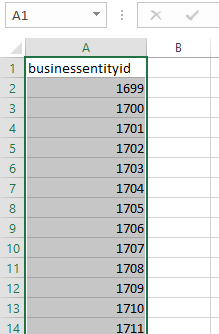
135. From the following tables write a query in SQL to fetch distinct businessentityid that are returned by both the specified query. Sort the result set by ascending order on businessentityid.

SELECT businessentityid

FROM person.businessentity

where businessentityid in (SELECT businessentityid FROM person.person where person.persontype = 'IN' )

ORDER BY businessentityid;



136. From the following table write a query which is the combination of two queries.

Return any distinct businessentityid from the 1st query that aren't also found in the 2nd query. Sort the result set in ascending order on businessentityid.

SELECT businessentityid

FROM person.businessentity

WHERE businessentityid NOT IN (

SELECT businessentityid

FROM person.person

WHERE person.persontype = 'IN'

)

ORDER BY businessentityid;



--137. From the following tables write a query in SQL to combine the ProductModelID and Name columns.A result set includes columns for productid 3 and 4. Sort the results by name ascending.

SELECT ProductID, Name

FROM Production.Product

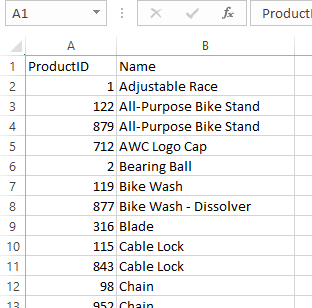
WHERE ProductID NOT IN (3, 4)

UNION ALL

SELECT ProductModelID, Name

FROM Production.ProductModel

ORDER BY Name;



138. From the following table write a query in SQL to find a total number of hours away from work can be calculated by adding vacation time and sick leave. Sort results ascending by Total Hours Away.

SELECT p.FirstName, p.LastName, VacationHours, SickLeaveHours,

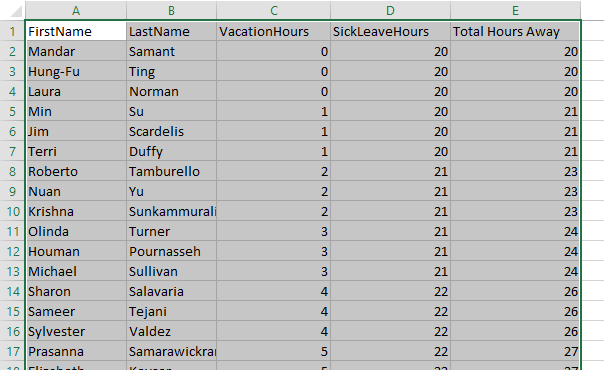
(CASE WHEN VacationHours IS NULL THEN 0 ELSE VacationHours END) +

(CASE WHEN SickLeaveHours IS NULL THEN 0 ELSE SickLeaveHours END) AS TotalHoursAway

FROM HumanResources.Employee AS e

JOIN Person.Person AS p ON e.BusinessEntityID = p.BusinessEntityID

ORDER BY TotalHoursAway ASC;



139. From the following table write a query in SQL to calculate the tax difference between the highest and lowest tax-rate state or province.

SELECT MAX(TaxRate) - MIN(TaxRate) AS "Tax Rate Difference"

FROM Sales.SalesTaxRate

WHERE StateProvinceID IS NOT NULL;

|  |
| --- |
| Tax Rate Difference |
| 14.60 |

140. From the following tables write a query in SQL to calculate sales targets per month for salespeople.

SELECT SS.BusinessEntityID AS SalesPersonID,

FirstName, LastName,

SalesQuota,

SalesQuota/12 AS [Sales Target Per Month]

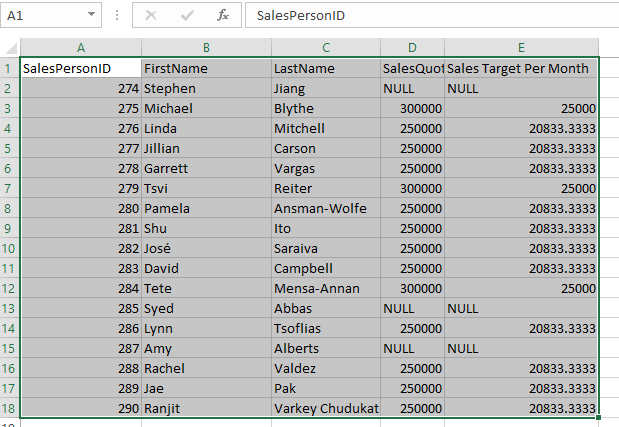
FROM Sales.SalesPerson AS SS

JOIN HumanResources.Employee AS HE

ON SS.BusinessEntityID = HE.BusinessEntityID

JOIN Person.Person AS p

ON HE.BusinessEntityID = p.BusinessEntityID;



140. From the following tables write a query in SQL to calculate sales targets per month for salespeople.

SELECT SS.BusinessEntityID AS SalesPersonID,

FirstName, LastName,

SalesQuota,

SalesQuota/12 AS [Sales Target Per Month]

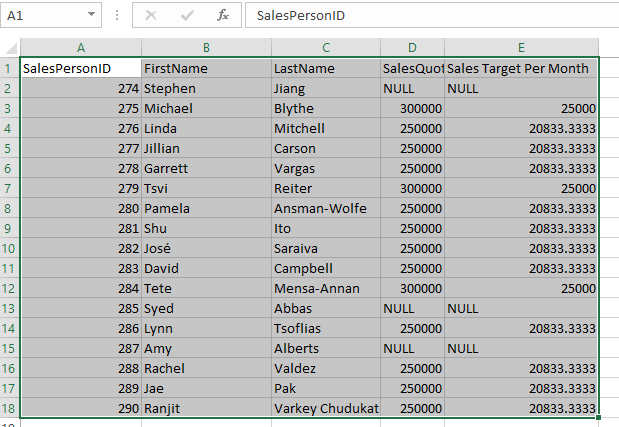
FROM Sales.SalesPerson AS SS

JOIN HumanResources.Employee AS HE

ON SS.BusinessEntityID = HE.BusinessEntityID

JOIN Person.Person AS p

ON HE.BusinessEntityID = p.BusinessEntityID;



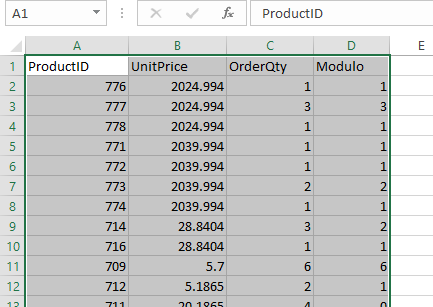
141.From the following table write a query in SQL to return the ID number, unit price,

and the modulus (remainder) of dividing product prices. Convert the modulo to an integer value.

SELECT ProductID, UnitPrice, OrderQty,

CAST (UnitPrice AS INT) % OrderQty AS Modulo

FROM Sales.SalesOrderDetail;



142. From the following table write a query in SQL to select employees who

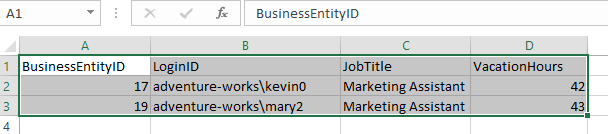
have the title of Marketing Assistant and more than 41 vacation hours.

SELECT BusinessEntityID, LoginID, JobTitle, VacationHours

FROM HumanResources.Employee

WHERE JobTitle = 'Marketing Assistant'

AND VacationHours > 41;



--143. From the following tables write a query in SQL to find all rows outside a specified range of rate between 27 and 30. Sort the result in ascending order on rate.

SELECT HVE.FirstName,

HVE.LastName,

HEP.Rate

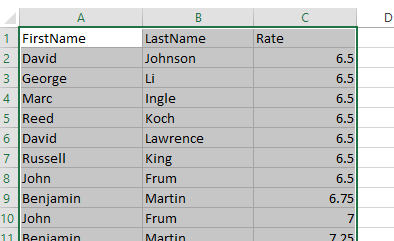
FROM HumanResources.vEmployee HVE

JOIN HumanResources.EmployeePayHistory HEP

ON HVE.BusinessEntityID = HEP.BusinessEntityID

WHERE HEP.Rate NOT BETWEEN 27 AND 30

ORDER BY HEP.Rate;

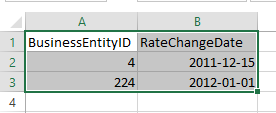


--144. From the follwing table write a query in SQL to retrieve rows whose datetime values are between '20111212' and '20120105'.

SELECT BusinessEntityID, RateChangeDate

FROM HumanResources.EmployeePayHistory

WHERE RateChangeDate BETWEEN '20111212' AND '20120105';

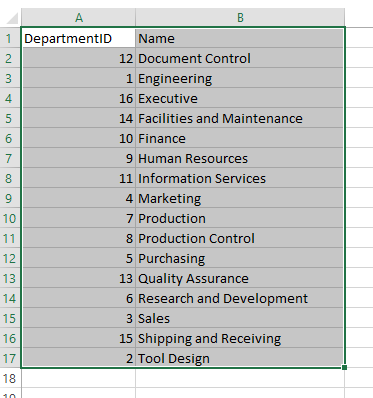


--145. From the following table write a query in SQL to return TRUE even if NULL is specified in the subquery. Return DepartmentID, Name and sort the result set in ascending order.

SELECT DepartmentID, Name

FROM HumanResources.Department

ORDER BY Name ASC;



--146. From the following tables write a query in SQL to get employees with Johnson last names. Return first name and last name.

SELECT PP.FirstName, PP.LastName

FROM Person.Person AS PP

JOIN HumanResources.Employee AS HE

ON PP.BusinessEntityID = HE.BusinessEntityID

WHERE PP.LastName = 'Johnson';

--147. From the following tables write a query in SQL to find stores whose name is the same name as a vendor.

SELECT DISTINCT SS.Name

FROM Sales.Store AS SS

join Purchasing.Vendor AS PV

on SS.Name = PV.Name

|  |
| --- |
| Name |

--148. From the following tables write a query in SQL to find employees of departments that start with P. Return first name, last name, job title.

SELECT pp.FirstName, pp.LastName, he.JobTitle

FROM Person.Person AS pp

JOIN HumanResources.Employee AS he

ON he.BusinessEntityID = pp.BusinessEntityID

JOIN HumanResources.EmployeeDepartmentHistory AS hed

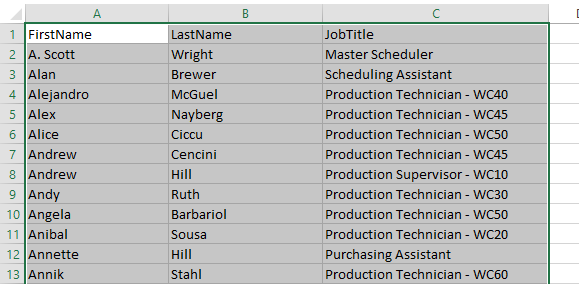
ON he.BusinessEntityID = hed.BusinessEntityID

JOIN HumanResources.Department AS hd

ON hd.DepartmentID = hed.DepartmentID

WHERE hd.Name LIKE 'P%'

order by pp.FirstName



149. From the following tables write a query in SQL to find all employees that do not belong to departments whose names begin with P.

SELECT pp.FirstName, pp.LastName, e.JobTitle

FROM Person.Person AS pp

JOIN HumanResources.Employee AS e

ON e.BusinessEntityID = pp.BusinessEntityID

LEFT JOIN (

SELECT edh.BusinessEntityID

FROM HumanResources.EmployeeDepartmentHistory AS edh

JOIN HumanResources.Department AS d

ON d.DepartmentID = edh.DepartmentID

WHERE d.Name LIKE 'P%'

) AS subquery

ON e.BusinessEntityID = subquery.BusinessEntityID

WHERE subquery.BusinessEntityID IS NULL

ORDER BY pp.LastName, pp.FirstName;



150. From the following table write a query in SQL to select employees who work as design engineers,tool designers, or marketing assistants.

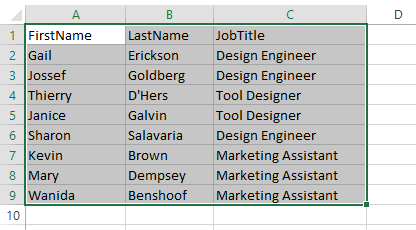
SELECT pp.FirstName, pp.LastName, he.JobTitle

FROM Person.Person pp

JOIN HumanResources.Employee he

ON pp.BusinessEntityID = he.BusinessEntityID

WHERE he.JobTitle IN ('Design Engineer', 'Tool Designer', 'Marketing Assistant');



--151. From the following tables write a query in SQL to identify all SalesPerson IDs for employees with sales quotas over $250,000.

--Return first name, last name of the sales persons.

SELECT p.FirstName, p.LastName

FROM Person.Person AS p

JOIN Sales.SalesPerson AS sp

ON p.BusinessEntityID = sp.BusinessEntityID

JOIN (

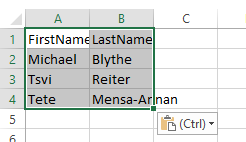
SELECT BusinessEntityID

FROM Sales.SalesPerson

WHERE SalesQuota > 250000

) AS subquery

ON p.BusinessEntityID = subquery.BusinessEntityID;



--152. From the following tables write a query in SQL to find the salespersons who do not have a quota greater than $250,000. Return first name and last name.

SELECT p.FirstName, p.LastName

FROM Person.Person AS p

JOIN Sales.SalesPerson AS sp

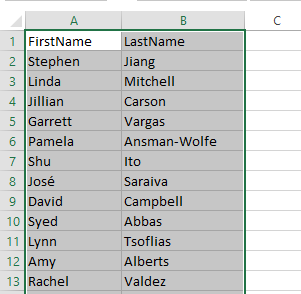
ON p.BusinessEntityID = sp.BusinessEntityID

WHERE p.BusinessEntityID NOT IN

(SELECT BusinessEntityID

FROM Sales.SalesPerson

WHERE SalesQuota > 250000);



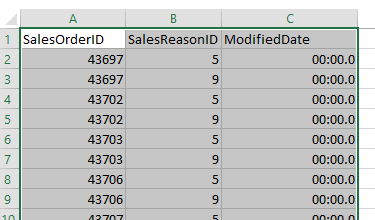
--153. From the following tables write a query in SQL to identify salesorderheadersalesreason and SalesReason tables with the same salesreasonid.

SELECT sohs.\*

FROM sales.salesorderheadersalesreason sohs

JOIN sales.SalesReason sr

ON sohs.salesreasonid = sr.salesreasonid;



--154. From the following table write a query in SQL to find all telephone numbers that have area code 415. Returns the first name, last name, and phonenumber. Sort the result set in ascending order by lastname.

SELECT pp.FirstName, pp.LastName, ppp.PhoneNumber

FROM Person.PersonPhone AS ppp

INNER JOIN Person.Person AS pp

ON ppp.BusinessEntityID = pp.BusinessEntityID

WHERE ppp.PhoneNumber LIKE '415%'

ORDER by pp.LastName;



--155. From the following tables write a query in SQL to identify all people with the first name 'Gail' with area codes other than 415. Return first name, last name, telephone number. Sort the result set in ascending order on lastname.

SELECT pp.FirstName, pp.LastName, ppp.PhoneNumber

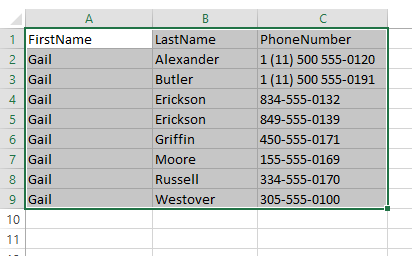
FROM Person.PersonPhone AS ppp

JOIN Person.Person AS pp

ON ppp.BusinessEntityID = pp.BusinessEntityID

WHERE ppp.PhoneNumber NOT LIKE '415%' AND pp.FirstName = 'Gail'

ORDER BY pp.LastName;

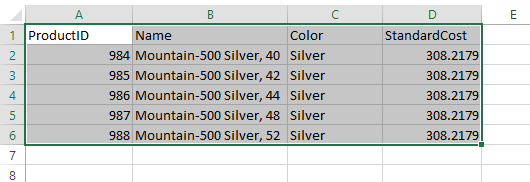


--156. From the following tables write a query in SQL to find all Silver colored bicycles with a standard price under $400. Return ProductID, Name, Color, StandardCost.

SELECT ProductID, Name, Color, StandardCost

FROM Production.Product

WHERE ProductNumber LIKE 'BK-%' AND StandardCost <= 400 AND Color = 'Silver'



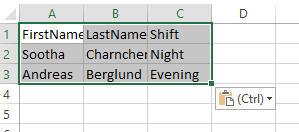
--157. From the following table write a query in SQL to retrieve the names of Quality Assurance personnel working the evening or night shifts. Return first name, last name, shift.

SELECT FirstName, LastName, Shift

FROM HumanResources.vEmployeeDepartmentHistory

WHERE Department = 'Quality Assurance'

AND Shift in ('Evening','Night');



--158. From the following table write a query in SQL to list all people with three-letter first names ending in 'an'.

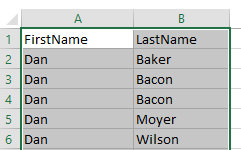
--Sort the result set in ascending order on first name. Return first name and last name.

SELECT FirstName, LastName

FROM Person.Person

WHERE FirstName LIKE '\_an'

ORDER BY FirstName;

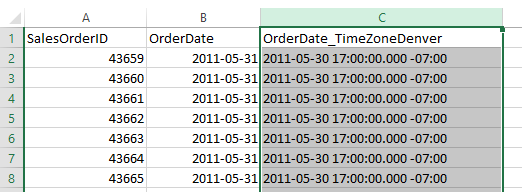


--159. From the following table write a query in SQL to convert the order date in the 'America/Denver' time zone. Return salesorderid, order date, and orderdate\_timezoneade.

SELECT SalesOrderID, OrderDate,

SWITCHOFFSET(OrderDate, '-07:00') AS OrderDate\_TimeZoneDenver

FROM Sales.SalesOrderHeader;



--160. From the following table write a query in SQL to convert order date in the 'America/Denver' time zone

--and also convert from 'America/Denver' time zone to 'America/Chicago' time zone

SELECT SalesOrderID, OrderDate,

TODATETIMEOFFSET(OrderDate, '-06:00') AS OrderDate\_Denver,

TODATETIMEOFFSET(OrderDate, '-05:00') AS OrderDate\_Chicago

FROM Sales.SalesOrderHeader;

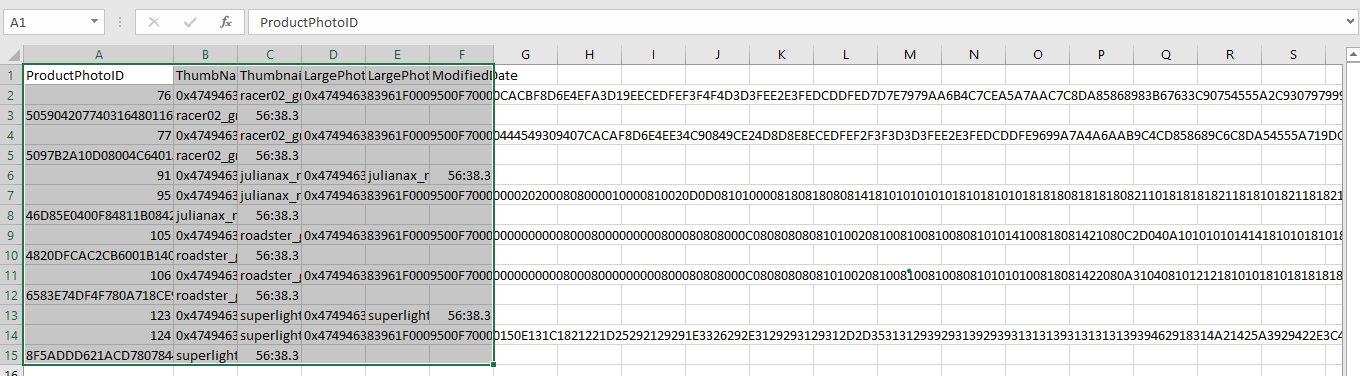
--161. From the following table wirte a query in SQL to search for rows with the 'green\_' character in the LargePhotoFileName column.

--Return all columns.

SELECT \*

FROM Production.ProductPhoto

WHERE LargePhotoFileName LIKE '%greena\_%' ESCAPE 'a';



--162. From the following tables write a query in SQL to obtain mailing addresses for companies in cities that begin with PA,outside the United States (US). Return AddressLine1, AddressLine2, City, PostalCode, CountryRegionCode.

SELECT isnull(AddressLine1,'') AddressLine1,

isnull(AddressLine2,'') AddressLine2,

City,

PostalCode,

CountryRegionCode

FROM Person.Address AS PA

JOIN Person.StateProvince AS SS ON PA.StateProvinceID = SS.StateProvinceID

WHERE CountryRegionCode NOT IN ('US')

AND City LIKE 'Pa%';



--163. From the following table write a query in SQL to specify that a JOIN clause can join multiple values. Return ProductID, product Name, and Color.

SELECT PP.ProductID, PP.Name, PP.Color

FROM Production.Product AS PP

JOIN (

SELECT 'Blade' AS Name

UNION ALL

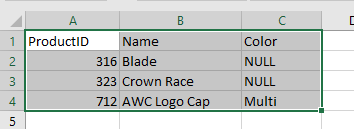
SELECT 'Crown Race'

UNION ALL

SELECT 'AWC Logo Cap'

) AS derived

ON PP.Name = derived.Name;



--164. From the following table write a query in SQL to find the SalesPersonID, salesyear, totalsales,salesquotayear, salesquota,

--and amt\_above\_or\_below\_quota columns. Sort the result set in ascending order on SalesPersonID, and SalesYear columns.

SELECT

Sales\_CTE.SalesPersonID,

Sales\_CTE.SalesYear,

CAST(Sales\_CTE.TotalSales AS VARCHAR(10)) AS TotalSales,

Sales\_Quota\_CTE.SalesQuotaYear,

CAST(Sales\_Quota\_CTE.SalesQuota AS VARCHAR(10)) AS SalesQuota,

CAST(Sales\_CTE.TotalSales - Sales\_Quota\_CTE.SalesQuota AS VARCHAR(10)) AS Amt\_Above\_or\_Below\_Quota

FROM

(

SELECT

SalesPersonID,

SUM(TotalDue) AS TotalSales,

year( OrderDate) AS SalesYear

FROM

Sales.SalesOrderHeader

WHERE

SalesPersonID IS NOT NULL

GROUP BY

SalesPersonID,

year(OrderDate)

) AS Sales\_CTE

JOIN

(

SELECT

BusinessEntityID,

SUM(SalesQuota) AS SalesQuota,

year( QuotaDate) AS SalesQuotaYear

FROM

Sales.SalesPersonQuotaHistory

GROUP BY

BusinessEntityID,

year( QuotaDate)

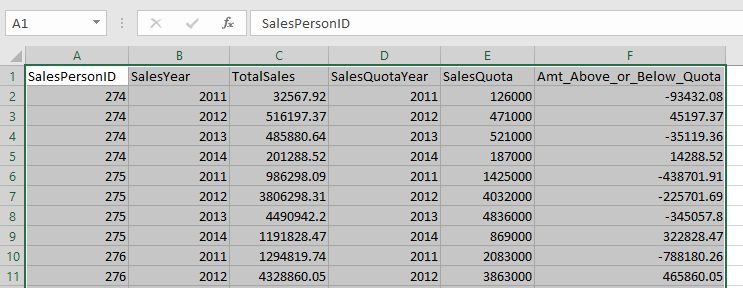
) AS Sales\_Quota\_CTE ON Sales\_Quota\_CTE.BusinessEntityID = Sales\_CTE.SalesPersonID

AND Sales\_CTE.SalesYear = Sales\_Quota\_CTE.SalesQuotaYear

ORDER BY

Sales\_CTE.SalesPersonID,

Sales\_CTE.SalesYear;



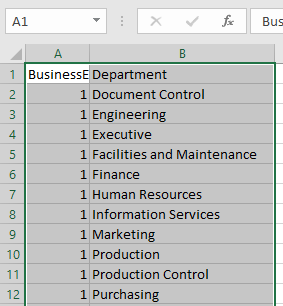
--165. From the following tables write a query in SQL to return the cross product of BusinessEntityID and Department columns.The following example returns the cross product of the two tables Employee and Department in the AdventureWorks2019 database. A list of all possible combinations of BusinessEntityID rows and all Department name rows are returned.

SELECT HRE.BusinessEntityID, HRD.Name AS Department

FROM HumanResources.Employee AS HRE

CROSS JOIN HumanResources.Department AS HRD

ORDER BY HRE.BusinessEntityID, HRD.Name;



--166. From the following tables write a query in SQL to return the SalesOrderNumber, ProductKey, and EnglishProductName columns.

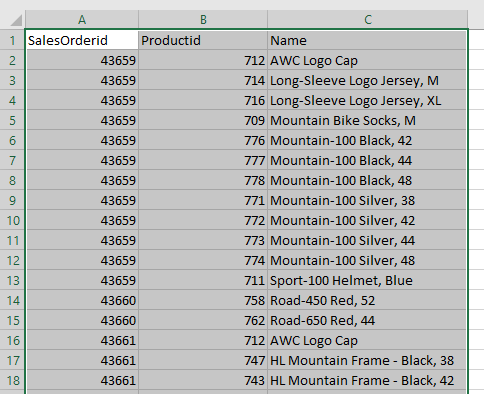
SELECT SS.SalesOrderid, PP.Productid, PP.Name

FROM sales.salesorderdetail AS SS

JOIN production.product AS PP

ON PP.Productid = SS.Productid

order by SS.SalesOrderid,PP.Name



--167. From the following tables write a query in SQL to return all orders with IDs greater than 60000.

SELECT SS.SalesOrderid, PP.Productid, PP.Name

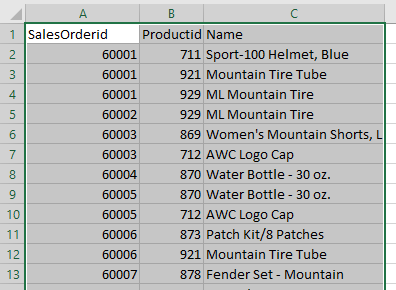
FROM sales.salesorderdetail AS SS

JOIN production.product AS PP

ON PP.Productid = SS.Productid

WHERE SS.SalesOrderid > 60000

order by SS.SalesOrderid



--168. From the following tables write a query in SQL to retrieve the SalesOrderid.

--A NULL is returned if no orders exist for a particular Territoryid. Return territoryid, countryregioncode, and salesorderid. Results are sorted by SalesOrderid, so that NULLs appear at the top.

SELECT

SST.Territoryid,

SST.countryregioncode,

SSO.SalesOrderid

FROM

sales.salesorderheader AS SSO

RIGHT OUTER JOIN

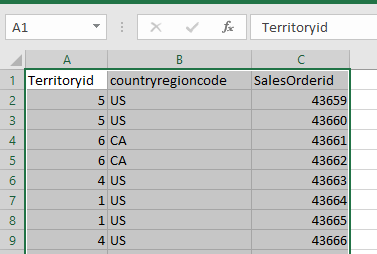
sales.salesterritory AS SST

ON

SSO.Territoryid = SST.Territoryid

ORDER BY

SSO.SalesOrderid;



--169. From the following table write a query in SQL to return all rows from both joined tables but returns NULL for values that do not match from the other table. Return territoryid, countryregioncode, and salesorderid. Results are sorted by SalesOrderid.

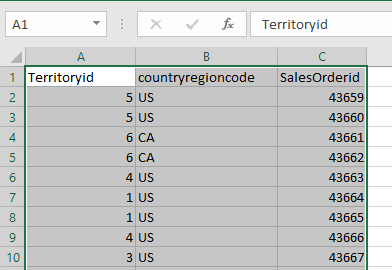
SELECT SST.Territoryid, SST.countryregioncode, SSO.SalesOrderid

FROM sales.salesterritory AS SST

FULL JOIN sales.salesorderheader AS SSO

ON SST.Territoryid = SSO.Territoryid

ORDER BY SSO.SalesOrderid;



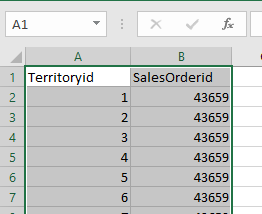
--170. From the following tables write a query in SQL to return a cross-product. Order the result set by SalesOrderid.

SELECT SST.Territoryid, SSO.SalesOrderid

FROM sales.salesterritory AS SST

CROSS JOIN sales.salesorderheader AS SSO

ORDER BY SSO.SalesOrderid;



--171. From the following table write a query in SQL to return all customers with BirthDate values after January 1, 1970 and the last name 'Smith'. Return businessentityid, jobtitle, and birthdate. Sort the result set in ascending order on birthday.

SELECT HE.businessentityid, jobtitle, birthdate

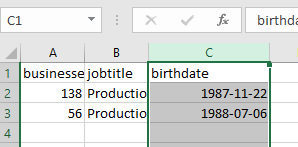
FROM humanresources.employee HE

join Person.Person PP

on PP.BusinessEntityID=HE.BusinessEntityID

WHERE BirthDate > '1970-01-01' and LastName like 'Smith'

ORDER BY birthdate;



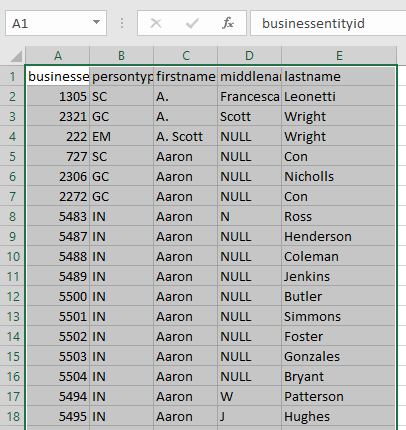
--172. From the following table write a query in SQL to return the rows with different firstname values from Adam. Return businessentityid, persontype, firstname, middlename,and lastname. Sort the result set in ascending order on firstname.

SELECT businessentityid, persontype, firstname, middlename, lastname

FROM person.person

WHERE firstname <> 'Adam'

ORDER BY firstname;



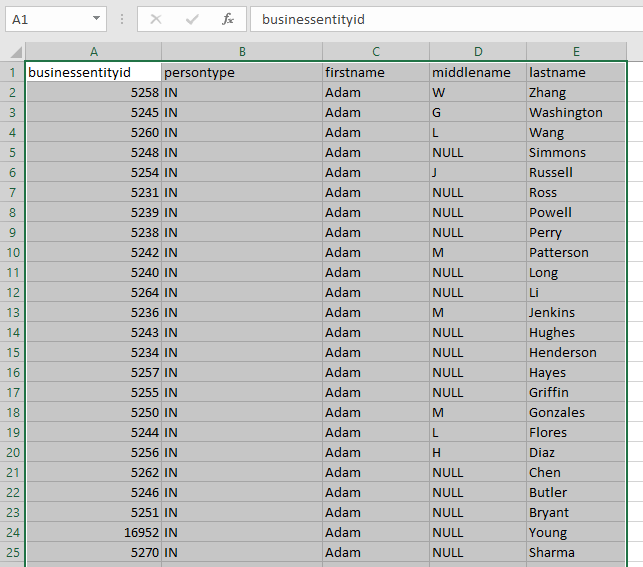
--173. From the following table write a query in SQL to find the rows where firstname doesn't differ from Adam's firstname. Return businessentityid, persontype, firstname, middlename,and lastname. Sort the result set in ascending order on firstname.

SELECT businessentityid, persontype, firstname, middlename, lastname

FROM person.person

WHERE firstname = 'Adam'

ORDER BY firstname ASC;



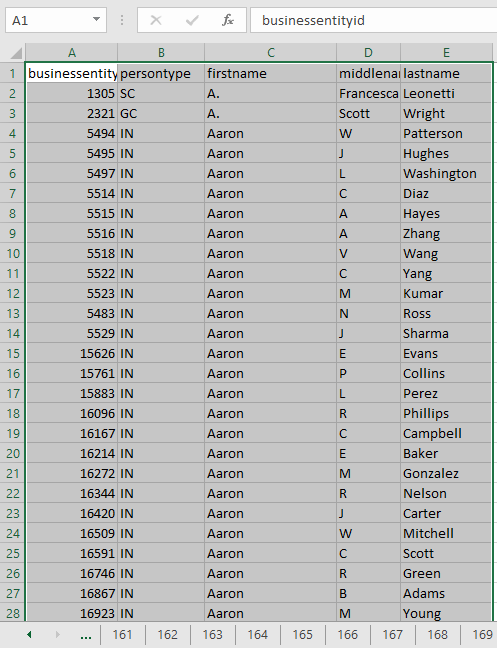
--174. From the following table write a query in SQL to find the rows where middlename differs from NULL. Return businessentityid, persontype, firstname, middlename,and lastname. Sort the result set in ascending order on firstname.

SELECT businessentityid, persontype, firstname, middlename,lastname

from person.person

WHERE middlename IS not NULL

order by firstname;



--175. From the following table write a query in SQL to identify the rows with a middlename that is not NULL. Return businessentityid, persontype, firstname, middlename,and lastname. Sort the result set in ascending order on firstname.

SELECT businessentityid, persontype, firstname, isnull(middlename,'') middlename,lastname

from person.person

WHERE middlename IS NULL

order by firstname;



--176. From the following table write a query in SQL to fetch all products with a weight of less than 10 pounds or unknown color.

--Return the name, weight, and color for the product. Sort the result set in ascending order on name.

SELECT Name,Weight, isnull(Color,'') Color

FROM Production.Product

WHERE Weight < 10 OR Color IS NULL

ORDER BY Name;



--177. From the following table write a query in SQL to list the salesperson whose salesytd begins with 1. Convert SalesYTD and current date in text format.

SELECT BusinessEntityID,

SalesYTD,

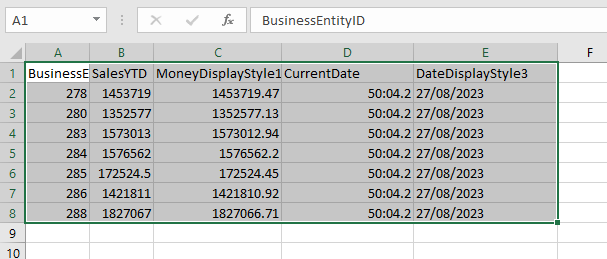
CONVERT(varchar, SalesYTD) AS MoneyDisplayStyle1,

GETDATE() AS CurrentDate,

CONVERT(varchar, GETDATE(), 103) AS DateDisplayStyle3

FROM Sales.SalesPerson

WHERE CONVERT(varchar, SalesYTD) LIKE '1%';



--178. From the following table write a query in SQL to return the count of employees by Name and Title, Name, and company total.

--Filter the results by department ID 12 or 14. For each row, identify its aggregation level in the Title column.

SELECT HRD.Name,

isnull(HRE.JobTitle,'') JobTitle ,

COUNT(\*) AS [Employee Count]

FROM HumanResources.Employee HRE

JOIN HumanResources.EmployeeDepartmentHistory HREDH

ON HRE.BusinessEntityID = HREDH.BusinessEntityID

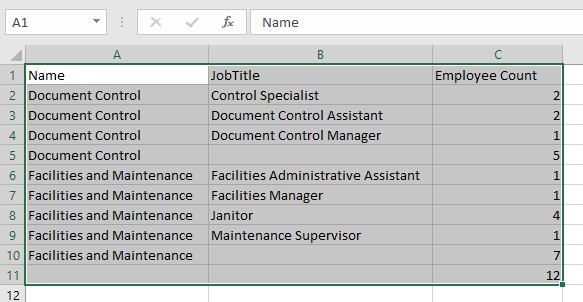
JOIN HumanResources.Department HRD

ON HRD.DepartmentID = HREDH.DepartmentID

WHERE HREDH.EndDate IS NULL

AND HRD.DepartmentID IN (12,14)

GROUP BY ROLLUP(HRD.Name, HRE.JobTitle);



--179. From the following tables write a query in SQL to return only rows with a count of employees by department. Filter the results by department ID 12 or 14. Return name, jobtitle, grouping level and employee count.

SELECT HRD.Name,

isnull(HRE.JobTitle,'') JobTitle ,

GROUPING(HRE.JobTitle) AS "Grouping Level" ,

COUNT(\*) AS [Employee Count]

FROM HumanResources.Employee HRE

JOIN HumanResources.EmployeeDepartmentHistory HREDH

ON HRE.BusinessEntityID = HREDH.BusinessEntityID

JOIN HumanResources.Department HRD

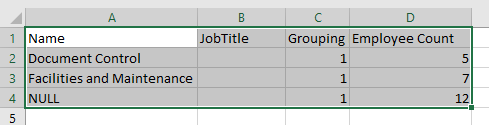
ON HRD.DepartmentID = HREDH.DepartmentID

WHERE HREDH.EndDate IS NULL

AND HRD.DepartmentID IN (12,14)

GROUP BY ROLLUP(HRD.Name, HRE.JobTitle)

HAVING GROUPING(HRE.JobTitle) = 1;



--180. From the following tables write a query in SQL to return only the rows that have a count of employees by title.

--Filter the results by department ID 12 or 14. Return name, jobtitle, grouping level and employee count.

SELECT HRD.Name,

isnull(HRE.JobTitle,'') JobTitle ,

GROUPING(HRE.JobTitle) AS "Grouping Level" ,

COUNT(\*) AS [Employee Count]

FROM HumanResources.Employee HRE

JOIN HumanResources.EmployeeDepartmentHistory HREDH

ON HRE.BusinessEntityID = HREDH.BusinessEntityID

JOIN HumanResources.Department HRD

ON HRD.DepartmentID = HREDH.DepartmentID

WHERE HREDH.EndDate IS NULL

AND HRD.DepartmentID IN (12,14)

GROUP BY ROLLUP(HRE.JobTitle,HRD.Name)

HAVING GROUPING(HRD.Name) = 0;

--181. From the following table write a query in SQL to return the difference in sales quotas for a specific employee over previous calendar quarters.

--Sort the results by salesperson with businessentity id 277 and quotadate year 2012 or 2013.

alter proc newProc

as

begin

declare @Year int

declare @quarter int

SELECT

@Year= DATEPART(YEAR,quotadate),

@quarter=DATEPART(QUARTER, quotadate)

FROM sales.salespersonquotahistory

WHERE businessentityid = 277 AND @Year IN (2012, 2013)

SELECT

@Year AS Year,

@quarter AS Quarter,

SalesQuota,

LAG(SalesQuota) OVER (ORDER BY @Year, @quarter) AS PrevQuota,

SalesQuota - LAG(SalesQuota) OVER (ORDER BY @Year,@quarter) AS Diff

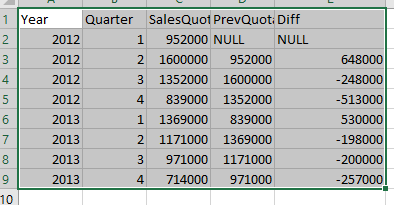
FROM sales.salespersonquotahistory

WHERE businessentityid = 277 AND @Year IN (2012, 2013)

ORDER BY DATEPART(YEAR, quotadate), DATEPART(QUARTER, quotadate);

end

exec newProc



--182. From the following table write a query in SQL to return a truncated date with 4 months added to the orderdate.

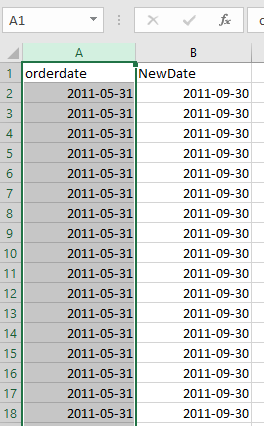
SELECT

orderdate,

DATEADD(MONTH, 4, orderdate) AS NewDate

FROM

Sales.salesorderheader;



--183. From the following table write a query in SQL to return the orders that have sales on or after December 2011. Return salesorderid, MonthOrderOccurred, salespersonid, customerid, subtotal, Running Total, and actual order date.

SELECT

salesorderid,

DATEADD (MONTH, DATEDIFF (MONTH, 0, orderdate), 0) AS MonthOrderOccurred,

salespersonid,

customerid,

subtotal,

SUM (subtotal) OVER (PARTITION BY customerid ORDER BY orderdate, salesorderid

ROWS UNBOUNDED PRECEDING

) AS RunningTotal,

orderdate AS ActualOrderDate

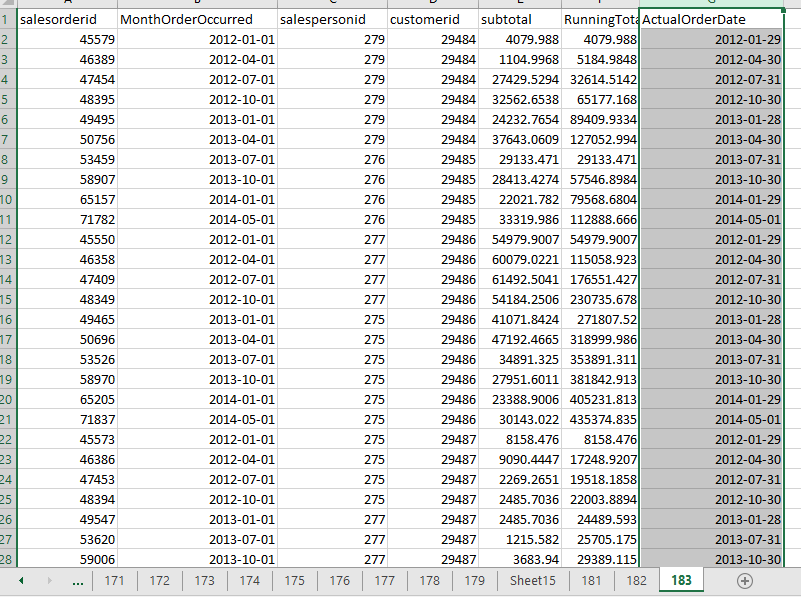
FROM

Sales.salesorderheader

WHERE

salespersonid IS NOT NULL

AND DATEADD (MONTH, DATEDIFF (MONTH, 0, orderdate), 0) >= '2011-12-01';



--184. From the following table write a query in SQL to repeat the 0 character four times before productnumber.

--Return name, productnumber and newly created productnumber.

SELECT

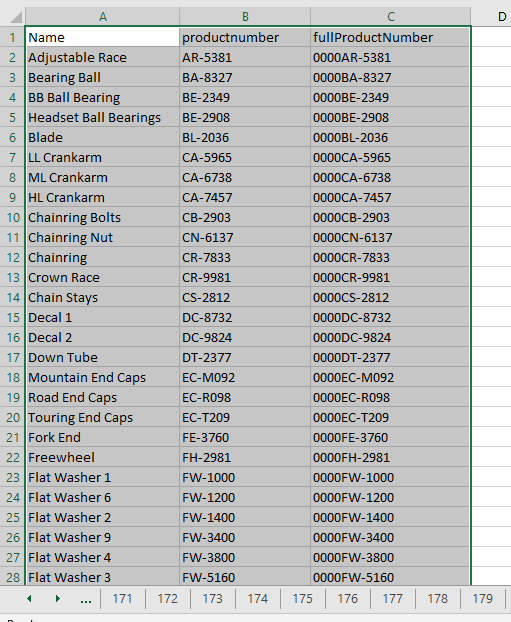
Name,

productnumber,

CONCAT('0000', productnumber) AS fullProductNumber

FROM

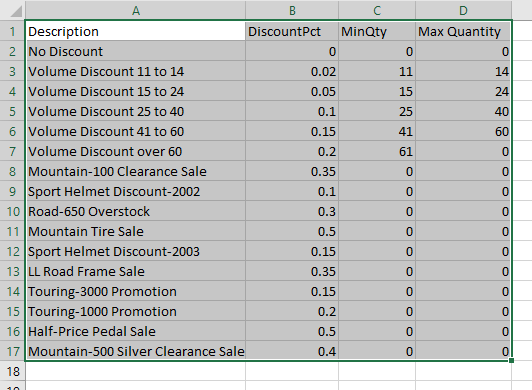
Production.Product;



--185. From the following table write a query in SQL to find all special offers.

--When the maximum quantity for a special offer is NULL, return MaxQty as zero.

select Description, DiscountPct, MinQty,isnull(MaxQty, 0.00) as [Max Quantity] from Sales.SpecialOffer

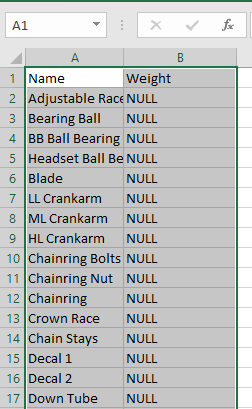


--186. From the following table write a query in SQL to find all products that have NULL in the weight column. Return name and weight.

SELECT Name, Weight

FROM Production.Product

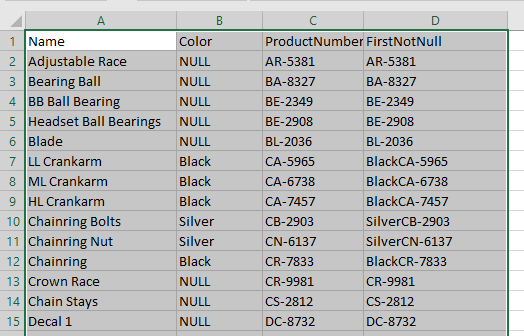
WHERE Weight is null;



--187. From the following table write a query in SQL to find the data from the first column that has a non-null value. Return name, color, productnumber, and firstnotnull column.

SELECT Name, Color, ProductNumber, COALESCE (Color, ProductNumber) AS FirstNotNull

FROM production.Product ;



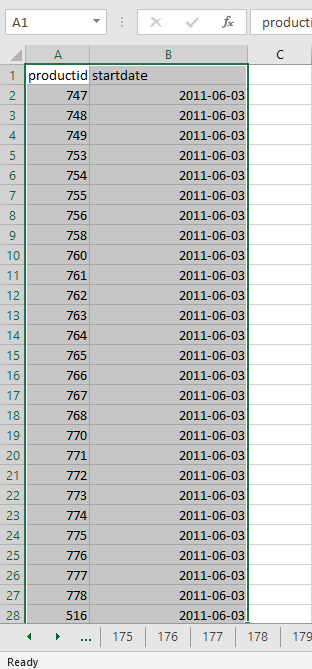
--188. From the following tables write a query in SQL to return rows only when both the productid and startdate values in the two tables matches.

SELECT a.productid, a.startdate

FROM production.workorder AS a

JOIN production.workorderrouting AS b

ON a.productid = b.productid AND a.startdate = b.actualstartdate;



--189. From the following tables write a query in SQL to return rows except both the productid and startdate

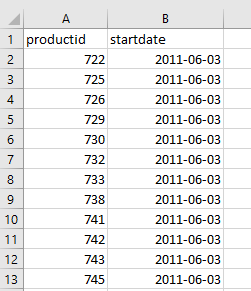
--values in the two tables matches.

SELECT a.productid, a.startdate

FROM production.workorder AS a

JOIN production.workorderrouting AS b

ON a.productid = b.productid AND a.startdate = b.actualstartdate;



--190. From the following table write a query in SQL to find all creditcardapprovalcodes starting with 1 and the third digit is 6. Sort the result set in ascending order on orderdate.

create proc modif

as

begin

SELECT salesorderid, orderdate, creditcardapprovalcode

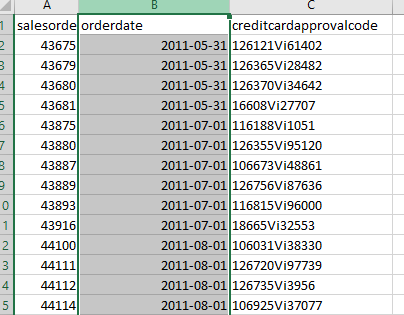
FROM sales.salesorderheader

WHERE creditcardapprovalcode LIKE '1\_6%'

ORDER by orderdate;

end

exec modif

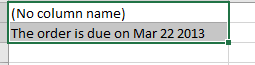


--191. From the following table write a query in SQL to concatenate character and date data types for the order ID 50001.

SELECT ('The order is due on ' + cast(DueDate as VARCHAR(20)))

FROM Sales.SalesOrderHeader

WHERE SalesOrderID = 50001;



--192. From the following table write a query in SQL to form one long string to display the last name and the first initial of the vice presidents. Sort the result set in ascending order on lastname.

SELECT (LastName +', ' + SUBSTRING(FirstName, 1, 1)+ '.') AS Name, HRE.JobTitle

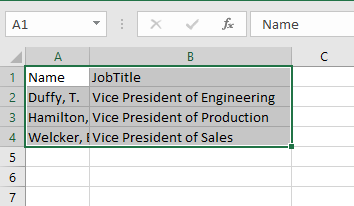
FROM Person.Person AS PP

JOIN HumanResources.Employee AS HRE

ON PP.BusinessEntityID = HRE.BusinessEntityID

WHERE HRE.JobTitle LIKE 'Vice President%'

ORDER BY LastName;

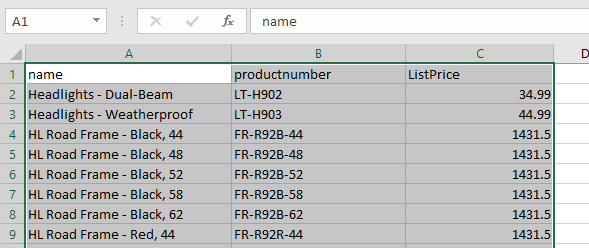


--193. From the following table write a query in SQL to return only the rows for Product that have a product line of R and that have days to manufacture that is less than 4. Sort the result set in ascending order on name.

select name,productnumber,ListPrice

from Production.Product

where ProductLine like 'R%' and DaysToManufacture<4

order by Name

--194. From the following tables write a query in SQL to return total sales and the discounts for each product. Sort the result set in descending order on productname.

SELECT

ProductName,

OrderQty \* UnitPrice AS NonDiscountSales,

OrderQty \* UnitPrice \* UnitPriceDiscount AS Discounts

FROM

(

SELECT

PP.Name AS ProductName,

SSOD.OrderQty,

SSOD.UnitPrice,

SSOD.UnitPriceDiscount

FROM

Production.Product AS PP

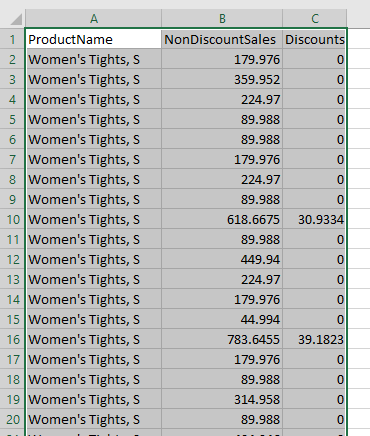
JOIN

Sales.SalesOrderDetail AS SSOD ON PP.ProductID = SSOD.ProductID

) AS SubQuery

ORDER BY

ProductName DESC;



--195. From the following tables write a query in SQL to calculate the revenue for each product in each sales order. Sort the result set in ascending order on productname.

SELECT 'Total income is ',

((SSOD.OrderQty \* SSOD.UnitPrice) - (SSOD.OrderQty \* SSOD.UnitPrice\* SSOD.UnitPriceDiscount)),

' for',

PP.Name AS ProductName

FROM

(

SELECT ProductID,

OrderQty,

UnitPrice,

UnitPriceDiscount

FROM Sales.SalesOrderDetail

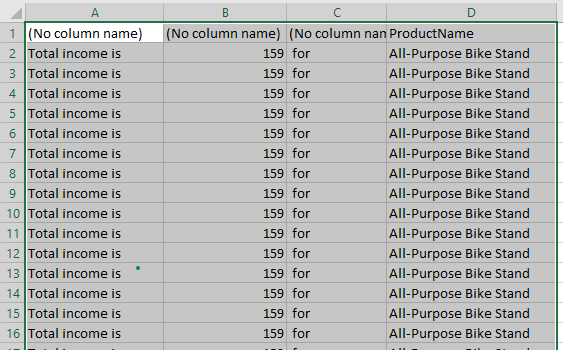
) AS SSOD

JOIN Production.Product AS PP

ON PP.ProductID = SSOD.ProductID

ORDER BY

PP.Name ;



--196. From the following tables write a query in SQL to retrieve one instance of each product name whose product model is a long sleeve logo jersey, and the ProductModelID numbers match between the tables.

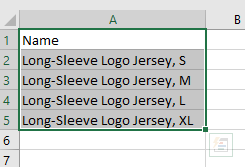
SELECT PP.Name

FROM Production.Product AS PP

JOIN Production.ProductModel AS PPM

ON PP.ProductModelID = PPM.ProductModelID

WHERE PPM.Name LIKE 'Long-Sleeve Logo Jersey%';



--197. From the following tables write a query in SQL to retrieve the first and last name of each employee whose bonus in the SalesPerson table is 5000.

SELECT PP.LastName, PP.FirstName

FROM Person.Person AS PP

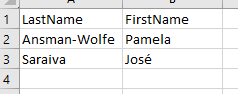
JOIN HumanResources.Employee AS HRE

ON HRE.BusinessEntityID = PP.BusinessEntityID

join Sales.SalesPerson AS SSP

ON HRE.BusinessEntityID = SSP.BusinessEntityID

WHERE SSP.Bonus= 5000



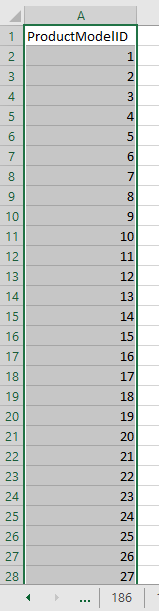
--198. From the following table write a query in SQL to find product models where the maximum list price is more than twice the average.

SELECT ProductModelID

FROM Production.Product

GROUP BY ProductModelID

HAVING MAX(ListPrice) <= 2 \* AVG(ListPrice);



--199. From the following table write a query in SQL to find the names of employees who have sold a particular product.

alter proc newfunc

@productame varchar(50)

as

begin

SELECT DISTINCT pp.LastName, pp.FirstName,p.Name

FROM Person.Person pp

JOIN HumanResources.Employee e

ON e.BusinessEntityID = pp.BusinessEntityID

JOIN Sales.SalesOrderHeader soh

ON soh.SalesPersonID = pp.BusinessEntityID

JOIN Sales.SalesOrderDetail sod

ON sod.SalesOrderID = soh.SalesOrderID

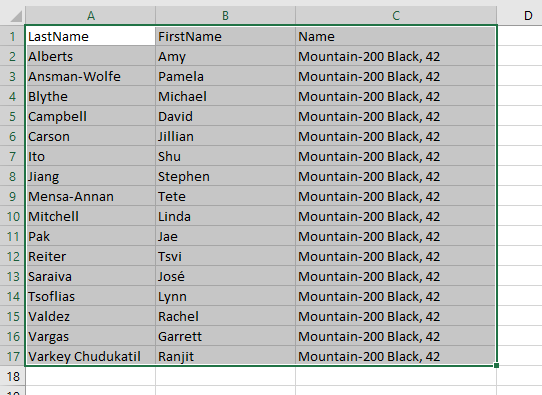
JOIN Production.Product p

ON p.ProductID = sod.ProductID

WHERE p.Name = @productame;

end

exec newfunc @productame = 'Mountain-200 Black, 42';



--200. Create a table public.gloves from Production.ProductModel for the ProductModelID 3 and 4.From the following table write a query in SQL to include the contents of the ProductModelID and Name columns of both the tables.

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Gloves' AND schema\_id = SCHEMA\_ID('public'))

BEGIN

SELECT ProductModelID, Name

INTO puplic.Gloves

FROM Production.ProductModel

WHERE ProductModelID IN (3, 4);

END;

select \* from puplic.Gloves

