```sql
use WideWorldImporters;

--1. How many orders are placed on 26th June 2015?
select count (*) as Orders_placed from Sales.Orders
where OrderDate = '2015-06-26';
--118 orders have been placed

--2. What is the total transaction amount done by the client whose CustomerID is 1?
select sum(TransactionAmount) as Total_trans_cus1 from Sales.CustomerTransactions
where CustomerID = 1;
--56,435.84

--3. How many unique secondary postal address lines are there?
select count(distinct PostalAddressLine2) from Sales.Customers;
--452

--4. What is the avg taxAmount given by each customer?
select CustomerID, avg(TaxAmount) as Avg_Tax_Amount from Sales.CustomerTransactions
group by CustomerID
order by CustomerID;

--5. Display all the details of the stockitemID "10", last edited from 1st January 2013 to 31st December 2013.
select * from Sales.InvoiceLines
where StockItemID = 10 and LastEditedWhen between '2013-01-01' and '2014-01-01'
order by LastEditedWhen;

select * from Sales.InvoiceLines
where StockItemID = 10 and year(LastEditedWhen) = 2013
order by LastEditedWhen;

--6. Display the average unit price of all the stockitemID where average line profit exceeds more than 150.
with new as
(
    select StockItemID, avg(UnitPrice) as avg_unit_price, avg(LineProfit) as avg_line_profit from Sales.InvoiceLines
    group by StockItemID
    having avg(LineProfit) > 150
)
select StockItemID, avg_unit_price from new
order by StockItemID;

--7. Provide the complete address to the client by integrating the line 1, line 2 of DeliveryAddress and PostalAddress.
```

```sql
select CustomerID, CustomerName,
DeliveryAddressLine1 + ', ' + DeliveryAddressLine2 as DeliveryAddress,
PostalAddressLine1 + ', ' +PostalAddressLine2 as PostalAddress
from Sales.Customers;

select CustomerID, CustomerName,
concat(DeliveryAddressLine1, DeliveryAddressLine2) as DeliveryAddress,
concat(PostalAddressLine1, ' ', PostalAddressLine2) as PostalAddress
from Sales.Customers;

--8. Fetch all the records for the stock item name starting with USB.
select * from Warehouse.StockItems
where StockItemName like 'USB%';

--9. Calculate the Sales (product of quantity and unit price) sold when picking complete is 1st January 2013.
select OrderID, sum(Quantity * UnitPrice) as Sales
from Sales.OrderLines
where PickingCompletedWhen like '2013-01-01%'
group by OrderID
order by OrderID;

--or total sales

with new as
(
    select OrderID, sum(Quantity * UnitPrice) as Sales
    from Sales.OrderLines
    where PickingCompletedWhen like '2013-01-01%'
    group by OrderID
)
select sum(Sales) as total_sales from new;

--10. Round off the transaction amount to the nearest integer value.
select *, round(TransactionAmount, 0) as rounded_tran_amount
from Sales.CustomerTransactions;

--or
select *, floor(TransactionAmount) from Sales.CustomerTransactions; --It will just show int value without roundig off

--11. Who are the top 10 customers with the highest total purchase amount?
select top 10 c.CustomerID, c.CustomerName, sum(ol.Quantity * ol.UnitPrice) as Purchase_Amount
from Sales.Customers as c
inner join Sales.Orders as o on c.CustomerID = o.CustomerID
inner join Sales.OrderLines as ol on o.OrderID = ol.OrderID
group by c.CustomerID, c.CustomerName
order by Purchase_Amount desc;
```

```
88
89  --12. What is the total transaction amount for each year?
90  select year(TransactionDate) as Tran_Year, sum(TransactionAmount) as
      Tran_Amount
91  from Sales.CustomerTransactions
92  group by year(TransactionDate)
93  order by Tran_Year;
94
95  --13. How many customers have made repeat purchases?
96  with new as
97  (
98      select CustomerID, OrderDate,
99      row_number() over (partition by CustomerID order by OrderDate) as
          Times_Ordered
100     from Sales.Orders
101 )
102 select count(distinct CustomerID) from new
103 where Times_Ordered > 1;
104
105 with new as
106 (
107     select CustomerID, OrderDate,
108     row_number() over (partition by CustomerID order by OrderDate) as
          Times_Ordered
109     from Sales.Orders
110 )
111 select CustomerID, count(Times_Ordered) as times
112 from new
113 group by CustomerID
114 having count(Times_Ordered) > 1
115 order by times desc;
116
117 --14. How many stock items in the warehouse have a quantity on hand below the
      reorder level?
118 select count(*) from Warehouse.StockItemHoldings
119 where QuantityOnHand < ReorderLevel;
120
121 --15. What is the total value of stock items held in the warehouse?
122 select sum(sh.QuantityOnHand * s.UnitPrice) as TotalValue
123 from Warehouse.StockItemHoldings as sh
124 inner join Warehouse.StockItems as s on sh.StockItemID = s.StockItemID;
125 -- 98,07,60,440.97
126
127 --16. What is the average quantity on hand for each stock item name in the
      warehouse?
128 select sh.StockItemID, s.StockItemName, avg(sh.QuantityOnHand) as avg_quantity
129 from Warehouse.StockItemHoldings as sh
130 inner join Warehouse.StockItems as s on sh.StockItemID = s.StockItemID
131 group by sh.StockItemID, s.StockItemName;
```

```
132
133 --17. Which stock item has the highest quantity on hand in the warehouse?
134 select top 1 StockItemID, QuantityOnHand from Warehouse.StockItemHoldings
135 order by QuantityOnHand desc;
136
137 select StockItemID, QuantityOnHand from Warehouse.StockItemHoldings
138 where QuantityOnHand = (select max(QuantityOnHand) from
      Warehouse.StockItemHoldings);
139
140 --18. What is the rank of each customer transaction based on the Amount
      transacted for each Transaction type?
141 select ct.CustomerID, tt.TransactionTypeName, sum(ct.TransactionAmount) as
      TotalTransactionAmount,
142 rank() over (partition by tt.TransactionTypeName order by sum
      (ct.TransactionAmount) desc) as rank_value
143 from Sales.CustomerTransactions as ct
144 inner join Application.TransactionTypes as tt
145 on ct.TransactionTypeID = tt.TransactionTypeID
146 group by ct.CustomerID, tt.TransactionTypeName
147 order by ct.CustomerID;
148
149 --19. Retrieve the total sales revenue for each year and month, as well as the
       grand total for all years and months.
150 select year(o.OrderDate) as OrderYear, sum(ol.Quantity * ol.UnitPrice) as
      TotalSales
151 from Sales.Orders as o
152 inner join Sales.OrderLines as ol
153 on o.OrderID = ol.OrderID
154 group by year(o.OrderDate)
155 order by OrderYear;
156
157 select year(o.OrderDate) as OrderYear, month(o.OrderDate) as OrderMonth, sum
      (ol.Quantity * ol.UnitPrice) as TotalSales
158 from Sales.Orders as o
159 inner join Sales.OrderLines as ol
160 on o.OrderID = ol.OrderID
161 group by year(o.OrderDate), month(o.OrderDate)
162 order by OrderYear, OrderMonth;
163
164 select year(o.OrderDate) as OrderYear, month(o.OrderDate) as OrderMonth, sum
      (Quantity * UnitPrice)  as TotalSales
165 from Sales.Orders as o
166 inner join Sales.OrderLines as ol
167 on o.OrderID = ol.OrderID
168 group by rollup (year(o.OrderDate), month(o.OrderDate));
169
170 --20. Display the details of the orders received by the customers on Each
      Date.
171 select Date_part = convert(date, ConfirmedDeliveryTime), count
```

```
    (ConfirmedReceivedBy) as OrdersReceived
from Sales.Invoices
where convert(date, ConfirmedDeliveryTime) is not null
group by convert(date, ConfirmedDeliveryTime)
order by convert(date, ConfirmedDeliveryTime);

--21. Display the order details (Order ID, Customer name, Order date) for all orders placed in
-- the year 2015 and in the month of May.
select o.OrderID, c.CustomerName, o.OrderDate
from Sales.Orders as o
inner join Sales.Customers as c on o.CustomerID = c.CustomerID
where year(o.OrderDate) = 2015 and month(o.OrderDate) = 5
order by o.OrderDate;

--22. Start a transaction, update the amount of a supplier transaction where SupplierTransactionID is 12345,
-- and rollback the transaction.
begin transaction;
update Purchasing.SupplierTransactions
set TransactionAmount = 0 where SupplierTransactionID = 12345;
rollback transaction;

--23. Retrieve the unique list of product names from the "Sales.OrderLines" and "Purchasing.SupplierTransactions" tables.
select Description from Sales.OrderLines
union
select Description from Purchasing.PurchaseOrderLines
order by Description;

--24. Retrieve the unique customer IDs between the "Sales.Customers" and "Sales.OrderLines" tables.
select CustomerID from Sales.Customers
union
select CustomerID from Sales.Orders
order by CustomerID;

--25. Retrieve the common customer IDs between the "Sales.Customers" and "Sales.OrderLines" tables.
select CustomerID from Sales.Customers
intersect
select CustomerID from Sales.Orders
order by CustomerID;

--26. Retrieve all products that are currently out of stock.
select * from Warehouse.StockItems
where QuantityPerOuter = 0;

--27. Retrieve the total number of orders placed by each customers.
```

```sql
select c.CustomerID, c.CustomerName, count(o.OrderID) as Total_orders
from Sales.Customers as c
inner join Sales.Orders as o
on c.CustomerID = o.CustomerID
group by c.CustomerID, c.CustomerName;

--28. Retrieve the top 10 products with the highest unit price.
select top 10 StockItemID, StockItemName, UnitPrice
from Warehouse.StockItems
order by UnitPrice desc;

--29. Retrieve the order details for a specific order with the given OrderID.
select * from
Sales.OrderLines inner join Warehouse.StockItems
on Sales.OrderLines.StockItemID = Warehouse.StockItems.StockItemID
where Sales.OrderLines.OrderID = 100;

--30. Retrive all products that have the word "blue" in their name.
select * from Warehouse.StockItems
where StockItemName like '%blue%';

--31. Retrieve the total number of orders placed each month.
select month(OrderDate) as OrderMont, count(OrderID) as TotalOrders
from Sales.Orders
group by month(OrderDate)
order by OrderMont;

--32. Retrieve the products with a unit price higher than the average unit price.
select StockItemID, StockItemName, UnitPrice
from Warehouse.StockItems
where UnitPrice > (select avg(UnitPrice) from Warehouse.StockItems)
order by StockItemID;

--33. Retrieve the top 5 products with the highest sales quantity.
select top 5 ol.StockItemID, si.StockItemName, sum(ol.Quantity) as TotalQuantity
from Sales.OrderLines as ol
inner join Warehouse.StockItems as si
on ol.StockItemID = si.StockItemID
group by ol.StockItemID, si.StockItemName
order by TotalQuantity desc;

--34. Retrieve the customers who have not placed any orders.
select c.CustomerID, c.CustomerName, count(OrderID) as Total_Orders
from Sales.Customers as c
left join Sales.Orders as o
on c.CustomerID = o.CustomerID
group by c.CustomerID, c.CustomerName
```

```sql
order by Total_Orders; -- No customer found with 0 orders.

-- or

select * from Sales.Customers
where CustomerID not in
(select distinct CustomerID from Sales.Orders);

--35. Retrieve the customers who have placed orders in the last 10 years.
select distinct Customers.CustomerName, Orders.OrderDate
from Sales.Customers
inner join Sales.Orders on Customers.CustomerID = Orders.CustomerID
where Orders.OrderDate >= dateadd(yyyy,-10,getdate())
order by Orders.OrderDate;

--36. How many customers are general retailers?
select count(cc.CustomerCategoryName)
from Sales.Customers
inner join Sales.CustomerCategories as cc
on Sales.Customers.CustomerCategoryID = cc.CustomerCategoryID
where cc.CustomerCategoryName = 'GeneralRetailer'; -- 0 customers

--37. Find the cities with no information on the population.
select CityName, LatestRecordedPopulation
from Application.Cities
where LatestRecordedPopulation is null;

--38. Calculate the percentage of customers held by each type of customer category in the total customer pool?
with new as
(
    select cc.CustomerCategoryName, count(c.CustomerID) as CountValue,
    TotalCountValue = (select count(c.CustomerID) as CountValue from
      Sales.Customers as c
                    inner join Sales.CustomerCategories as cc on
                  c.CustomerCategoryID = cc.CustomerCategoryID)
    from Sales.Customers as c
    inner join Sales.CustomerCategories as cc
    on c.CustomerCategoryID = cc.CustomerCategoryID
    group by cc.CustomerCategoryName
)
select CustomerCategoryName, round((cast(CountValue as float)/TotalCountValue)*100, 2) as PercentCustomers
from new
order by PercentCustomers desc;

--39. List all the employee details of Wide World Importers who are not permitted to login to the system.
select * from Application.People
```

```
306 where IsPermittedToLogon = 0;
307
308 --40. Find details of the top 5 highest purchasing customer in 2014.
309 with new as
310 (
311     select c.CustomerID, c.CustomerName, o.OrderID, o.OrderDate, year
        (o.OrderDate) as OrderYear,
312          sum(ol.Quantity * ol.UnitPrice) as TotalAmount
313     from Sales.Customers as c
314     inner join Sales.Orders as o on c.CustomerID = o.CustomerID
315     inner join Sales.OrderLines as ol on o.OrderID = ol.OrderID
316     group by c.CustomerID, c.CustomerName, o.OrderID, o.OrderDate
317 )
318 select top 5 CustomerID, CustomerName, count(OrderID) as TotalOrders, sum
    (TotalAmount) as TotalAmount
319 from new
320 where OrderYear = 2014
321 group by CustomerID, CustomerName
322 order by TotalAmount desc; -- Based on highest amount spent
323
324 --41. Find the details of the top 5 sales persons with highest orders
    initiated.
325 with new as
326 (
327     select o.SalespersonPersonID, o.OrderID, sum(ol.Quantity * ol.UnitPrice)
        as TotalAmount
328     from Sales.Orders as o
329     inner join Sales.OrderLines as ol
330     on o.OrderID = ol.OrderID
331     group by o.SalespersonPersonID, o.OrderID
332 )
333 select top 5 SalespersonPersonID, count(OrderID) as TotalOrders, sum
    (TotalAmount) as TotalAmount
334 from new
335 group by SalespersonPersonID
336 order by TotalOrders desc;
337
338 --42. List the top 5 highest purchasing cities.
339 select top 1 * from Application.Cities;
340 select top 1 * from Sales.OrderLines; -- No matching columms
341
342 --43. Which is the largest order ever placed? Display its details.
343 select top 1 *, (ol.Quantity * ol.UnitPrice) as TotalPrice
344 from Sales.OrderLines as ol
345 order by TotalPrice desc;
346
347 --44. Find the total revenue produced by each supermarket customer in 2013.
348 select c.CustomerID, c.CustomerName, sum(ol.Quantity * ol.UnitPrice) as
    TotalPrice
```

```sql
from Sales.Customers as c
inner join Sales.Orders as o on c.CustomerID = o.CustomerID
inner join Sales.OrderLines as ol on o.OrderID = ol.OrderID
where year(o.OrderDate) = 2013
group by c.CustomerID, c.CustomerName
order by c.CustomerID;

--45. Retrieve the order details for orders placed in the year 2016.
select * from
Sales.Orders as o
inner join Sales.OrderLines as ol on o.OrderID = ol.OrderID
inner join Warehouse.StockItems as si on ol.StockItemID = si.StockItemID
where year(o.OrderDate) = 2016
order by o.OrderDate, o.OrderID;

--46. Retrieve the top 10 products with the highest profit margin.
select top 10 StockItemID, StockItemName, UnitPrice, RecommendedRetailPrice,
  (RecommendedRetailPrice - UnitPrice) as Profit
from Warehouse.StockItems
order by Profit desc;

--47. Retrieve the average unit price for each supplier
select s.SupplierID, s.SupplierName, avg(si.UnitPrice) as AvgPrice
from Purchasing.Suppliers as s
inner join Warehouse.StockItems as si on s.SupplierID = si.SupplierID
group by s.SupplierID, s.SupplierName
order by s.SupplierID;

--48. Calculate the total revenue generated by each product category.
select sg.StockGroupName, sum(ol.Quantity * ol.UnitPrice) as TotalRevenue
from Sales.OrderLines as ol
inner join Warehouse.StockItemStockGroups as sisg on ol.StockItemID =
  sisg.StockItemID
inner join Warehouse.StockGroups as sg on sisg.StockItemStockGroupID =
  sg.StockGroupID
group by sg.StockGroupName
order by sg.StockGroupName;

--49. Retrieve the names of all products with a unit price greater than $50.
select distinct StockItemID, StockItemName
from Warehouse.StockItems
where UnitPrice > 50;

--50. Find the total quantity sold for each product.
select si.StockItemID, si.StockItemName, sum(ol.Quantity) as TotalQuantity
from Sales.OrderLines as ol
inner join Warehouse.StockItems as si on ol.StockItemID = si.StockItemID
group by si.StockItemID, si.StockItemName
order by si.StockItemID;
```

```sql

--51. Calculate the average duration of delivery spent on Every Delivery Method to deliver to customer.
with new as
(
    select dm.DeliveryMethodID, dm.DeliveryMethodName,
    avg(datediff(minute, o.PickingCompletedWhen, i.ConfirmedDeliveryTime)) as AvgMinutes
    from Sales.Orders as o
    inner join sales.Invoices as i on o.OrderID = i.OrderID
    inner join Application.DeliveryMethods as dm on i.DeliveryMethodID = dm.DeliveryMethodID
    group by dm.DeliveryMethodID, dm.DeliveryMethodName
)
select DeliveryMethodID, DeliveryMethodName,
(AvgMinutes/60) as Hours,
(AvgMinutes%60) as Minutes
from new;

--52. Retrieve the names of customers who have placed more than 100 orders.
select c.CustomerID, c.CustomerName, count(o.OrderID) as OrdersPlaced
from Sales.Customers as c
inner join Sales.Orders as o on c.CustomerID = o.CustomerID
group by c.CustomerID, c.CustomerName
having count(o.OrderID) > 100
order by OrdersPlaced;

--53. Display the names of customers who have placed at least one order in each month of 2013.
with new as
(
    select c.CustomerID, c.CustomerName, month(o.OrderDate) as OrderMonth, count(o.OrderID) as Orders,
    row_number() over (partition by c.CustomerID order by month(o.OrderDate)) as RowNumber
    from Sales.Customers as c
    inner join Sales.Orders as o on c.CustomerID = o.CustomerID
    group by c.CustomerID, c.CustomerName, month(o.OrderDate)
)
select CustomerID, CustomerName
from new
where RowNumber = 12;

--54. Calculate the total revenue for each year
select year(o.OrderDate) as OrderYear, sum(ol.Quantity * ol.UnitPrice) as TotalRevenue
from Sales.OrderLines as ol
inner join Sales.Orders as o on ol.OrderID = o.OrderID
group by year(o.OrderDate)
```

```
437 order by OrderYear;
438
439 --55. Retrieve the names of customers who have placed orders for products with
       a unit price between $50 and $100.
440 select distinct c.CustomerID, c.CustomerName
441 from Sales.Customers as c
442 inner join Sales.Orders as o on c.CustomerID = o.CustomerID
443 inner join Sales.OrderLines as ol on o.OrderID = ol.OrderID
444 inner join Warehouse.StockItems as si on ol.StockItemID = si.StockItemID
445 where si.UnitPrice between 50 and 100
446 order by c.CustomerID;
447
448 --56. What are the top 5 best-selling products in terms of revenue?
449 select top 5 ol.StockItemID, si.StockItemName, sum(ol.Quantity * ol.UnitPrice)
       as TotalRevenue
450 from Sales.OrderLines as ol
451 inner join Warehouse.StockItems as si on ol.StockItemID = si.StockItemID
452 group by ol.StockItemID, si.StockItemName
453 order by TotalRevenue desc;
454
455 --57. How many new customers were acquired each month?
456 SELECT YEAR(O.OrderDate) AS Year, MONTH(O.OrderDate) AS Month, COUNT(DISTINCT
      C.CustomerID) AS
457 NewCustomers
458 FROM Sales.Orders O
459 JOIN Sales.Customers C ON O.CustomerID = C.CustomerID
460 GROUP BY YEAR(O.OrderDate), MONTH(O.OrderDate);
461
462 --58. How many orders have been placed in each quarter of the year?
463 select year(OrderDate) as OrderYear, datepart(quarter, OrderDate) as
      OrderQuarter, count(OrderID) as TotalOrders
464 from Sales.Orders
465 group by year(OrderDate), datepart(quarter, OrderDate)
466 order by year(OrderDate), datepart(quarter, OrderDate);
467
468 --59. How many new customers were acquired each year?
469 SELECT YEAR(O.OrderDate) AS Year, COUNT(DISTINCT C.CustomerID) AS NewCustomers
470 FROM Sales.Orders O
471 JOIN Sales.Customers C ON O.CustomerID = C.CustomerID
472 GROUP BY YEAR(O.OrderDate);
473
474 --60. Retrieve the average quantity sold per month for each product.
475 with new as
476 (
477     select si.StockItemID, si.StockItemName,
478     year(o.OrderDate) as OrderYear, month(o.OrderDate) as OrderMonth, sum
        (ol.Quantity) as TotalQuantity
479     from Warehouse.StockItems as si
480     inner join Sales.OrderLines as ol on si.StockItemID = ol.StockItemID
```

```
481      inner join Sales.Orders as o on ol.OrderID = o.OrderID
482      group by si.StockItemID, si.StockItemName, year(o.OrderDate), month
           (o.OrderDate)
483  )
484  select StockItemID, StockItemName, avg(TotalQuantity) as AvgQuantity
485  from new
486  group by StockItemID, StockItemName
487  order by StockItemID, StockItemName;
```