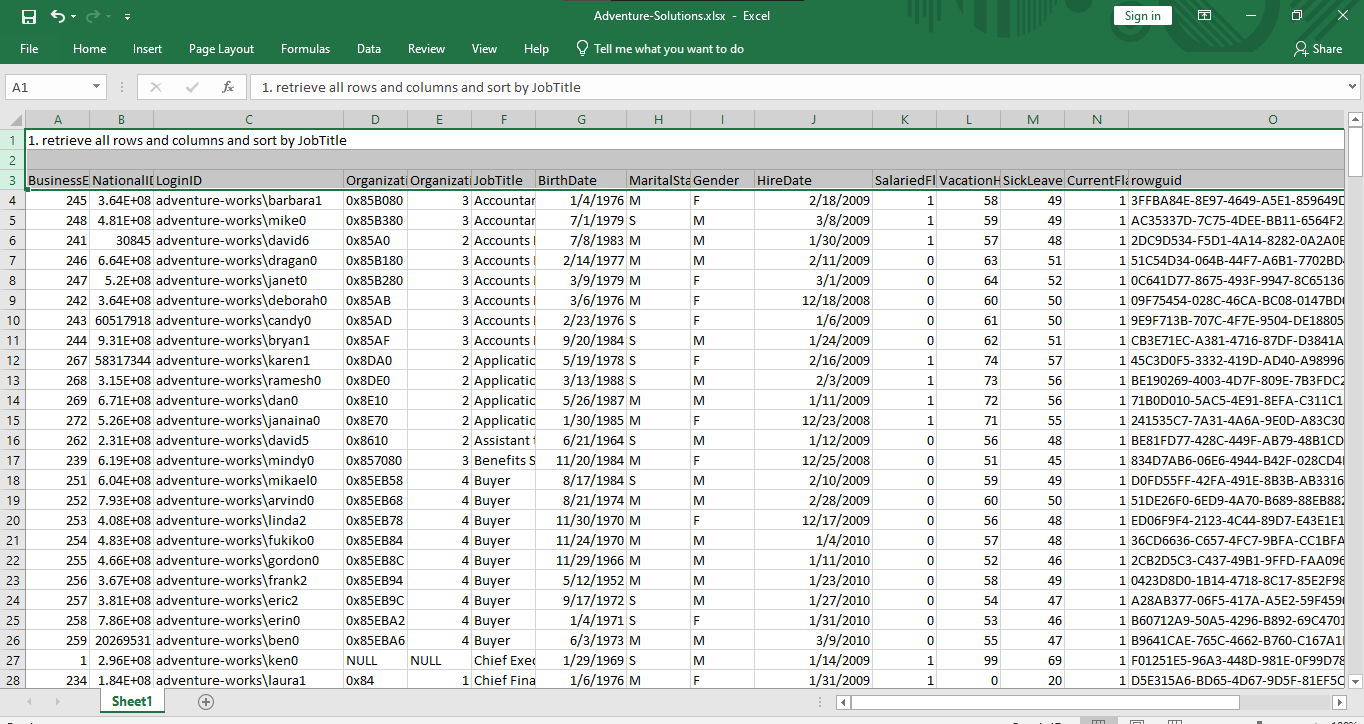
1. From the following table write a query in SQL to retrieve all rows and columns from the employee table in the Adventureworks database. Sort the result set in ascending order on jobtitle.

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 ,

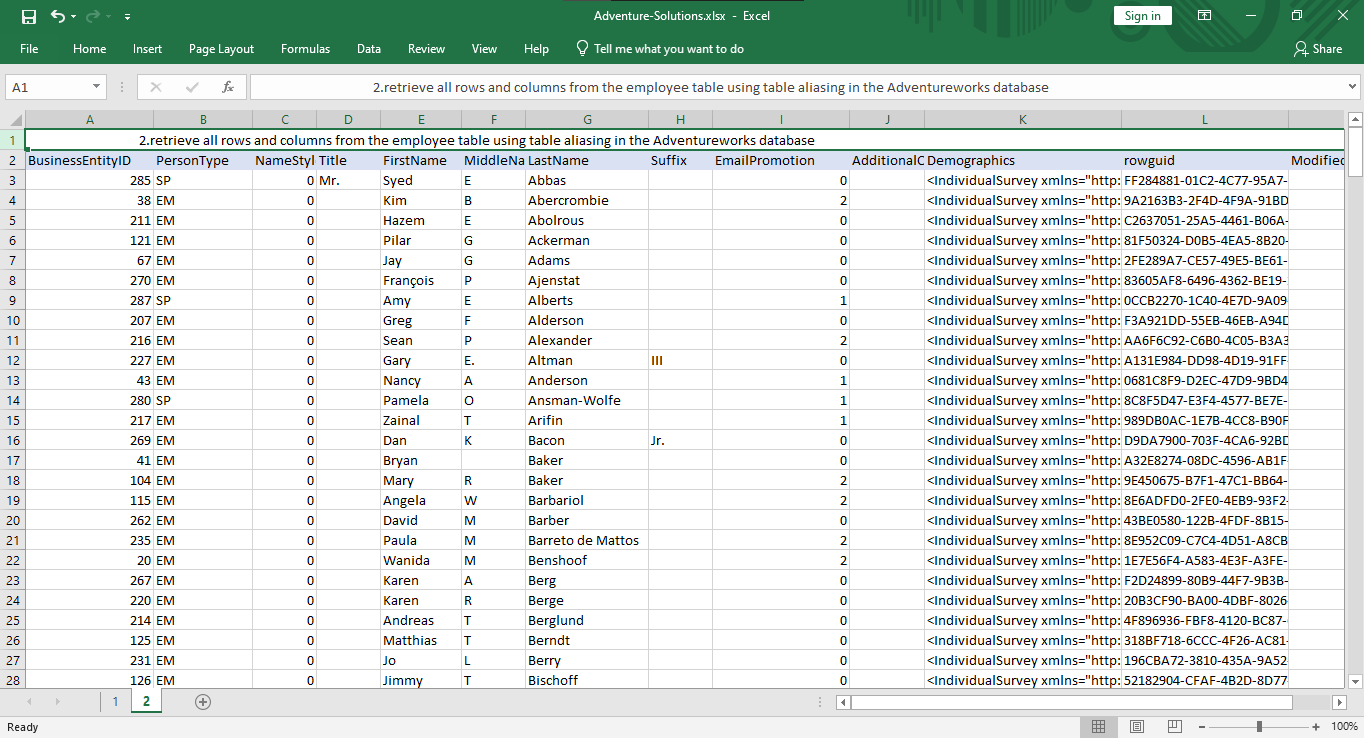
ISNULL(P.ModifiedDate,'')as ModifiedDate

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.

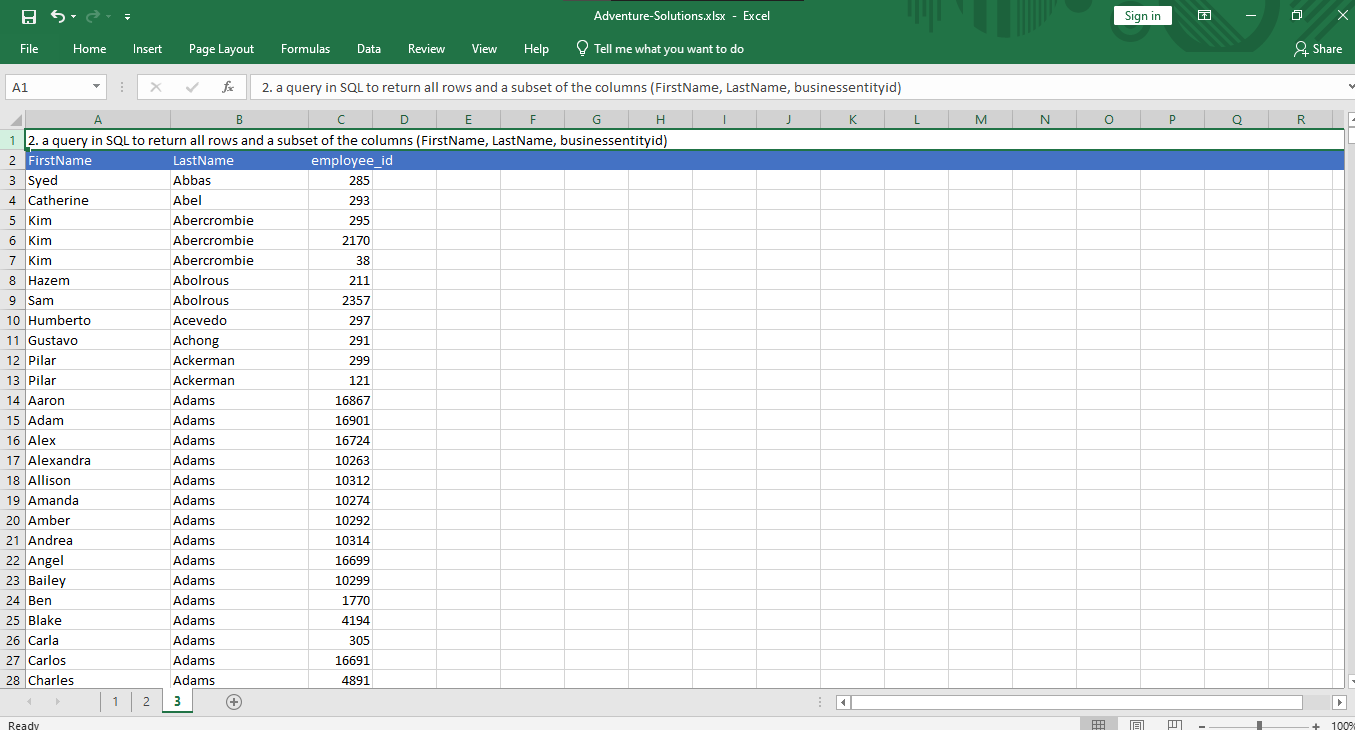
select P.FirstName,

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,

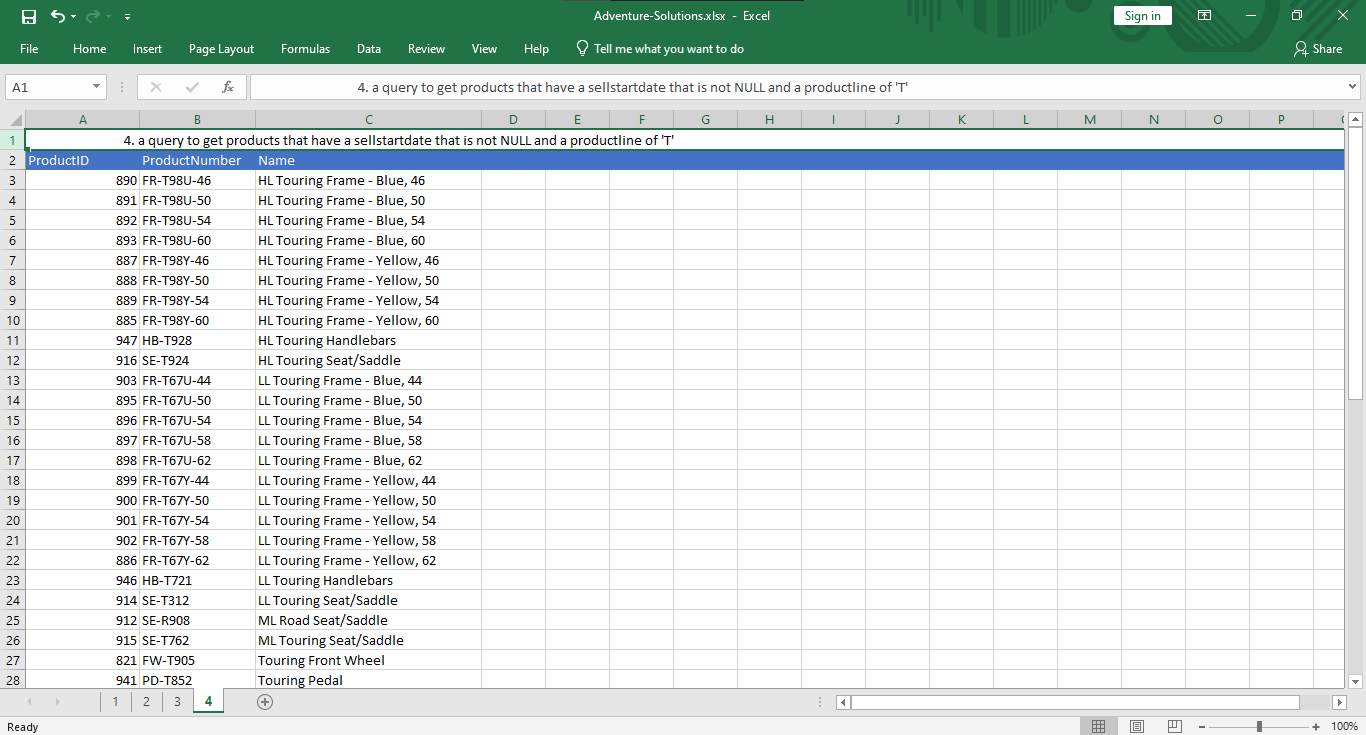
P.ProductNumber,

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,

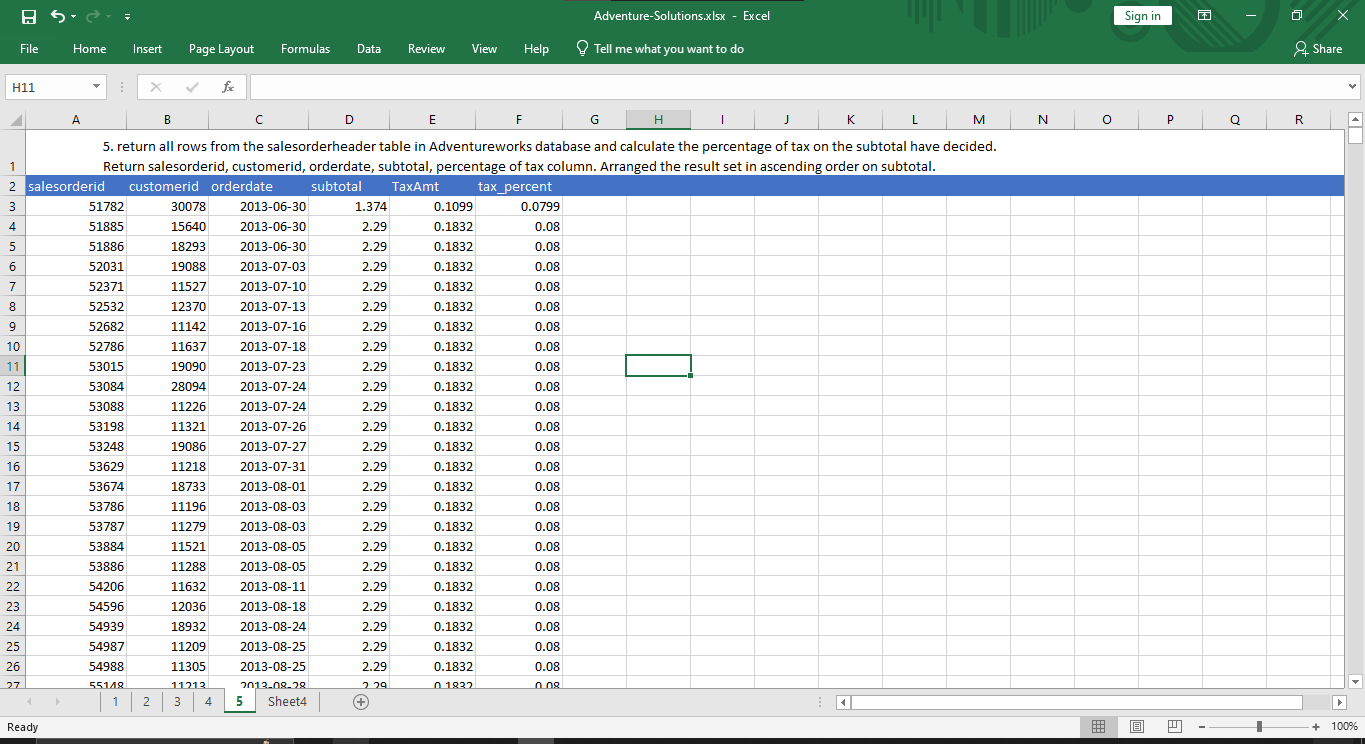
subtotal,

TaxAmt,

(TaxAmt/subtotal) as tax\_percent

from sales.salesorderheader S

order by subtotal asc



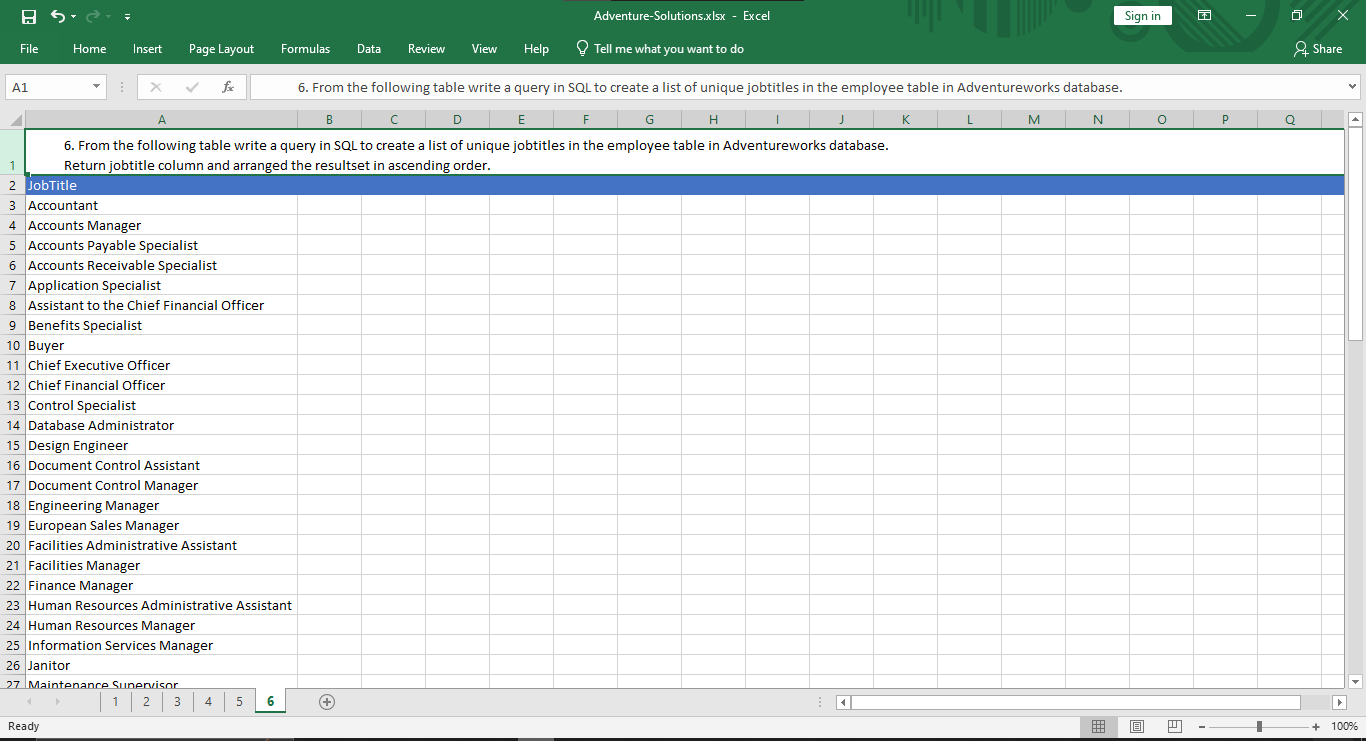
===========================================================================

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



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

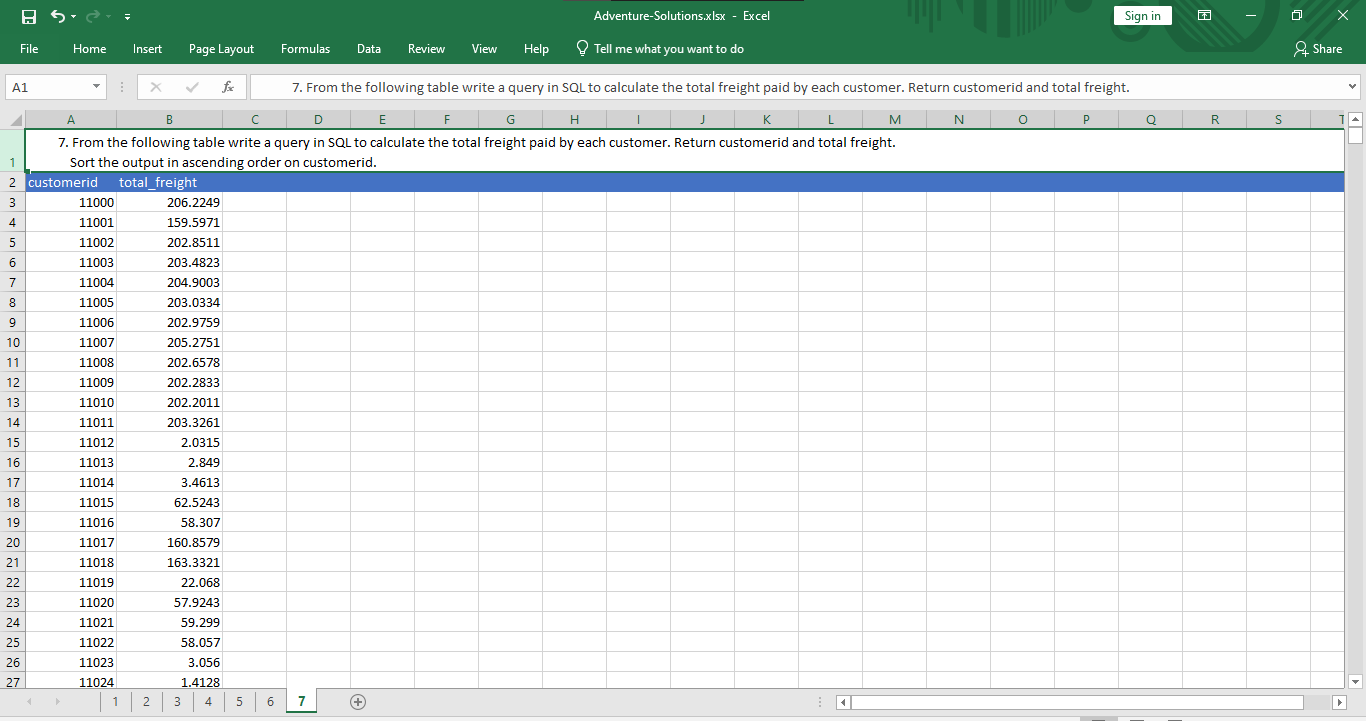
7. From the following table write a query in SQL to calculate the total freight paid by each customer. Return customerid and total freight. Sort the output in ascending order on customerid.

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,

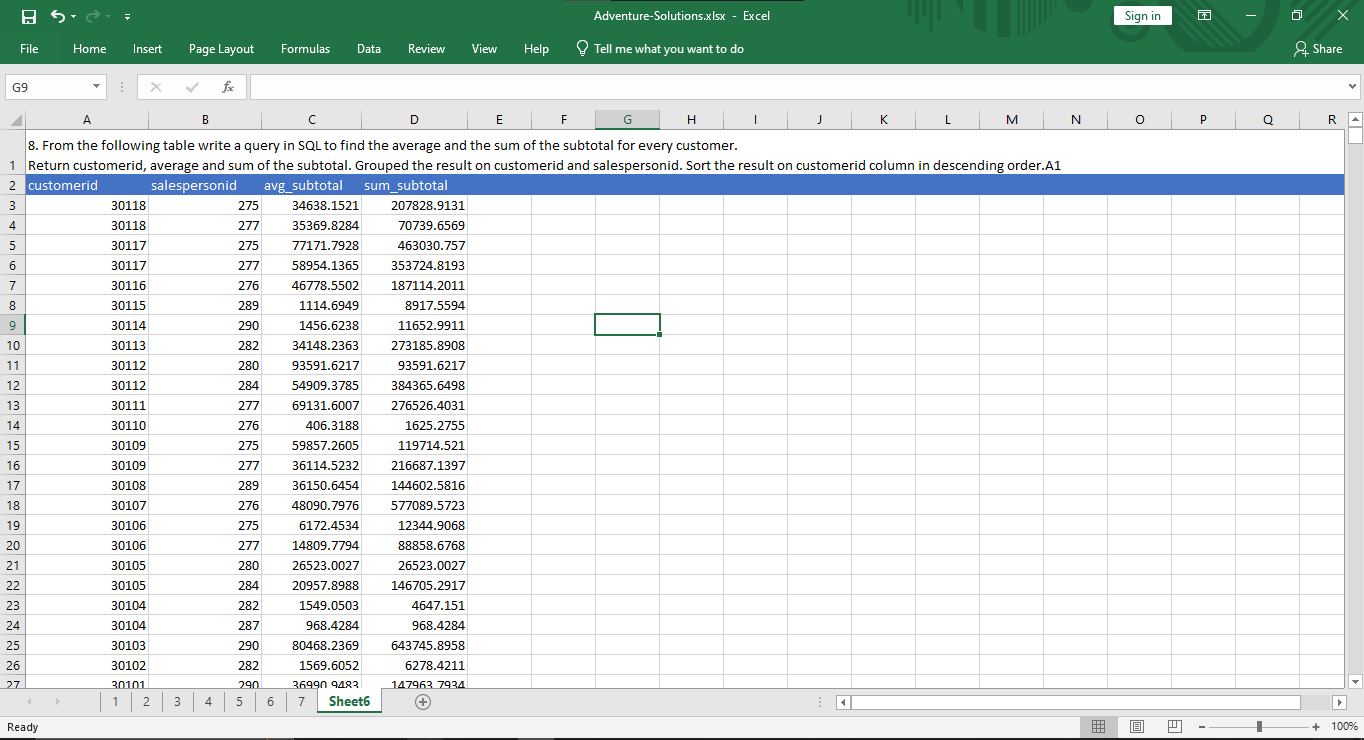
(sum(subtotal)/count(\*)) as avg\_subtotal,

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,

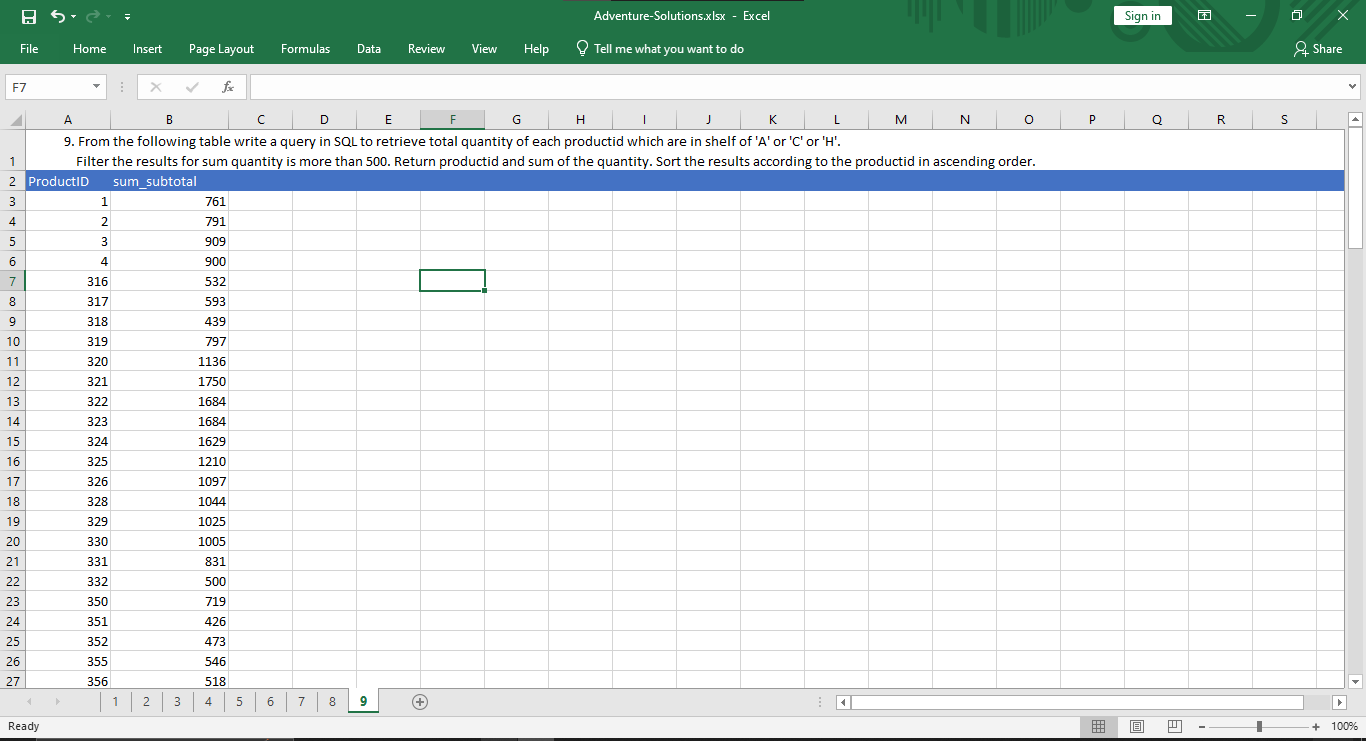
sum(Quantity) as sum\_subtotal

from production.productinventory

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

group by ProductID

order by productid asc



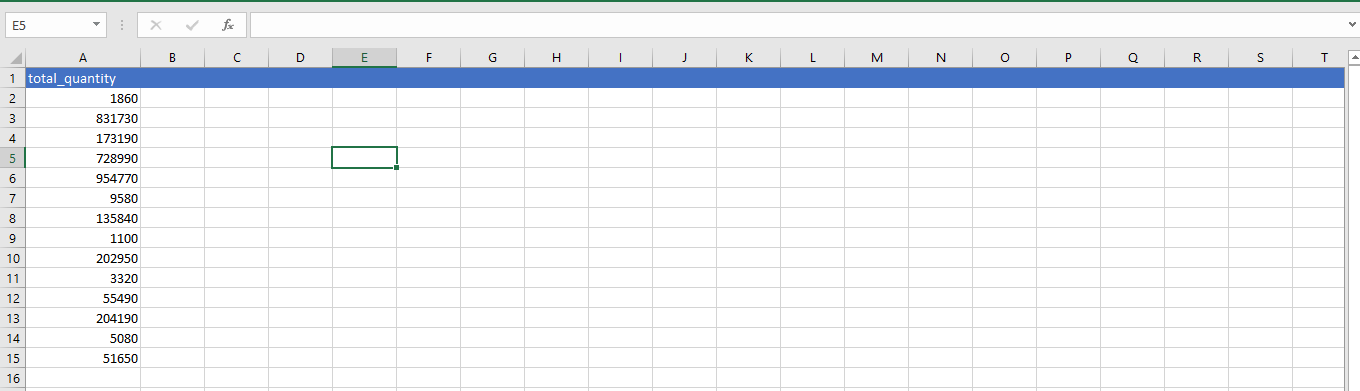
=========================================================================

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

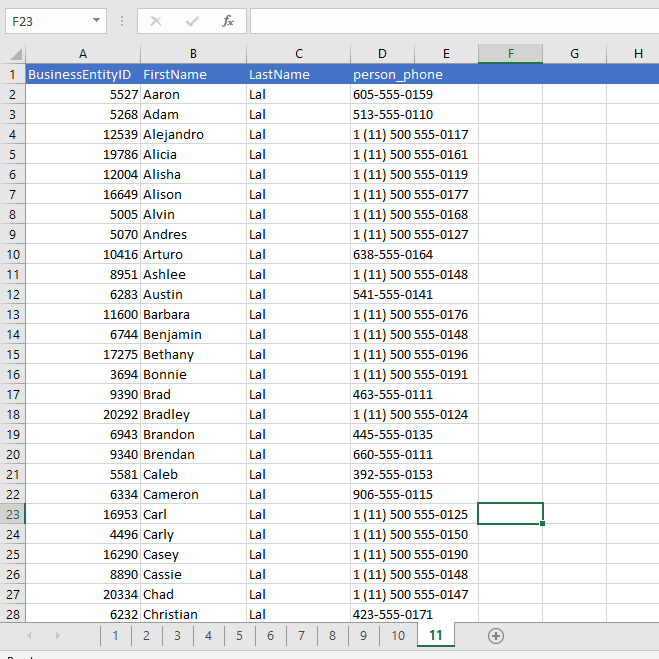
from Person.Person P

join Person.PersonPhone F

on P.BusinessEntityID=F.BusinessEntityID

where LastName like 'L%'

order by LastName,FirstName



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

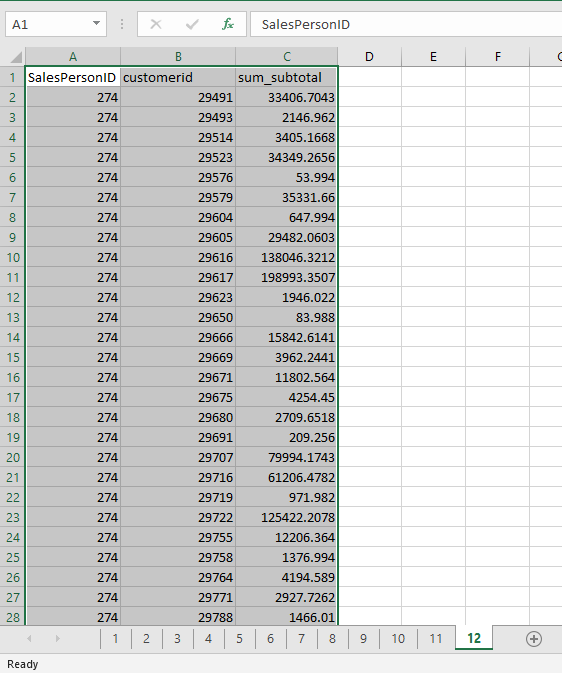
12. From the following table write a query in SQL to find the sum of subtotal column. Group the sum on distinct salespersonid and customerid. Rolls up the results into subtotal and running total. Return salespersonid, customerid and sum of subtotal column i.e. sum\_subtotal.

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

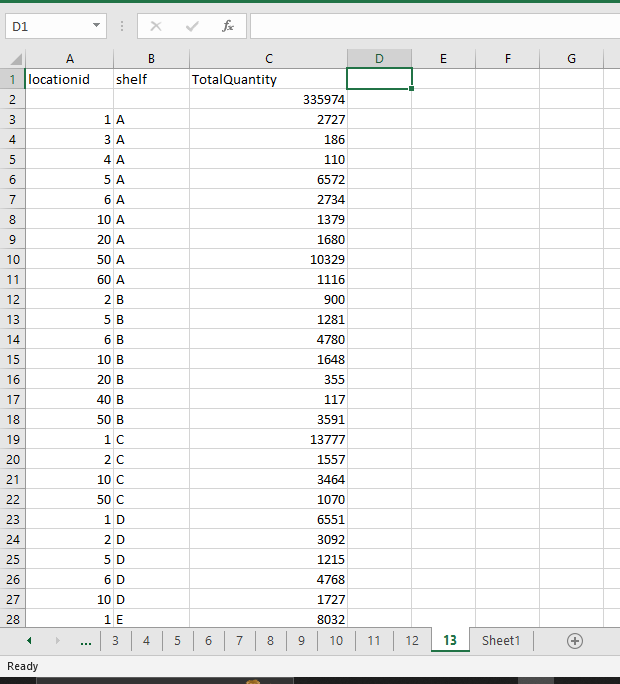
GROUP BY GROUPING SETS

(

(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

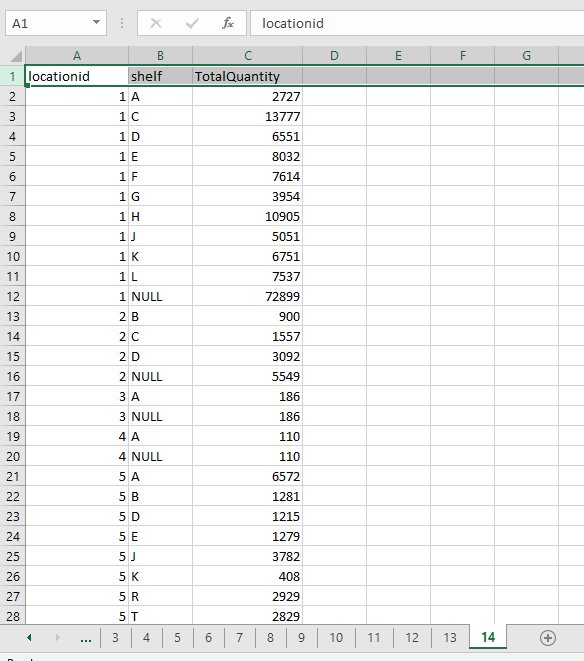
GROUP BY CUBE (locationid, shelf);

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

SELECT locationid, shelf, SUM (quantity) AS TotalQuantity

FROM production.productinventory

GROUP BY ROLLUP (locationid, Shelf)



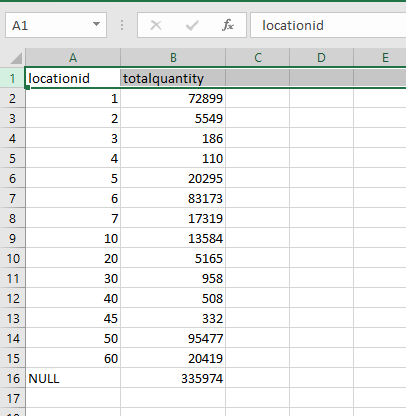
===========================================================================

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,( ))



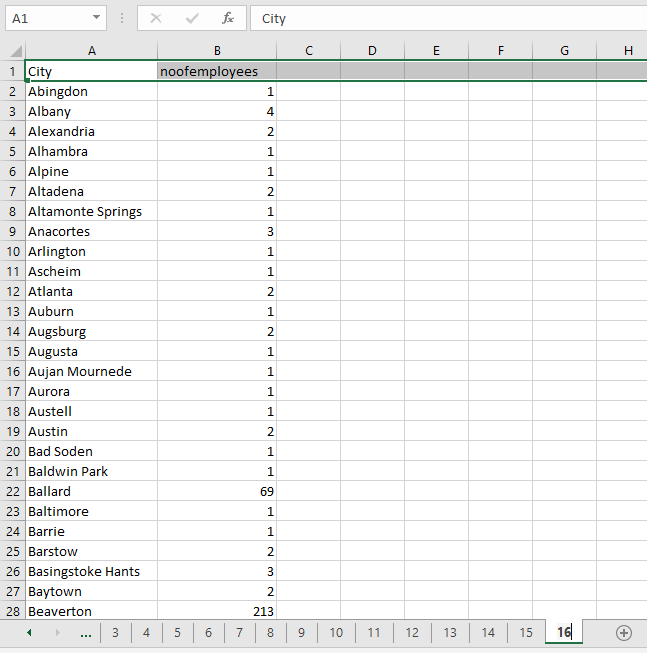
==========================================================================

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.

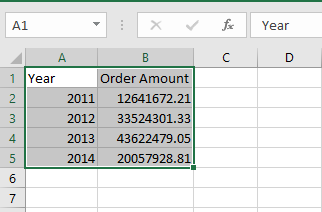
select YEAR(OrderDate) as [Year],

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],

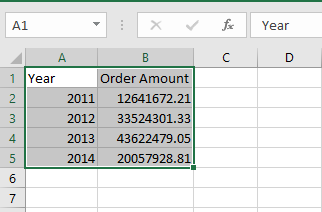
SUM(SubTotal) as [Order Amount]

from Sales.SalesOrderHeader

where YEAR(OrderDate) < '2016'

group by YEAR(OrderDate)

order by YEAR(OrderDate)



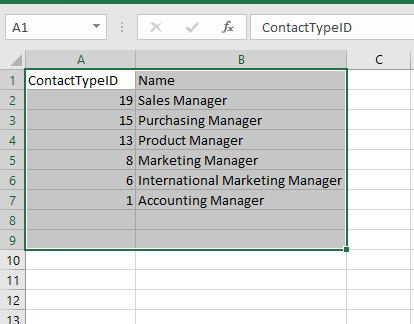
==========================================================================

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

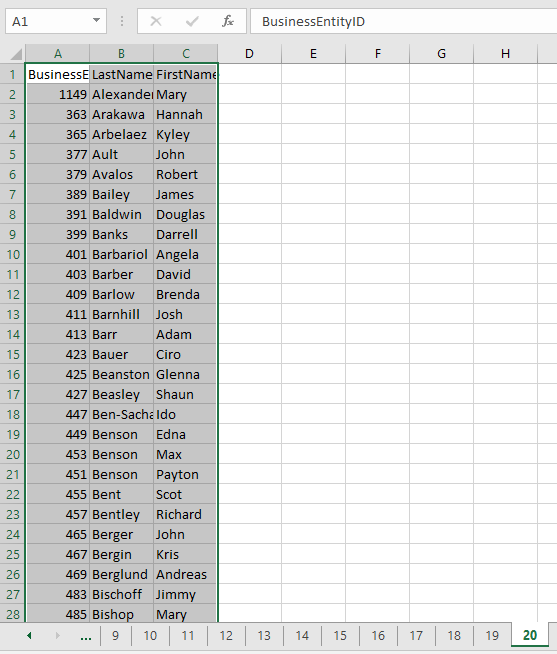
ON pc.ContactTypeID = pb.ContactTypeID

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

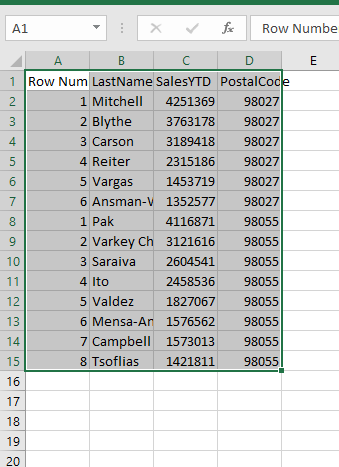
join person.Address pa

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

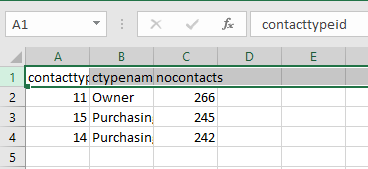
join Person.ContactType pc

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,

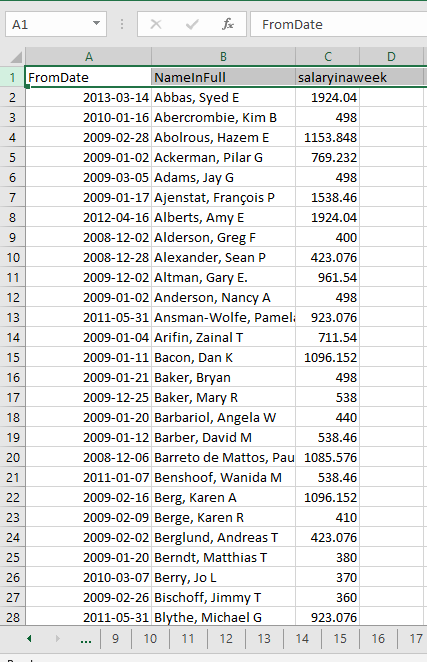
(HE.Rate\*40) salaryinaweek

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

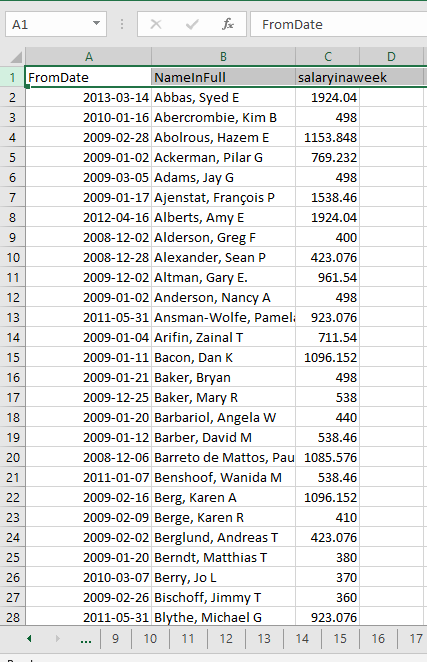
WHERE HE.RateChangeDate = (SELECT MAX(RateChangeDate)

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],

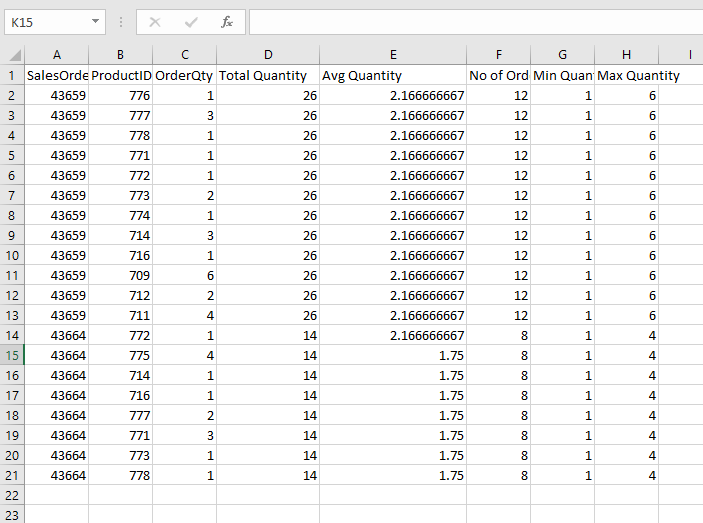
count(\*) OVER (PARTITION BY ss.SalesOrderId ) [No of Orders],

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,

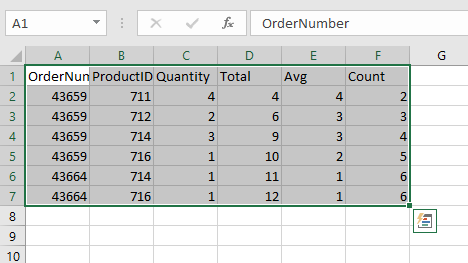
SUM(OrderQty) OVER (ORDER BY SalesOrderID, ProductID) AS Total,

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.

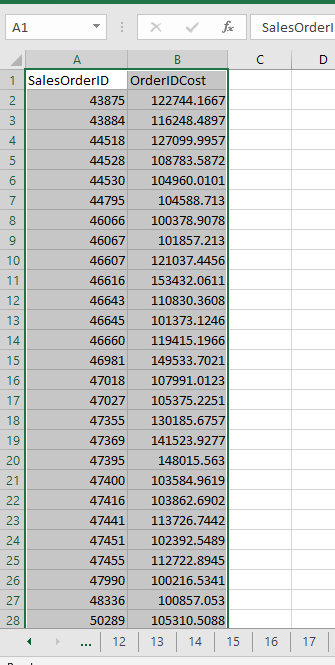
SELECT SalesOrderID, SUM (orderqty\*unitprice) AS OrderIDCost

FROM Sales.SalesOrderDetail

GROUP BY SalesOrderID

HAVING SUM(orderqty\*unitprice) > 100000.00

ORDER BY SalesOrderID;



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

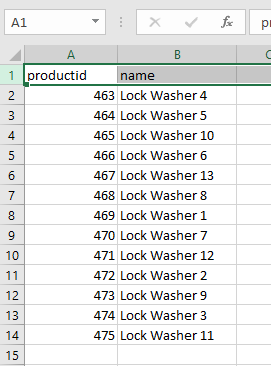
28. From the following table write a query in SQL to retrieve products whose names start with 'Lock Washer'. Return product ID, and name and order the result set in ascending order on product ID column.

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;



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

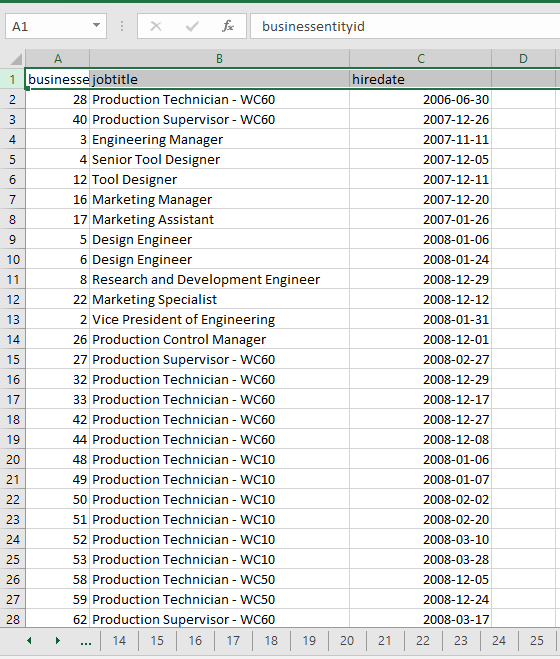
30. From the following table write a query in SQL to retrieve records of employees.

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)



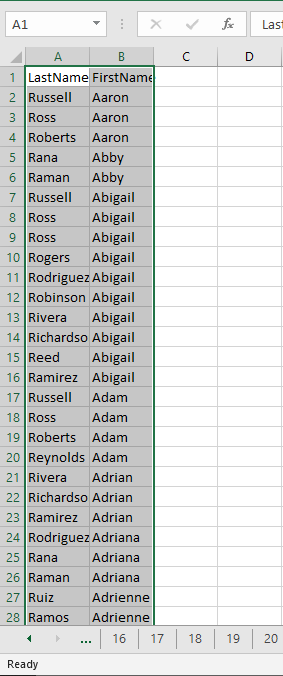
31. From the following table write a query in SQL to retrieve those persons whose last name begins with letter 'R'. Return lastname,and firstname and display the result in ascending order on firstname and descending order on lastname columns.

SELECT LastName, FirstName

FROM Person.Person

WHERE LastName LIKE 'R%'

ORDER BY FirstName ASC, LastName DESC;



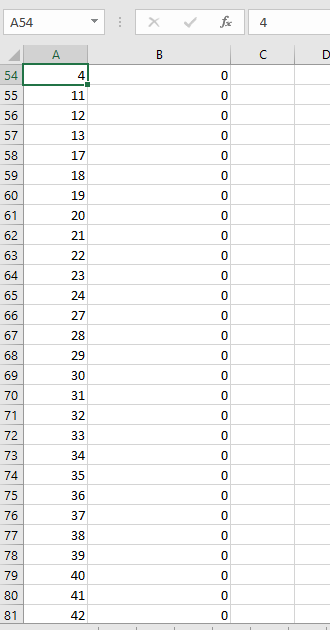
32. From the following table write a query in SQL to ordered the BusinessEntityID column descendingly when SalariedFlag set to 'true' and BusinessEntityID in ascending order when SalariedFlag set to 'false'. Return BusinessEntityID, SalariedFlag columns.

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.

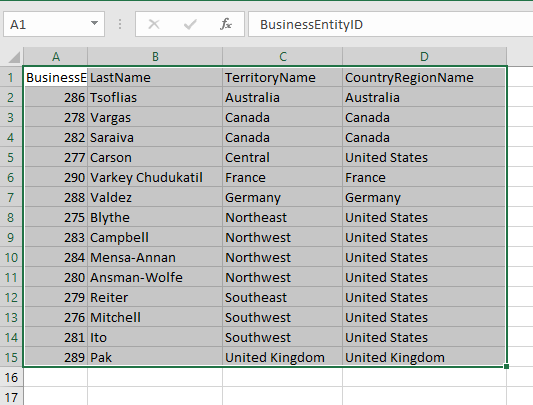
SELECT BusinessEntityID, LastName, TerritoryName, CountryRegionName

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

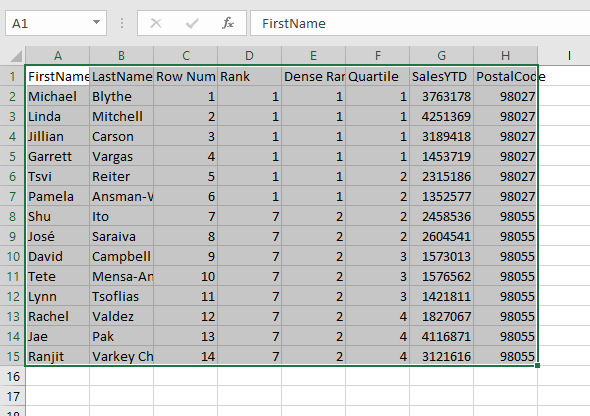
INNER JOIN Person.Person AS p

ON s.BusinessEntityID = p.BusinessEntityID

INNER JOIN Person.Address AS a

ON a.AddressID = p.BusinessEntityID

WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0;



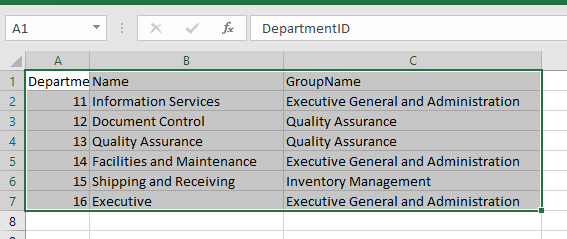
==========================================================================

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.

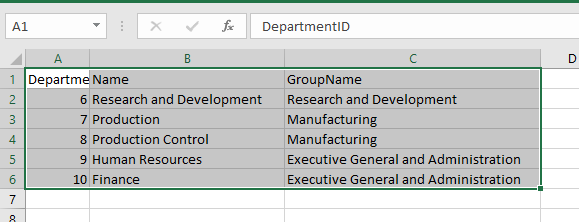
SELECT DepartmentID, Name, GroupName

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'

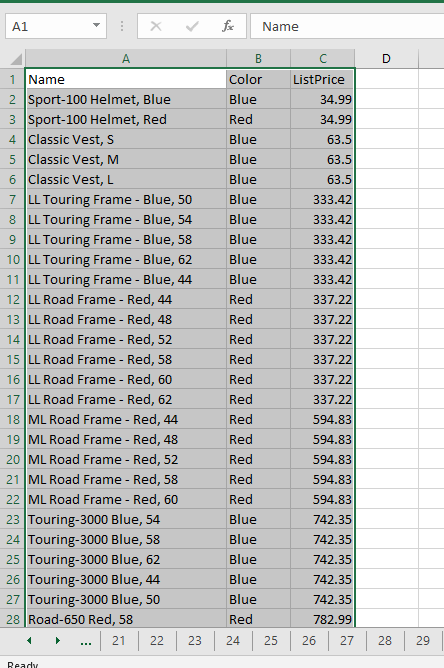
UNION ALL

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.

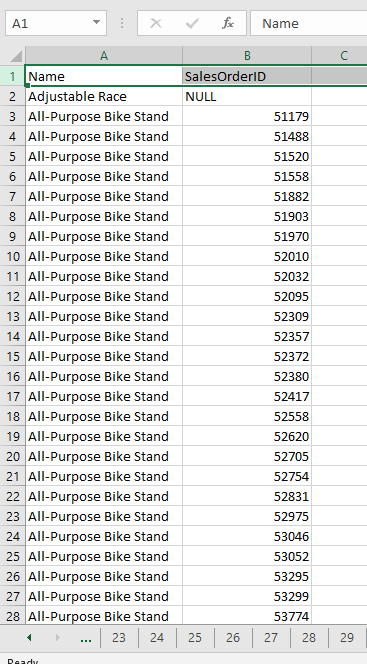
SELECT pp.Name, SS.SalesOrderID

FROM Production.Product AS pp

FULL OUTER 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.

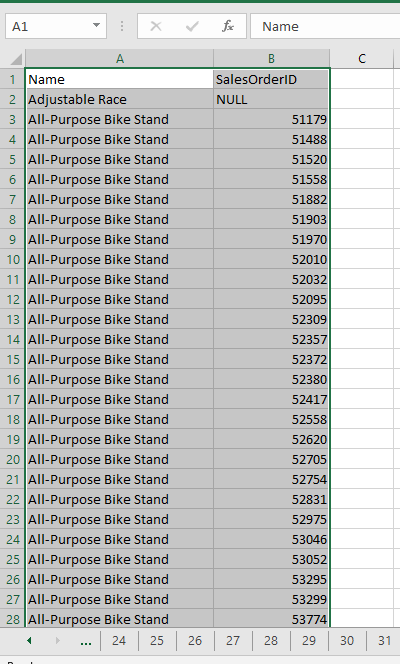
SELECT pp.Name, ss.SalesOrderID

FROM Production.Product AS pp

LEFT OUTER 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

INNER JOIN Sales.SalesOrderDetail AS ss

ON pp.ProductID = ss.ProductID

ORDER BY pp.Name ;

