```sql
/*1) 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 ASC;


/*2)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.*
FROM Person.Person AS e
ORDER BY LastName ASC;

/*3)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 FirstName, LastName, BusinessEntityID AS Employee_id
FROM Person.Person
ORDER BY LastName ASC;

/*4)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 ProductID, ProductNumber, Name
FROM Production.Product
WHERE SellStartDate IS NOT NULL
  AND ProductLine = 'T'
ORDER BY Name ASC;

/*5)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 / SubTotal) * 100 AS Tax_Percent
FROM Sales.SalesOrderHeader
ORDER BY SubTotal ASC;

/*6)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)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)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. Group the result on customerid and
salespersonid. Sort the result on customerid column in descending order.*/

SELECT CustomerID, SalesPersonID,
       AVG(SubTotal) AS Avg_SubTotal,
       SUM(SubTotal) AS Sum_SubTotal
FROM Sales.SalesOrderHeader
GROUP BY CustomerID, SalesPersonID
ORDER BY CustomerID DESC;

/*9)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 Total_Quantity
FROM Production.ProductInventory
WHERE Shelf IN ('A', 'C', 'H')
GROUP BY ProductID
HAVING SUM(Quantity) > 500
ORDER BY ProductID ASC;
```

```sql
/*10)Write a query in SQL to find the total quantity for a group of locationid
  multiplied by 10.*/

SELECT LocationID, SUM(Quantity) * 10 AS Total_Quantity
FROM Production.ProductInventory
GROUP BY LocationID;

/*11) 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, p.FirstName, p.LastName, ph.PhoneNumber
FROM Person.Person AS p
JOIN Person.PersonPhone AS ph
  ON p.BusinessEntityID = ph.BusinessEntityID
WHERE p.LastName LIKE 'L%'
ORDER BY p.LastName, p.FirstName;

/*12) 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
GROUP BY ROLLUP (SalesPersonID, CustomerID);

/*13)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 LocationID, Shelf
ORDER BY LocationID, Shelf;

/*14)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
```

```sql
GROUP BY ROLLUP (LocationID, Shelf);

/*15)Write a query in SQL to find the total quantity for each locationid and
  calculate the grandtotal for all locations.
Return locationid and total quantity. Group the results on locationid.*/

SELECT LocationID, SUM(Quantity) AS TotalQuantity
FROM Production.ProductInventory
GROUP BY LocationID
WITH ROLLUP;

/*16)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(*) AS NoOfEmployees
FROM Person.Address
GROUP BY City
ORDER BY City ASC;

/*17) 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(TotalDue) AS OrderAmount
FROM Sales.SalesOrderHeader
GROUP BY YEAR(OrderDate)
ORDER BY Year ASC;

/*18)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 YearOfOrderDate, SUM(TotalDue) AS TotalDueOrder
FROM Sales.SalesOrderHeader
WHERE YEAR(OrderDate) <= 2016
GROUP BY YEAR(OrderDate)
ORDER BY YearOfOrderDate ASC;

/*19)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
```

```sql
WHERE Name LIKE '%Manager%'
ORDER BY Name 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 p.BusinessEntityID, p.LastName, p.FirstName
FROM Person.Person AS p
JOIN Person.BusinessEntityContact AS bec
  ON p.BusinessEntityID = bec.PersonID
JOIN Person.ContactType AS ct
  ON bec.ContactTypeID = ct.ContactTypeID
WHERE ct.Name = 'Purchasing Manager'
ORDER BY p.LastName, p.FirstName;

SELECT


/*21)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. Shorts the postalcode in ascending order.*/

SELECT ROW_NUMBER() OVER (PARTITION BY a.PostalCode ORDER BY s.SalesYTD DESC) AS RowNumber,
      p.LastName, s.SalesYTD, a.PostalCode
FROM Sales.SalesPerson AS s
JOIN Person.Person AS p
  ON s.BusinessEntityID = p.BusinessEntityID
JOIN Person.Address AS a
  ON s.TerritoryID = a.StateProvinceID
WHERE s.SalesYTD <> 0
ORDER BY a.PostalCode ASC, s.SalesYTD DESC;

/*22)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 ct.ContactTypeID, ct.Name AS CTypeName, COUNT(*) AS NoContacts
FROM Person.BusinessEntityContact AS bec
```

```
JOIN Person.ContactType AS ct
  ON bec.ContactTypeID = ct.ContactTypeID
GROUP BY ct.ContactTypeID, ct.Name
HAVING COUNT(*) >= 100
ORDER BY NoContacts DESC;

/*23)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 RateChangeDate AS FromDate,
       p.LastName + ', ' + p.FirstName + ISNULL(' ' + p.MiddleName, '') AS
         NameInFull,
       Rate * 40 AS SalaryInAWeek
FROM HumanResources.EmployeePayHistory AS eph
JOIN Person.Person AS p
  ON eph.BusinessEntityID = p.BusinessEntityID
ORDER BY NameInFull ASC;

/*24)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 eph.RateChangeDate AS FromDate,
       p.LastName + ', ' + p.FirstName + ISNULL(' ' + p.MiddleName, '') AS
         NameInFull,
       eph.Rate * 40 AS SalaryInAWeek
FROM HumanResources.EmployeePayHistory AS eph
JOIN Person.Person AS p
  ON eph.BusinessEntityID = p.BusinessEntityID
WHERE eph.RateChangeDate = (
    SELECT MAX(RateChangeDate)
    FROM HumanResources.EmployeePayHistory
    WHERE BusinessEntityID = eph.BusinessEntityID )
ORDER BY NameInFull ASC;

/*25)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 SalesOrderID, ProductID, OrderQty,
       SUM(OrderQty) OVER (PARTITION BY SalesOrderID) AS TotalQuantity,
       AVG(OrderQty) OVER (PARTITION BY SalesOrderID) AS AvgQuantity,
```

```
       COUNT(OrderQty) OVER (PARTITION BY SalesOrderID) AS NoOfOrders,
       MIN(OrderQty) OVER (PARTITION BY SalesOrderID) AS MinQuantity,
       MAX(OrderQty) OVER (PARTITION BY SalesOrderID) AS MaxQuantity
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN (43659, 43664);
```