Group 1 Paul Pluszczewicz

Contains detailed information of all 30 queries.

Includes all charts, explanations, and database execution samples.

Contents

[Diagrams of AdventureWorks2014 10](#_Toc528357940)

[Show the diagrams in standard and key view 10](#_Toc528357941)

[Standard View of AdventureWorks2014. 11](#_Toc528357942)

[Diagrams of AdventureWorksDW2014 12](#_Toc528357943)

[Show the diagrams in standard and key view 12](#_Toc528357944)

[Standard View of AdventureWorksDW2014. 13](#_Toc528357945)

[Diagrams of TSQLV4 14](#_Toc528357946)

[Show the diagrams in standard and key view 14](#_Toc528357947)

[Standard View of TSQLV4. 15](#_Toc528357948)

[Diagrams of WideWorldImporters 16](#_Toc528357949)

[Show the diagrams in standard and key view 16](#_Toc528357950)

[Standard View of WideWorldImporters. 17](#_Toc528357951)

[Diagrams of WideWorldImportersDW 17](#_Toc528357952)

[Show the diagrams in standard and key view 17](#_Toc528357953)

[Standard View of WideWorldImportersDW. 19](#_Toc528357954)

[Problem 01: List all workers in department 7 using AdventureWorks2014. 19](#_Toc528357955)

[What we need to do to solve this problem: 20](#_Toc528357956)

[Use AdventureWorks2014. 20](#_Toc528357957)

[Diagram(s) of tables 20](#_Toc528357958)

[Columns from Standard view 21](#_Toc528357959)

[Project following columns from their respective tables in the select clause 21](#_Toc528357960)

[Order by 21](#_Toc528357961)

[Query to solve this problem: 21](#_Toc528357962)

[Sample Output with (180) rows 22](#_Toc528357963)

[Problem 02: Find all Salespeople that made no sales last year using AdventureWorks2014. 22](#_Toc528357964)

[What we need to solve this problem: 22](#_Toc528357965)

[Use AdventureWorks2014. 23](#_Toc528357966)

[Diagram(s) of tables 23](#_Toc528357967)

[Columns from Standard view 24](#_Toc528357968)

[Project following columns from their respective tables in the select clause 24](#_Toc528357969)

[Query to solve this problem: 25](#_Toc528357970)

[Sample Output with (1) row 25](#_Toc528357971)

[Problem 03: Find all employees that work in sales using AdventureWorks2014. 26](#_Toc528357972)

[What we need to solve this problem: 26](#_Toc528357973)

[Use AdventureWorks2014. 26](#_Toc528357974)

[Diagram(s) of tables 26](#_Toc528357975)

[Columns from Standard view 27](#_Toc528357976)

[Project following columns from their respective tables in the select clause 27](#_Toc528357977)

[Order by 27](#_Toc528357978)

[Query to solve this problem: 28](#_Toc528357979)

[Sample Output with (17) rows 28](#_Toc528357980)

[Problem 04: Show all the employees and how frequently they are paid using AdventureWorks2014. 29](#_Toc528357981)

[What we need to solve this problem: 29](#_Toc528357982)

[Use AdventureWorks2014. 29](#_Toc528357983)

[Diagram(s) of tables 29](#_Toc528357984)

[29](#_Toc528357985)

[Columns from Standard view 30](#_Toc528357986)

[Project following columns from their respective tables in the select clause 30](#_Toc528357987)

[Query to solve this problem: 31](#_Toc528357988)

[Sample Output with (290) rows 31](#_Toc528357989)

[Problem 05: Show all salaried employees using AdventureWorks2014. 32](#_Toc528357990)

[What we need to solve this problem: 32](#_Toc528357991)

[Use AdventureWorks2014. 32](#_Toc528357992)

[Diagram(s) of tables 32](#_Toc528357993)

[32](#_Toc528357994)

[Columns from Standard view 33](#_Toc528357995)

[Project following columns from their respective tables in the select clause 33](#_Toc528357996)

[Query to solve this problem: 34](#_Toc528357997)

[Sample Output with (56) rows 34](#_Toc528357998)

[Problem 06: Find by CustomerID, the number of distinct items and total quantities of all items ever ordered by customer using AdventureWorks2014. 35](#_Toc528357999)

[What we need to solve this problem: 35](#_Toc528358000)

[Use AdventureWorks2014. 35](#_Toc528358001)

[Diagram(s) of tables 35](#_Toc528358002)

[35](#_Toc528358003)

[Columns from Standard view 36](#_Toc528358004)

[Project following columns from their respective tables in the select clause 36](#_Toc528358005)

[Query to solve this problem: 36](#_Toc528358006)

[Sample Output with (19119) rows 37](#_Toc528358007)

[Problem 07: Find by CustomerID, the number of all orders ever made by customer using AdventureWorks2014. 38](#_Toc528358008)

[What we need to solve this problem: 38](#_Toc528358009)

[Use AdventureWorks2014. 38](#_Toc528358010)

[Diagram(s) of tables 38](#_Toc528358011)

[38](#_Toc528358012)

[Columns from Standard view 39](#_Toc528358013)

[Project following columns from their respective tables in the select clause 39](#_Toc528358014)

[Order By 39](#_Toc528358015)

[Query to solve this problem: 40](#_Toc528358016)

[Sample Output with (19119) rows 40](#_Toc528358017)

[Problem 08: Find by CustomerID, all online orders and store orders a customer has ever made using AdventureWorks2014. 41](#_Toc528358018)

[What we need to solve this problem: 41](#_Toc528358019)

[Use AdventureWorks2014. 41](#_Toc528358020)

[Diagram(s) of tables 41](#_Toc528358021)

[Columns from Standard view 42](#_Toc528358022)

[Project following columns from their respective tables in the select clause 42](#_Toc528358023)

[Order By 42](#_Toc528358024)

[Query to solve this problem: 43](#_Toc528358025)

[Sample Output with (19119) rows 43](#_Toc528358026)

[Problem 09: Find all customers who have ever recieved a discount on an ordered item using AdventureWorks2014. 44](#_Toc528358027)

[What we need to solve this problem: 44](#_Toc528358028)

[Use AdventureWorks2014. 44](#_Toc528358029)

[Diagram(s) of tables 44](#_Toc528358030)

[44](#_Toc528358031)

[Columns from Standard view 45](#_Toc528358032)

[Project following columns from their respective tables in the select clause 45](#_Toc528358033)

[Order By 45](#_Toc528358034)

[Query to solve this problem: 46](#_Toc528358035)

[Sample Output with (353) rows 46](#_Toc528358036)

[Problem 10: Find all customers who have made an order before 2012 using AdventureWorks2014. 47](#_Toc528358037)

[What we need to solve this problem: 47](#_Toc528358038)

[Use AdventureWorks2014. 47](#_Toc528358039)

[Diagram(s) of tables 47](#_Toc528358040)

[Columns from Standard view 48](#_Toc528358041)

[Project following columns from their respective tables in the select clause 48](#_Toc528358042)

[Order By 48](#_Toc528358043)

[Query to solve this problem: 49](#_Toc528358044)

[Sample Output with (1406) rows 49](#_Toc528358045)

[Problem 11: Find total number workers in day, evening, and night shifts using AdventureWorks2014. 50](#_Toc528358046)

[What we need to solve this problem: 50](#_Toc528358047)

[Use AdventureWorks2014. 50](#_Toc528358048)

[Diagram(s) of tables 50](#_Toc528358049)

[Columns from Standard view 51](#_Toc528358050)

[Project following columns from their respective tables in the select clause 51](#_Toc528358051)

[Query to solve this problem: 52](#_Toc528358052)

[Sample Output with (3) rows 52](#_Toc528358053)

[Problem 12: List all workers in department 7 using AdventureWorks2014. 53](#_Toc528358054)

[What we need to solve this problem: 53](#_Toc528358055)

[Use AdventureWorks2014. 53](#_Toc528358056)

[Diagram(s) of tables 53](#_Toc528358057)

[Columns from Standard view 54](#_Toc528358058)

[Project following columns from their respective tables in the select clause 54](#_Toc528358059)

[Order By 54](#_Toc528358060)

[Query to solve this problem: 55](#_Toc528358061)

[Sample Output with (180) rows 55](#_Toc528358062)

[Problem 13: Find number of distinct currency rate ids per customer using AdventureWorks2014. 56](#_Toc528358063)

[What we need to solve this problem: 56](#_Toc528358064)

[Use AdventureWorks2014. 56](#_Toc528358065)

[Diagram(s) of tables 56](#_Toc528358066)

[Columns from Standard view 57](#_Toc528358067)

[Project following columns from their respective tables in the select clause 57](#_Toc528358068)

[Order By 57](#_Toc528358069)

[Query to solve this problem: 58](#_Toc528358070)

[Sample Output with (180) rows 58](#_Toc528358071)

[Problem 14: Show all the distinct departments employees belong to within an organization level in AdventureWorks2014. 59](#_Toc528358072)

[What we need to solve this problem: 59](#_Toc528358073)

[Use AdventureWorks2014. 59](#_Toc528358074)

[Diagram(s) of tables 59](#_Toc528358075)

[Columns from Standard view 60](#_Toc528358076)

[Project following columns from their respective tables in the select clause 60](#_Toc528358077)

[Query to solve this problem: 61](#_Toc528358078)

[Sample Output with (5) rows 61](#_Toc528358079)

[Problem 15: Show top 50 customers by Total Spent ever using AdventureWorks2014. 62](#_Toc528358080)

[What we need to solve this problem: 62](#_Toc528358081)

[Use AdventureWorks2014. 62](#_Toc528358082)

[Diagram(s) of tables 62](#_Toc528358083)

[62](#_Toc528358084)

[Columns from Standard view 63](#_Toc528358085)

[Project following columns from their respective tables in the select clause 63](#_Toc528358086)

[Order By 63](#_Toc528358087)

[Query to solve this problem: 64](#_Toc528358088)

[Sample Output with (50) rows 64](#_Toc528358089)

[Problem 16: Show list of Top 100 Best Selling Products in 2013 using AdventureWorks2014. 65](#_Toc528358090)

[What we need to solve this problem: 65](#_Toc528358091)

[Use AdventureWorks2014. 65](#_Toc528358092)

[Diagram(s) of tables 65](#_Toc528358093)

[65](#_Toc528358094)

[Columns from Standard view 66](#_Toc528358095)

[Project following columns from their respective tables in the select clause 66](#_Toc528358096)

[Order By 66](#_Toc528358097)

[Query to solve this problem: 67](#_Toc528358098)

[Sample Output with (50) rows 67](#_Toc528358099)

[Problem 17: Show list of Employees born before 1985 AdventureWorks2014. 68](#_Toc528358100)

[What we need to solve this problem: 68](#_Toc528358101)

[Use AdventureWorks2014. 68](#_Toc528358102)

[Diagram(s) of tables 68](#_Toc528358103)

[Columns from Standard view 69](#_Toc528358104)

[Project following columns from their respective tables in the select clause 69](#_Toc528358105)

[Order By 69](#_Toc528358106)

[Query to solve this problem: 70](#_Toc528358107)

[Sample Output with (201) rows 70](#_Toc528358108)

[Problem 18: Show list of total sick leave hours per Department using AdventureWorks2014. 71](#_Toc528358109)

[What we need to solve this problem: 71](#_Toc528358110)

[Use AdventureWorks2014. 71](#_Toc528358111)

[Diagram(s) of tables 71](#_Toc528358112)

[Columns from Standard view 72](#_Toc528358113)

[Project following columns from their respective tables in the select clause 72](#_Toc528358114)

[Order By 72](#_Toc528358115)

[Query to solve this problem: 73](#_Toc528358116)

[Sample Output with (16) rows 73](#_Toc528358117)

[Problem 19: Show list of stores with most distinct products sold using AdventureWorks2014. 74](#_Toc528358118)

[What we need to solve this problem: 74](#_Toc528358119)

[Use AdventureWorks2014. 74](#_Toc528358120)

[Diagram(s) of tables 74](#_Toc528358121)

[74](#_Toc528358122)

[Columns from Standard view 75](#_Toc528358123)

[Project following columns from their respective tables in the select clause 75](#_Toc528358124)

[Order By 75](#_Toc528358125)

[Query to solve this problem: 76](#_Toc528358126)

[Sample Output with (635) rows 76](#_Toc528358127)

[Problem 20: Show total amount of money made by a store using AdventureWorks2014. 77](#_Toc528358128)

[What we need to solve this problem: 77](#_Toc528358129)

[Use AdventureWorks2014. 77](#_Toc528358130)

[Diagram(s) of tables 77](#_Toc528358131)

[77](#_Toc528358132)

[Columns from Standard view 78](#_Toc528358133)

[Project following columns from their respective tables in the select clause 78](#_Toc528358134)

[Order By 78](#_Toc528358135)

[Query to solve this problem: 79](#_Toc528358136)

[Sample Output with (635) rows 79](#_Toc528358137)

[Problem 21: Show total employees in and not in management using AdventureWorks2014. 80](#_Toc528358138)

[What we need to solve this problem: 80](#_Toc528358139)

[Use AdventureWorks2014. 80](#_Toc528358140)

[Diagram(s) of tables 80](#_Toc528358141)

[Columns from Standard view 81](#_Toc528358142)

[Project following columns from their respective tables in the select clause 81](#_Toc528358143)

[Query to solve this problem: 81](#_Toc528358144)

[Sample Output with (2) rows 82](#_Toc528358145)

[Problem 22: Show all customers that had a discount on a purchase using AdventureWorks2014. 83](#_Toc528358146)

[What we need to solve this problem: 83](#_Toc528358147)

[Use AdventureWorks2014. 83](#_Toc528358148)

[Diagram(s) of tables 83](#_Toc528358149)

[84](#_Toc528358150)

[Columns from Standard view 85](#_Toc528358151)

[Project following columns from their respective tables in the select clause 85](#_Toc528358152)

[Query to solve this problem: 85](#_Toc528358153)

[Sample Output with (353) rows 86](#_Toc528358154)

[Problem 23: Determine how many expensive orders a customer has made using AdventureWorks2014. 87](#_Toc528358155)

[What we need to solve this problem: 87](#_Toc528358156)

[Use AdventureWorks2014. 87](#_Toc528358157)

[Diagram(s) of tables 87](#_Toc528358158)

[Columns from Standard view 88](#_Toc528358159)

[Project following columns from their respective tables in the select clause 88](#_Toc528358160)

[Query to solve this problem: 88](#_Toc528358161)

[Sample Output with (353) rows 89](#_Toc528358162)

[Problem 24: Determine how many customer orders were picked up using AdventureWorks2014. 90](#_Toc528358163)

[What we need to solve this problem: 90](#_Toc528358164)

[Use AdventureWorks2014. 90](#_Toc528358165)

[Diagram(s) of tables 90](#_Toc528358166)

[Columns from Standard view 91](#_Toc528358167)

[Project following columns from their respective tables in the select clause 91](#_Toc528358168)

[Query to solve this problem: 91](#_Toc528358169)

[Sample Output with (19119) rows 92](#_Toc528358170)

[Problem 25: Determine how many customer orders had a good discount using TSQLV4. 93](#_Toc528358171)

[What we need to solve this problem: 93](#_Toc528358172)

[Use TSQLV4. 93](#_Toc528358173)

[Diagram(s) of tables 93](#_Toc528358174)

[Columns from Standard view 94](#_Toc528358175)

[Project following columns from their respective tables in the select clause 94](#_Toc528358176)

[Order By 94](#_Toc528358177)

[Query to solve this problem: 94](#_Toc528358178)

[Sample Output with (89) rows 95](#_Toc528358179)

[Problem 26: Determine how many customer orders had an expensive item using TSQLV4. 96](#_Toc528358180)

[What we need to solve this problem: 96](#_Toc528358181)

[Use TSQLV4. 96](#_Toc528358182)

[Diagram(s) of tables 96](#_Toc528358183)

[Columns from Standard view 97](#_Toc528358184)

[Project following columns from their respective tables in the select clause 97](#_Toc528358185)

[Order By 97](#_Toc528358186)

[Query to solve this problem: 97](#_Toc528358187)

[Sample Output with (89) rows 98](#_Toc528358188)

[Problem 27: Determine how many products need to be chilled using WideWorldImporters. 99](#_Toc528358189)

[What we need to solve this problem: 99](#_Toc528358190)

[Use WideWorldImporters. 99](#_Toc528358191)

[Diagram(s) of tables 99](#_Toc528358192)

[Columns from Standard view 100](#_Toc528358193)

[Project following columns from their respective tables in the select clause 100](#_Toc528358194)

[Order By 100](#_Toc528358195)

[Query to solve this problem: 100](#_Toc528358196)

[Sample Output with (991) rows 101](#_Toc528358197)

[Problem 28: Determine how many products are meant to be surprises using WideWorldImporters. 102](#_Toc528358198)

[What we need to solve this problem: 102](#_Toc528358199)

[Use WideWorldImporters. 102](#_Toc528358200)

[Diagram(s) of tables 102](#_Toc528358201)

[Columns from Standard view 103](#_Toc528358202)

[Project following columns from their respective tables in the select clause 103](#_Toc528358203)

[Order By 103](#_Toc528358204)

[Query to solve this problem: 103](#_Toc528358205)

[Sample Output with (663) rows 104](#_Toc528358206)

[Problem 29: Determine how many customer orders use shipper three using TSQLV4. 105](#_Toc528358207)

[What we need to solve this problem: 105](#_Toc528358208)

[Use TSQLV4. 105](#_Toc528358209)

[Diagram(s) of tables 105](#_Toc528358210)

[Columns from Standard view 106](#_Toc528358211)

[Project following columns from their respective tables in the select clause 106](#_Toc528358212)

[Order By 106](#_Toc528358213)

[Query to solve this problem: 106](#_Toc528358214)

[Sample Output with (78) rows 107](#_Toc528358215)

[Problem 30: Determine how many products were packed by person 14 using WideWorldImporters. 108](#_Toc528358216)

[What we need to solve this problem: 108](#_Toc528358217)

[Use WideWorldImporters. 108](#_Toc528358218)

[Diagram(s) of tables 108](#_Toc528358219)

[Columns from Standard view 109](#_Toc528358220)

[Project following columns from their respective tables in the select clause 109](#_Toc528358221)

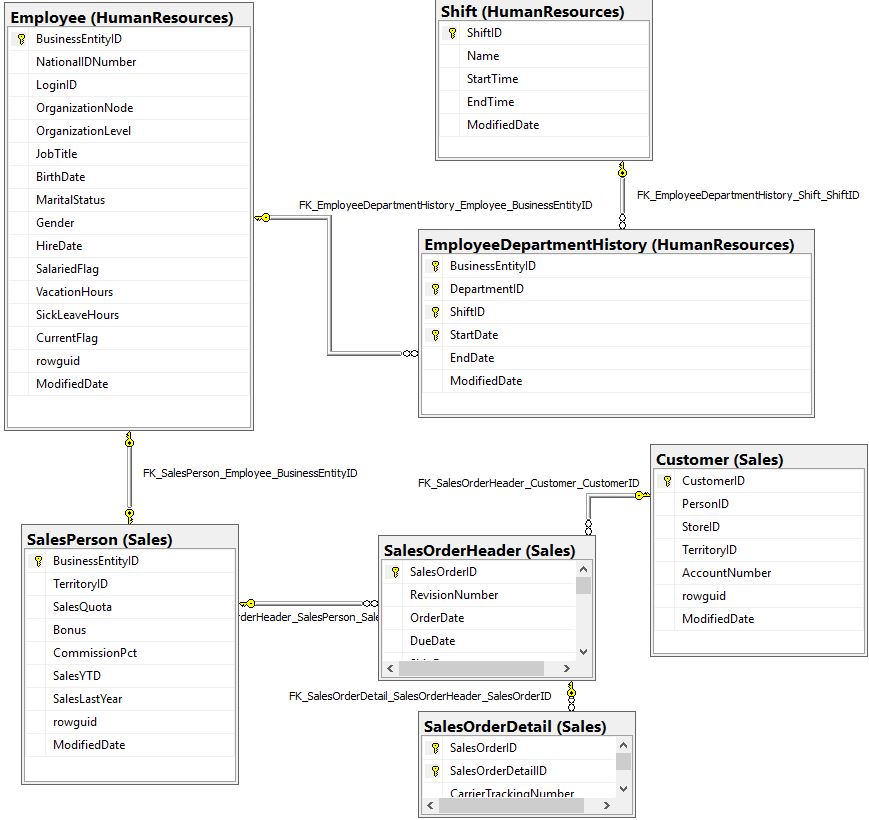
[Order By 109](#_Toc528358222)

[Query to solve this problem: 109](#_Toc528358223)

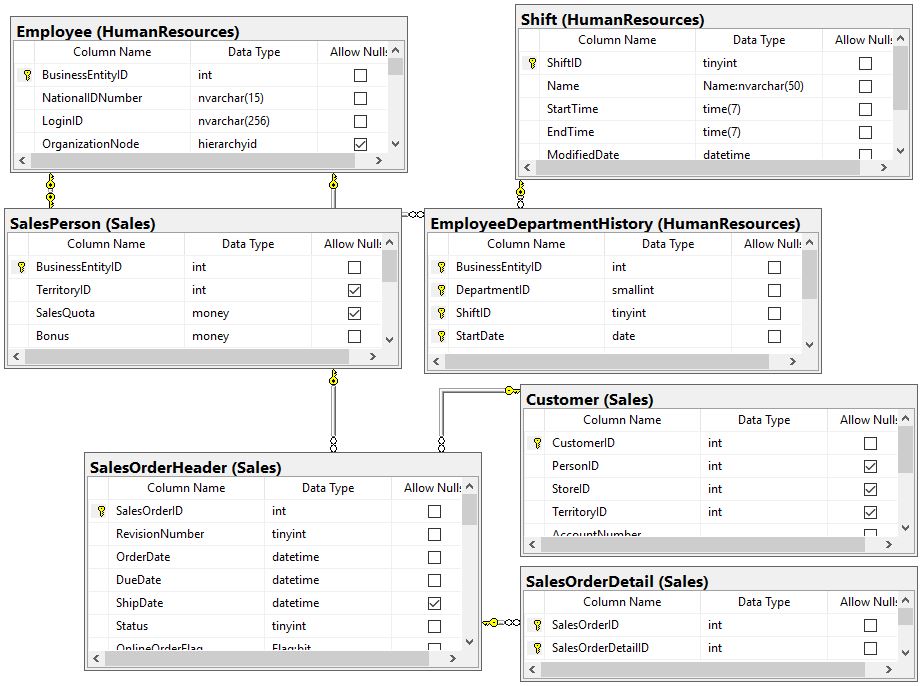
[Sample Output with (70510) rows 110](#_Toc528358224)

# Diagrams of AdventureWorks2014

## Show the diagrams in standard and key view

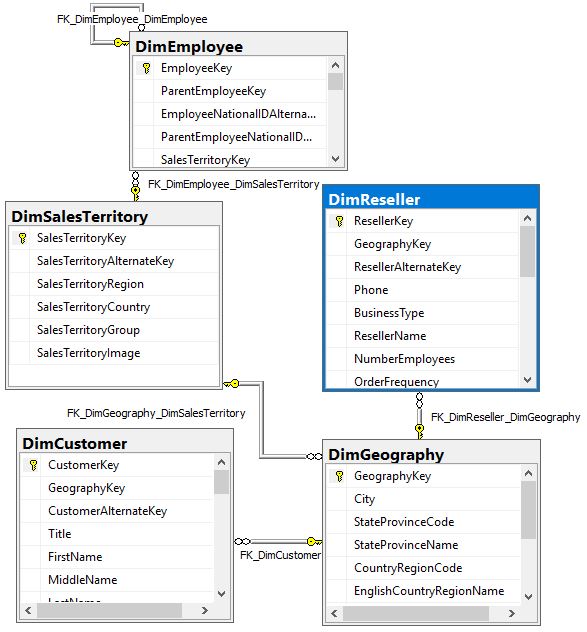


## Standard View of AdventureWorks2014.

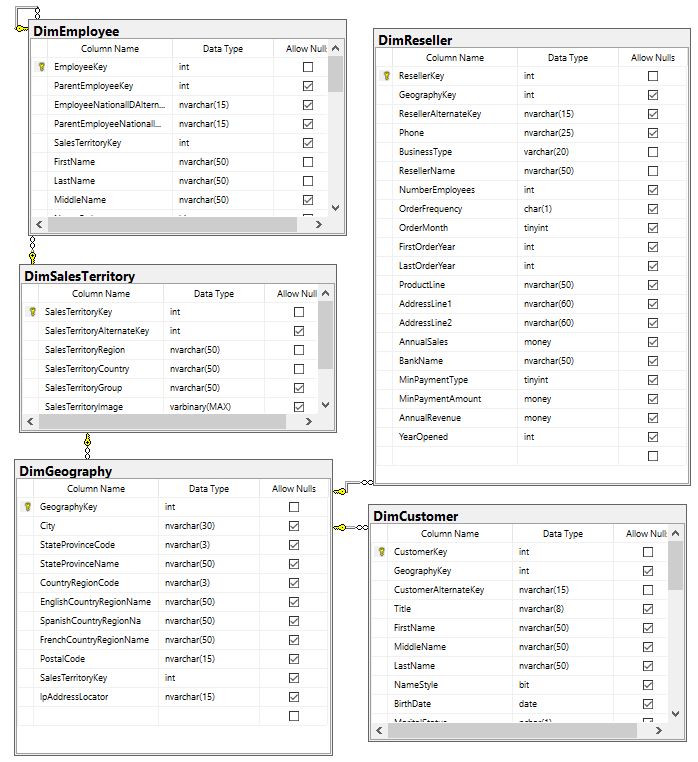


# Diagrams of AdventureWorksDW2014

## Show the diagrams in standard and key view

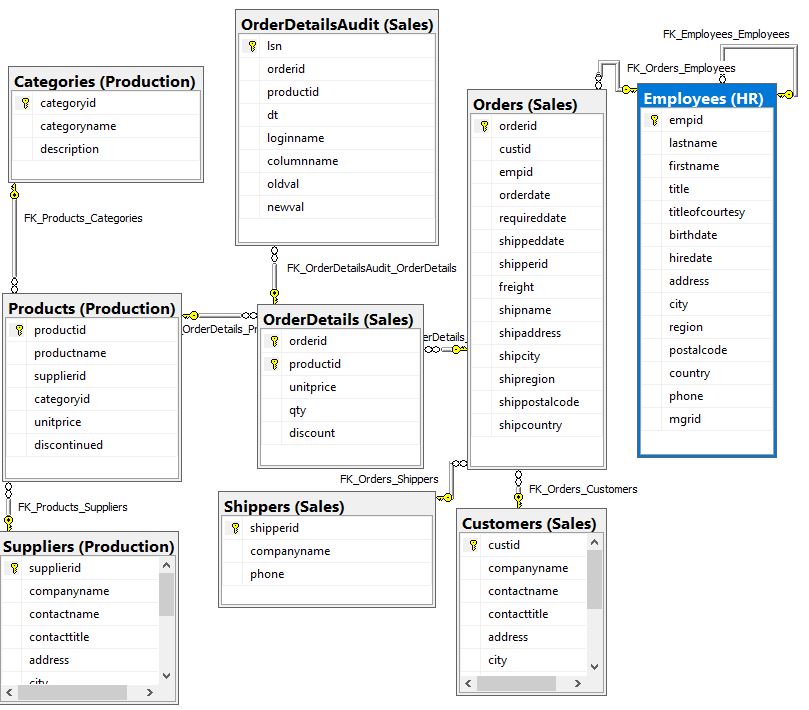


## Standard View of AdventureWorksDW2014.

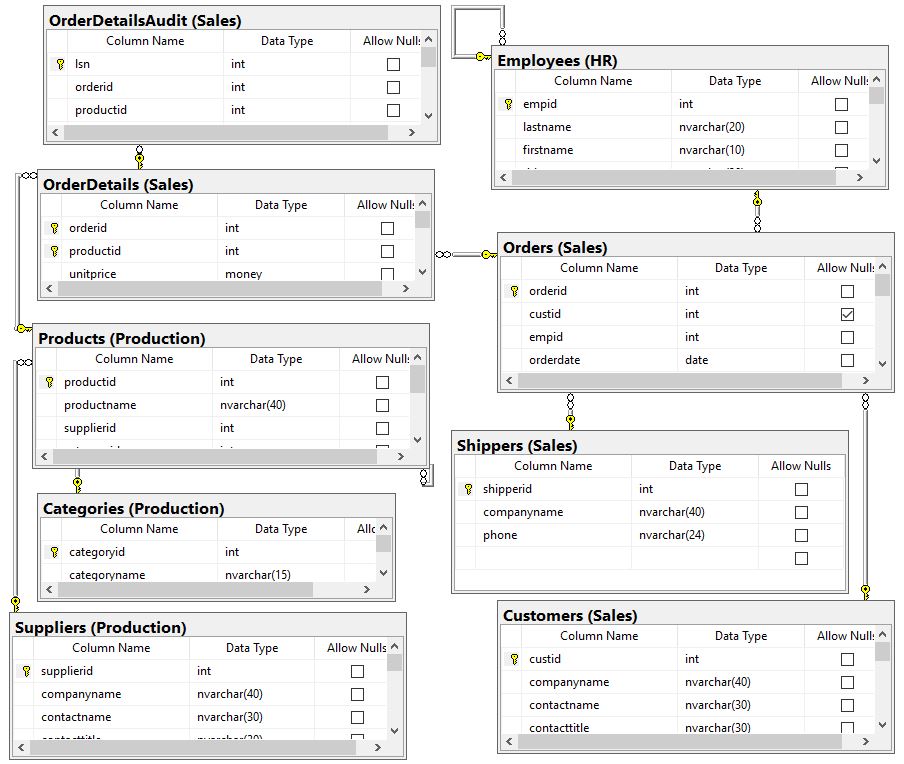


# Diagrams of TSQLV4

## Show the diagrams in standard and key view

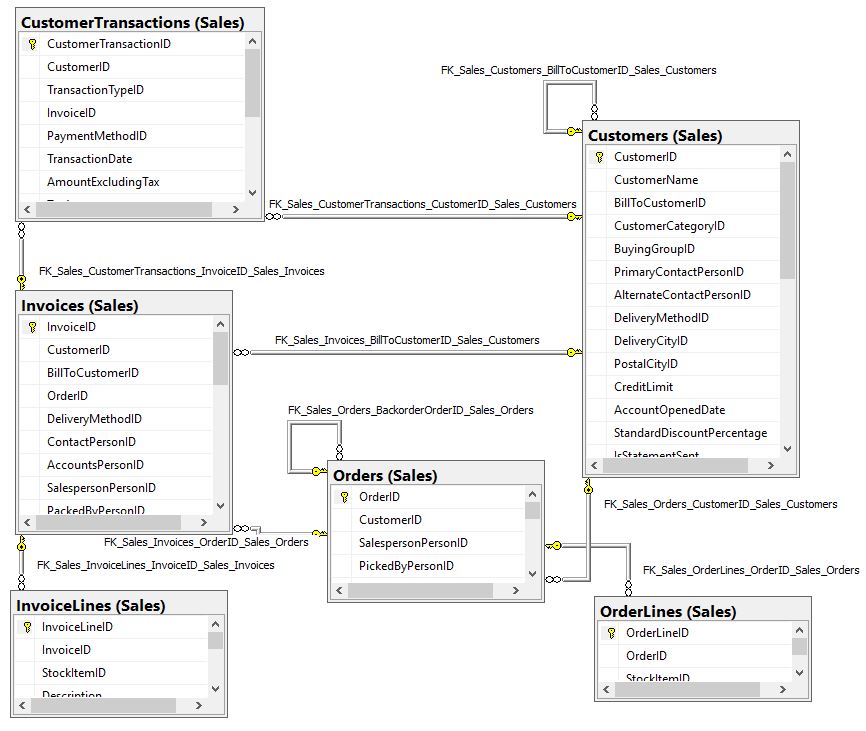


## Standard View of TSQLV4.

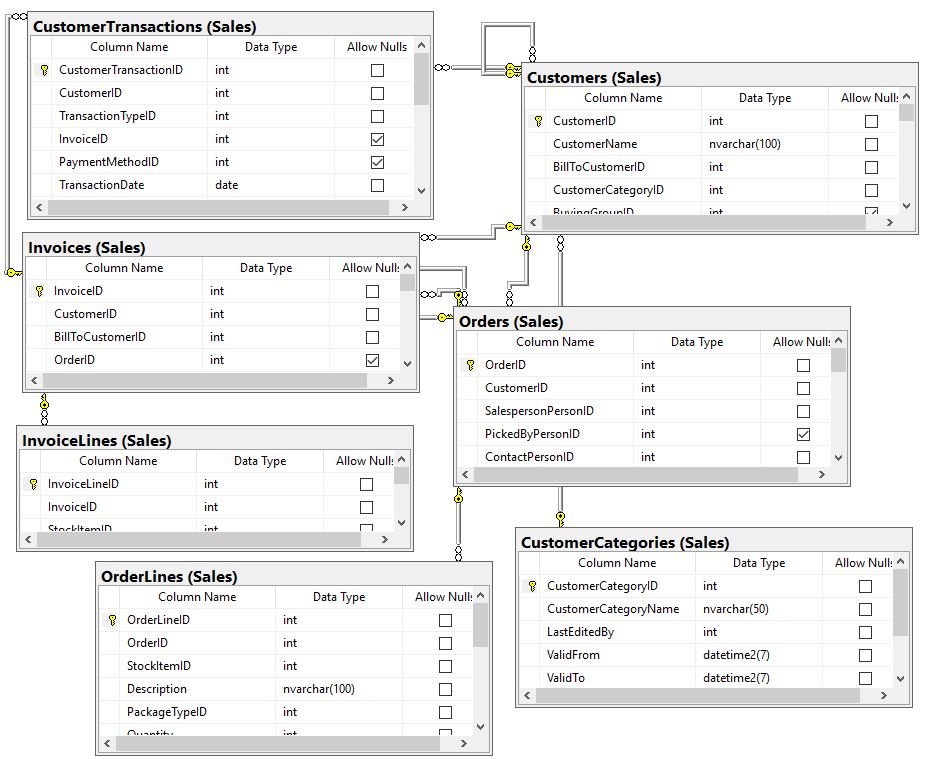


# Diagrams of WideWorldImporters

## Show the diagrams in standard and key view



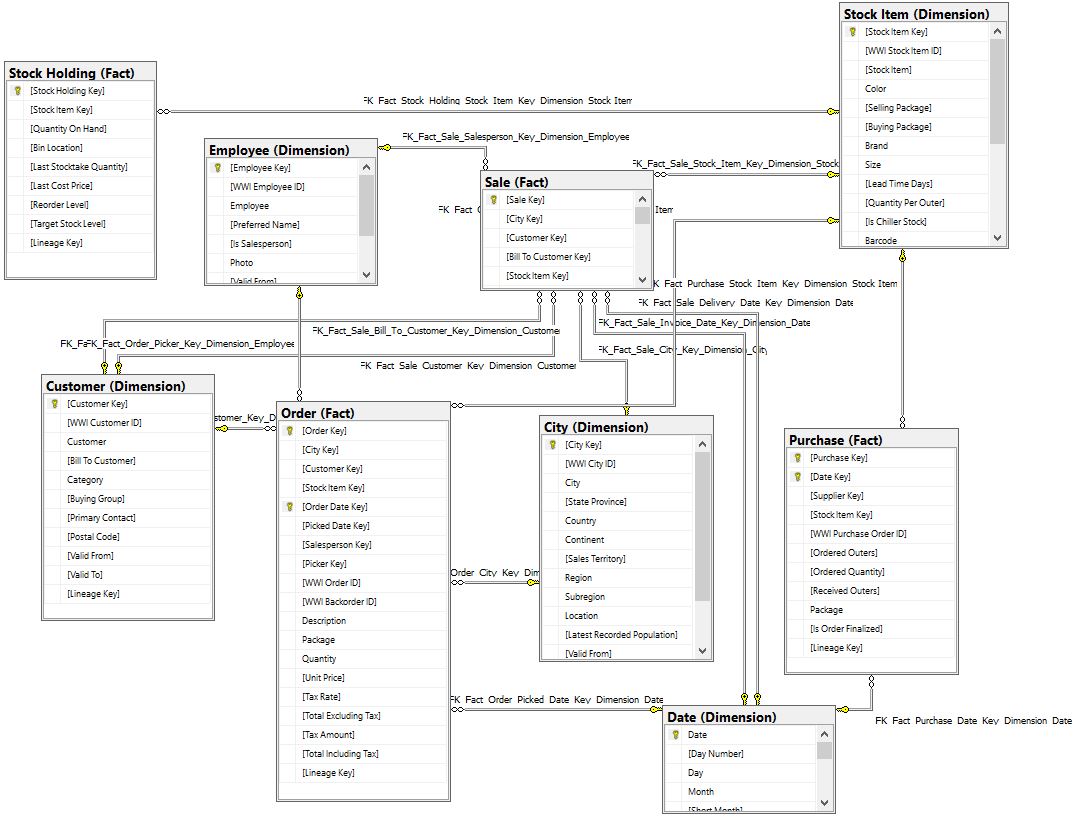
## Standard View of WideWorldImporters.



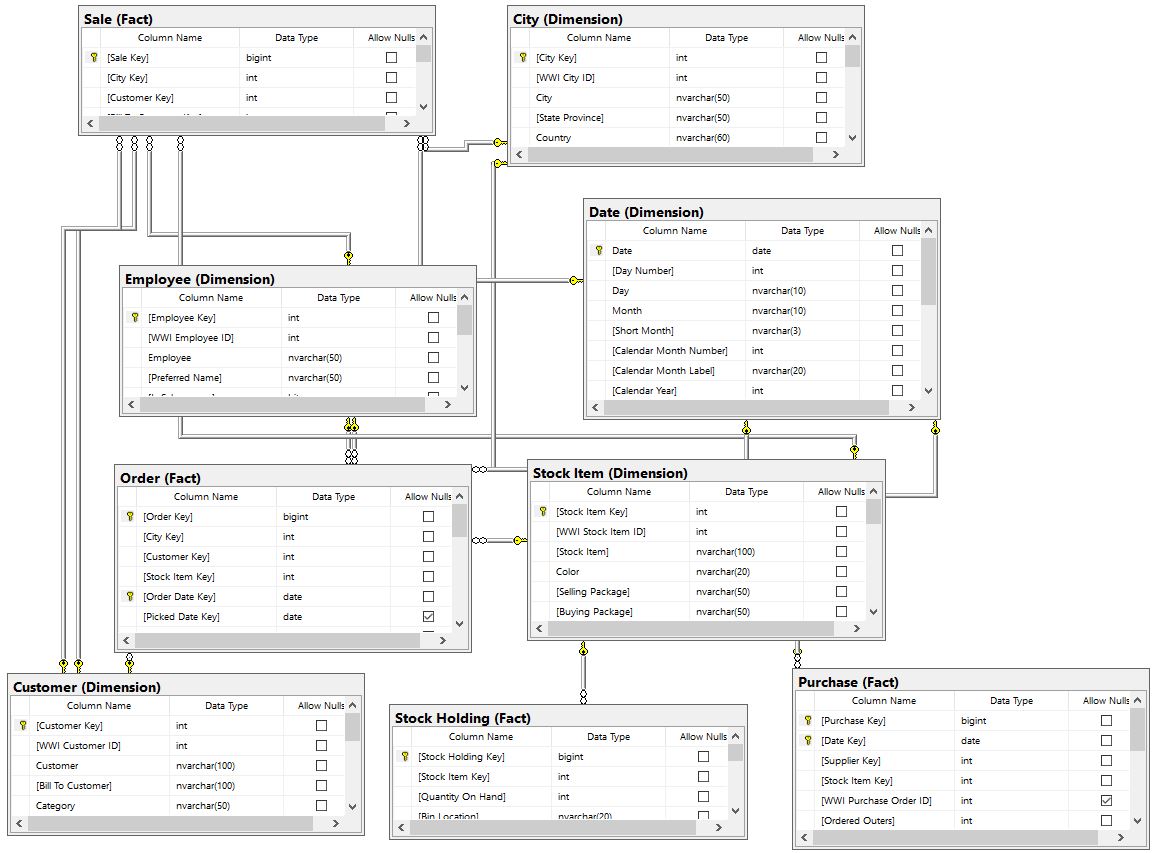
# 

# Diagrams of WideWorldImportersDW

## Show the diagrams in standard and key view



## Standard View of WideWorldImportersDW.



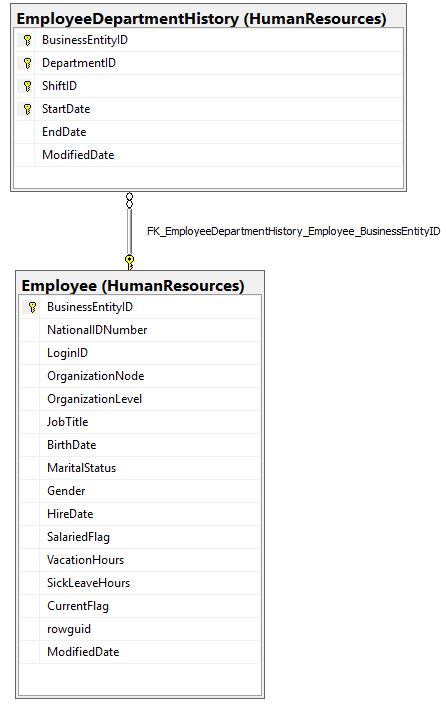
# Problem 01: List all workers in department 7 using AdventureWorks2014.

## What we need to do to solve this problem:

We want to show all Employees that work in department 7 by displaying their BusinessEntityID, LoginID, and DepartmentID. We use a where clause to show only employees who have a DepartmentID equal to 7.

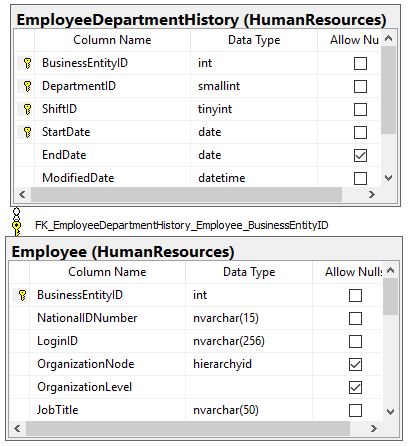
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the EmployeeDepartmentHistory table, and the Employee table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Employee | BusinessEntityID |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| HumanResources.Employee | BusinessEntityID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Employee\_1.BusinessEntityID

,Employee\_1.LoginID

,edh.DepartmentID

FROM HumanResources.Employee AS Employee\_1

INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh

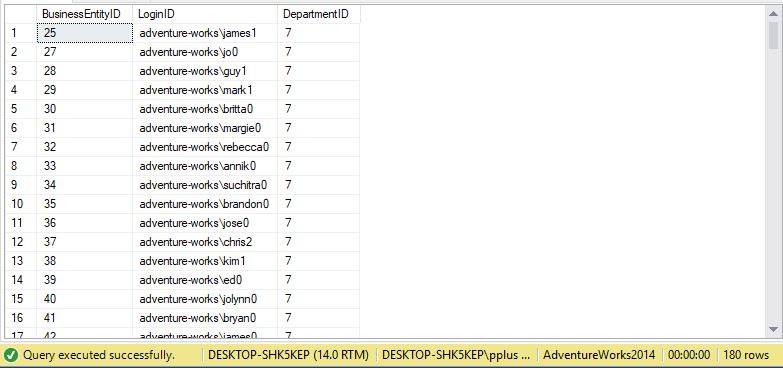
ON Employee\_1.BusinessEntityID = edh.BusinessEntityID

WHERE edh.DepartmentID = 7

## ORDER BY Employee\_1.BusinessEntityID;

## 

## Sample Output with (180) rows



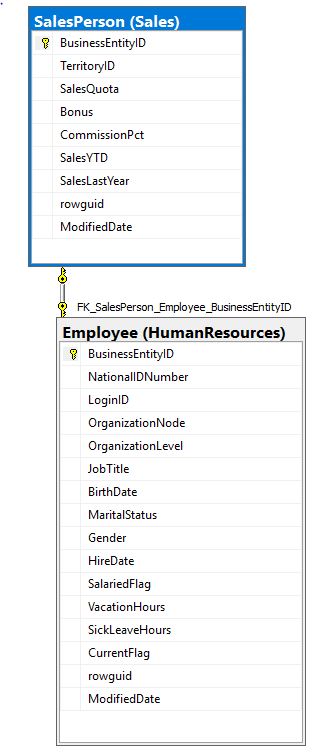
# Problem 02: Find all Sales representatives that made no sales last year using AdventureWorks2014.

## What we need to solve this problem:

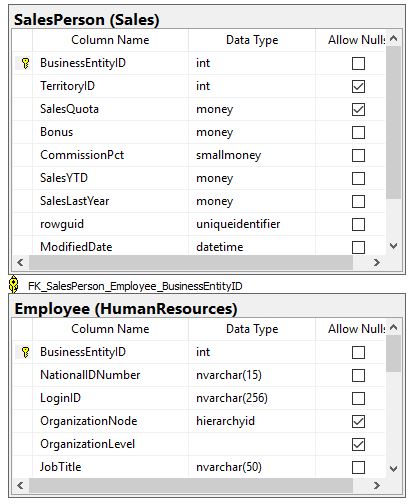
We want to show all sales representatives that made no sales last year by displaying their BusinessEntityID, and use a where clause to show only employees who have a DepartmentID equal to 7. We want to make sure all employees listed are sales representatives and not managers, as managers do are not responsible for making sales.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesPerson table, and the Employee table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.SalesPerson | BusinessEntityID |
| Sales.SalesPerson | SalesLastYear |
| Sales.SalesPerson | SalesYTD |
| HumanResources.Employee | LoginID |
| HumanResources.Employee | JobTitle |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Salesp.BusinessEntityID

,Salesp.SalesLastYear

,Salesp.SalesYTD

,Employee\_1.LoginID

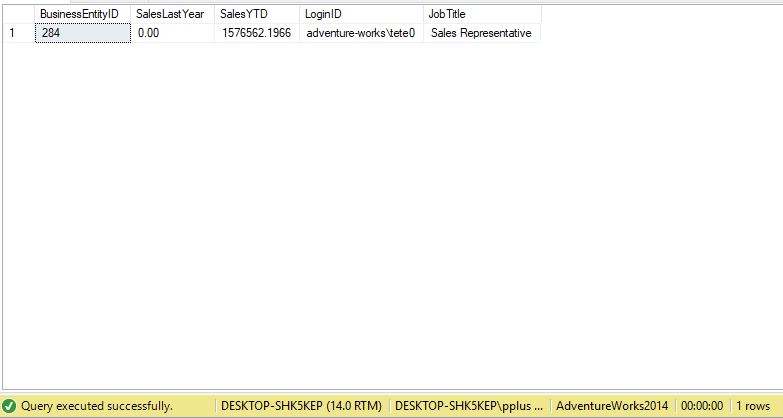
,Employee\_1.JobTitle

FROM Sales.SalesPerson AS Salesp INNER JOIN

HumanResources.Employee AS Employee\_1 ON Salesp.BusinessEntityID = Employee\_1.BusinessEntityID

WHERE (Salesp.SalesLastYear = 0 AND Employee\_1.OrganizationLevel = 3);

## Sample Output with (1) row



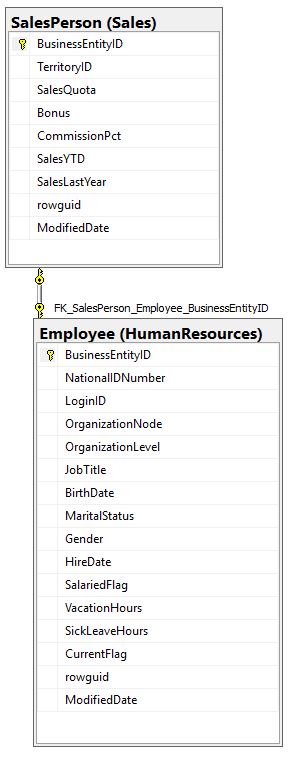
# Problem 03: Find all employees that work in sales using AdventureWorks2014.

## What we need to solve this problem:

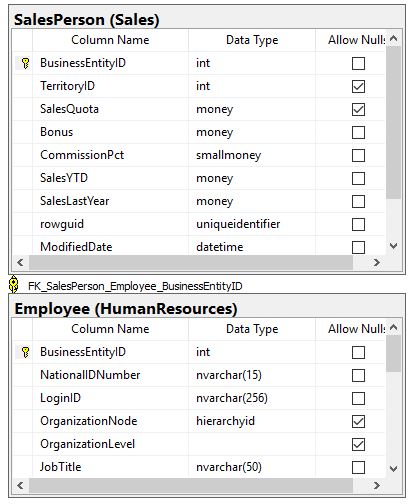
We want to show all Employees that work in sales. We can accomplish this by displaying their BusinessEntityID, LoginID, and JobTitle. We use an inner join so that all BusinessEntityIDs in both the SalesPerson and Employee tables are matched. Then, we want to put all results in order, by BusinessEntityID.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesPerson table, and the Employee table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Employee | BusinessEntityID  LoginID  JobTitle |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| HumanResources.Employee | BusinessEntityID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

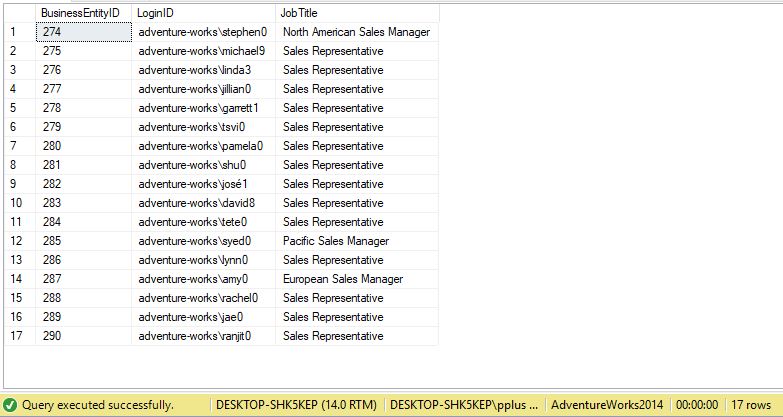
SELECT Employee\_1.BusinessEntityID, Employee\_1.LoginID, Employee\_1.JobTitle

FROM Sales.SalesPerson INNER JOIN

HumanResources.Employee AS Employee\_1 ON Sales.SalesPerson.BusinessEntityID = Employee\_1.BusinessEntityID

ORDER BY Employee\_1.BusinessEntityID;

## Sample Output with (17) rows



# Problem 04: Show all the employees and how frequently they are paid using AdventureWorks2014.

## What we need to solve this problem:

We want to show how often all employees are paid. We can accomplish this by displaying their BusinessEntityID , LoginID and PayFrequency.

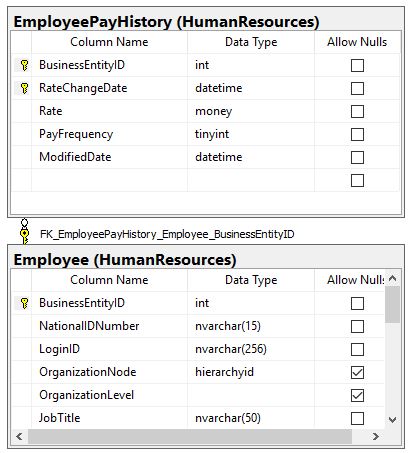
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the EmployeePayHistory table, and the Employee table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Employee | BusinessEntityID |
| HumanResources.EmployeePayHistory | PayFrequency |

## Query to solve this problem:

USE AdventureWorks2014;

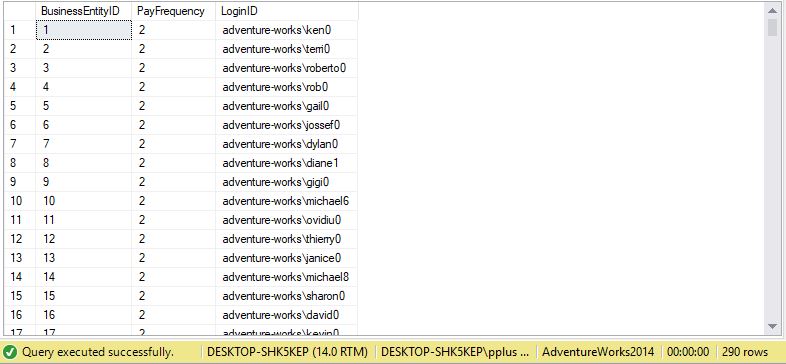
GO

SELECT DISTINCT Employee\_1.BusinessEntityID, Payh.PayFrequency, Employee\_1.LoginID

FROM HumanResources.Employee AS Employee\_1

INNER JOIN HumanResources.EmployeePayHistory AS Payh ON Employee\_1.BusinessEntityID = Payh.BusinessEntityID;

## Sample Output with (290) rows



# Problem 05: Show all sales from Pennsylvania with more than 50 item quantities ordered using WideWorldImportersDW.

## What we need to solve this problem:

We want to show all orders made in Pennsylvania with more than 50 item quantities ordered. We can accomplish this by displaying Order Key, Quantity and Description of an order, as well as the City Key, City, and State Province column in the result.

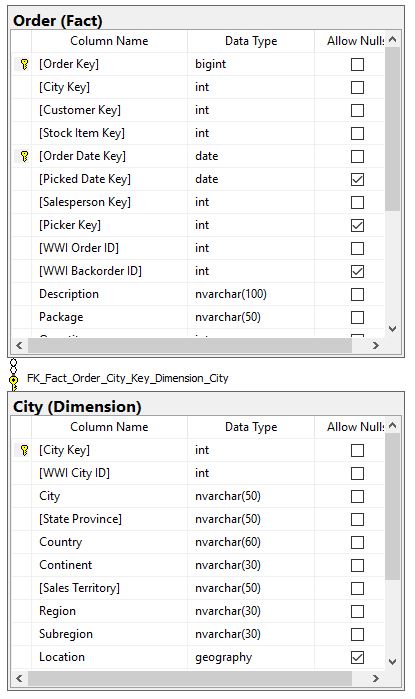
## Use WideWorldImportersDW.

## Diagram(s) of tables

We will be using the Fact.[Order] table, and the Dimension.City table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Dimension.City | [City Key]  City  [State Province] |
| Fact.[Order] | Description  Quantity  [Order Key] |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Fact.[Order] | Quantity | ASC |

## Query to solve this problem:

USE WideWorldImportersDW;

GO

SELECT Fact.[Order].[Order Key]

,Dimension.City.[City Key]

, Dimension.City.City

, Dimension.City.[State Province]

, Fact.[Order].Description

, Fact.[Order].Quantity

FROM Dimension.City INNER JOIN

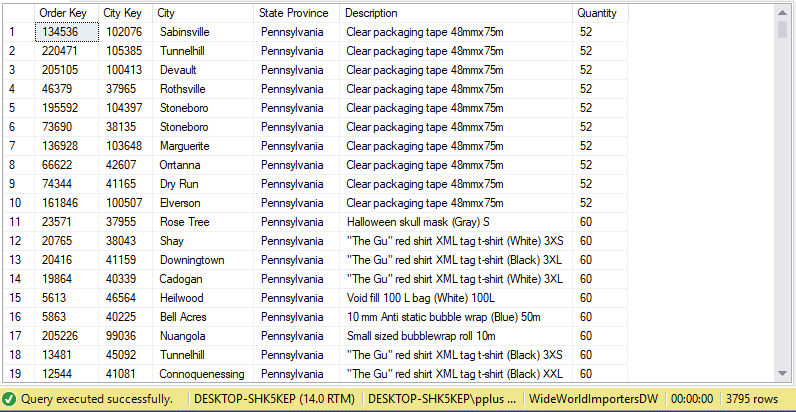
Fact.[Order] ON Dimension.City.[City Key] = Fact.[Order].[City Key]

WHERE Dimension.City.[State Province] = 'Pennsylvania'

AND Fact.[Order].Quantity > 50

ORDER BY Fact.[Order].Quantity ASC;

## Sample Output with (3795) rows



# Problem 06: Find by CustomerID, the number of distinct items and total quantities of all items ever ordered by customer using AdventureWorks2014.

## What we need to solve this problem:

We want to show all the DistinctItems and TotalItems each Customer has ever purchased. We do this by showing all customers by CustomerID, counting all rows as DistinctItems, and sum of all OrderQty as TotalItems.

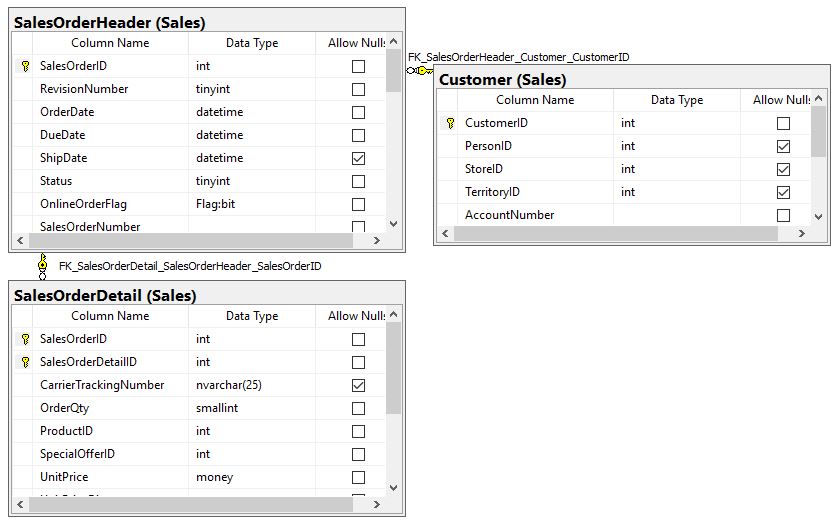
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | TotalItems  DistinctItems |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID,

COUNT(\*) as DistinctItems, SUM (Sales.SalesOrderDetail.OrderQty) as TotalItems

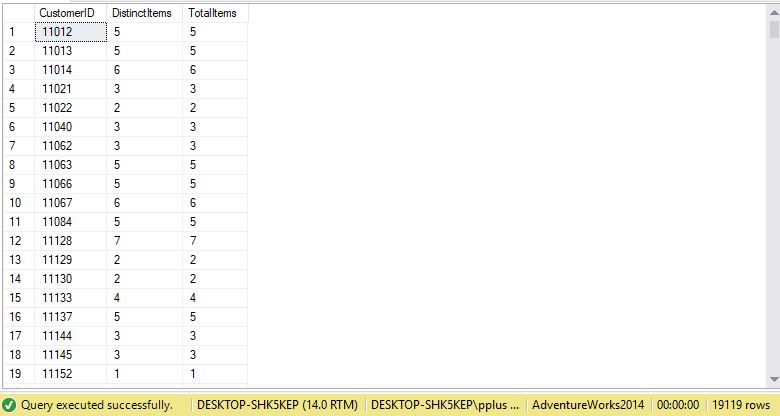
FROM Sales.Customer

INNER JOIN Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID

INNER JOIN Sales.SalesOrderDetail ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID

GROUP BY Sales.Customer.CustomerID;

## Sample Output with (19119) rows



# Problem 07: Find by CustomerID, the number of all orders ever made by customer using AdventureWorks2014.

## What we need to solve this problem:

We want to show all the orders a customer has ever made. To do this, we group all rows by CustomerID to have a single column per customer. Next, we count all distinct SalesOrderIDs. Then, we order all by CustomerID.

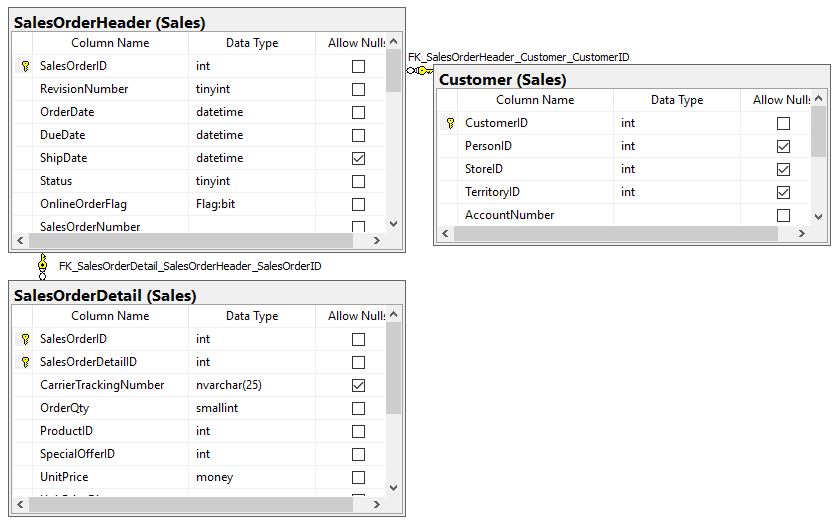
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | OrderQuantity |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customer | CustomerID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID,

COUNT(DISTINCT Sales.SalesOrderDetail.SalesOrderID) as OrderQuantity

FROM Sales.Customer INNER JOIN

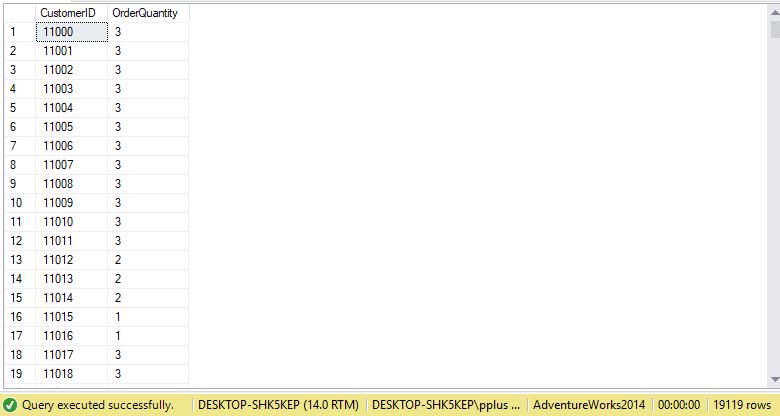
Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID INNER JOIN

Sales.SalesOrderDetail ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID

GROUP BY Sales.Customer.CustomerID

ORDER BY Sales.Customer.CustomerID;

## Sample Output with (19119) rows



# Problem 08: Find by CustomerID, all online orders and store orders a customer has ever made using AdventureWorks2014.

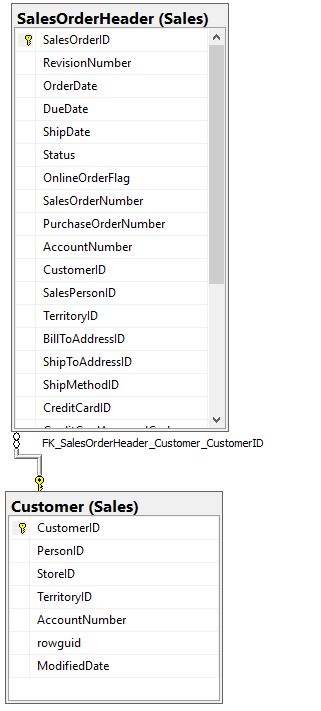
## What we need to solve this problem:

We want to show all the orders a customer has ever made online or in store. To do this, we group all rows by CustomerID to have a single column per customer. Next, we count all orders whose OnlineOrderFlag is true. Then, we count all orders whose OnlineOrderFlag is false.

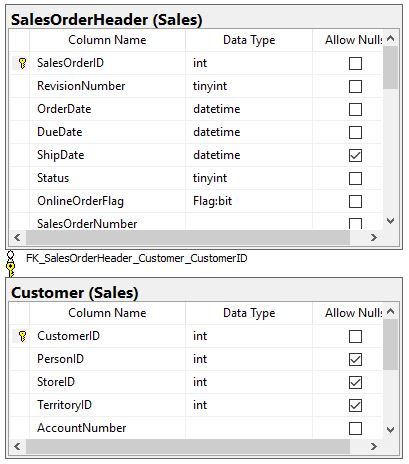
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table and the Customer table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | OnlineOrders  StoreOrders |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customer | CustomerID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID,

COUNT(CASE WHEN Sales.SalesOrderHeader.OnlineOrderFlag = 1 THEN 1 END) as OnlineOrders,

COUNT(CASE WHEN Sales.SalesOrderHeader.OnlineOrderFlag = 0 THEN 1 END) as StoreOrders

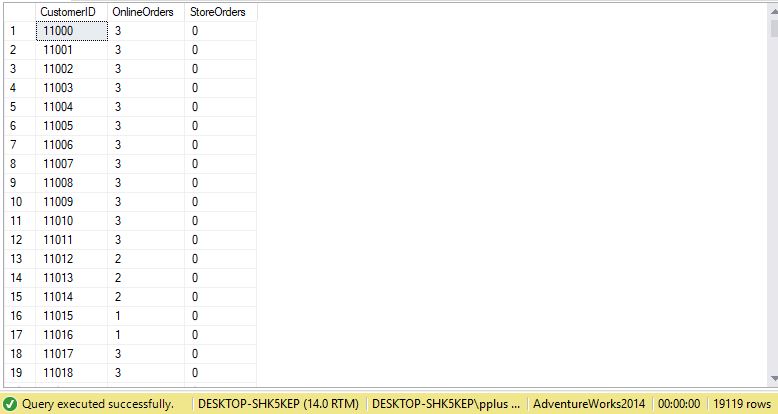
FROM Sales.Customer

INNER JOIN Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID

GROUP BY Sales.Customer.CustomerID

ORDER BY Sales.Customer.CustomerID;

## Sample Output with (19119) rows



# Problem 09: Find all customers who have ever recieved a discount on an ordered item using AdventureWorks2014.

## What we need to solve this problem:

We want to show all the orders that a customer has received a discount on. To do this, we group all rows by CustomerID to have a single column per customer. Next, we count SalesOrderID as the number of different items orders. Then, we count all rows that have a UnitPriceDiscount > 0. Finally, order by CustomerID.

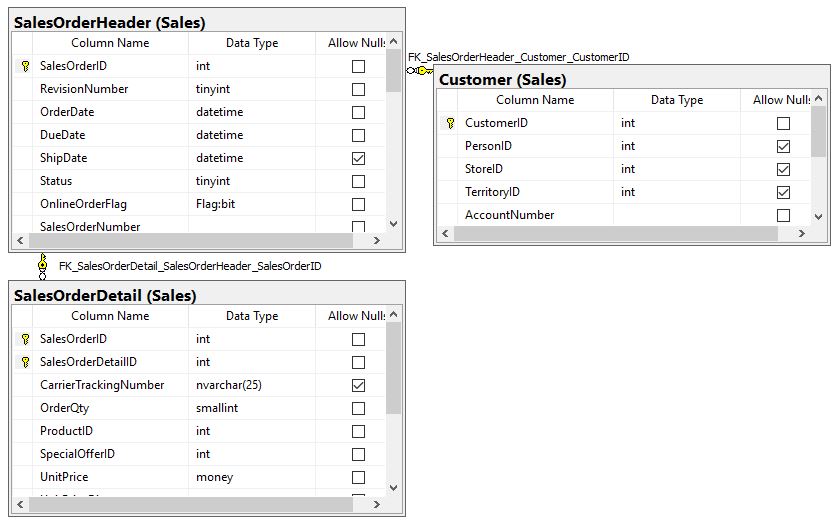
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | OrderQuantity  DiscountedItems |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customer | CustomerID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID

,COUNT(Sales.SalesOrderDetail.SalesOrderID) as OrderQuantity

,COUNT(CASE WHEN Sales.SalesOrderDetail.UnitPriceDiscount > 0.00 THEN 1 END) as DiscountedItems

FROM Sales.Customer INNER JOIN

Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID INNER JOIN

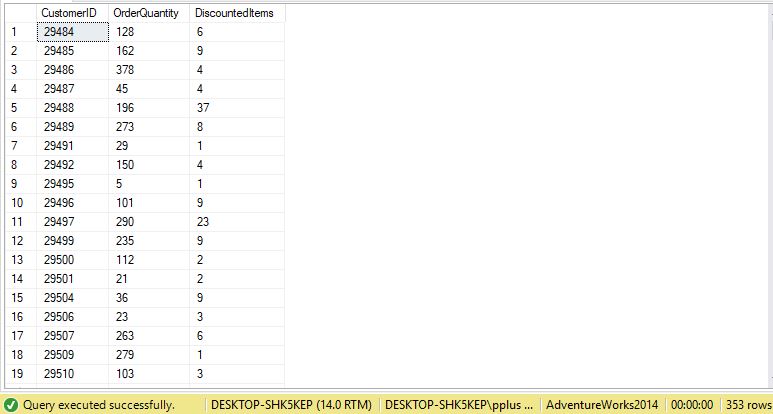
Sales.SalesOrderDetail ON Sales.SalesOrderHeader.SalesOrderID = Sales.SalesOrderDetail.SalesOrderID

GROUP BY Sales.Customer.CustomerID

HAVING COUNT(CASE WHEN Sales.SalesOrderDetail.UnitPriceDiscount > 0.00 THEN 1 END) > 0

ORDER BY Sales.Customer.CustomerID;

## Sample Output with (353) rows



# Problem 10: Find all customers who have made an order before 2012 using AdventureWorks2014.

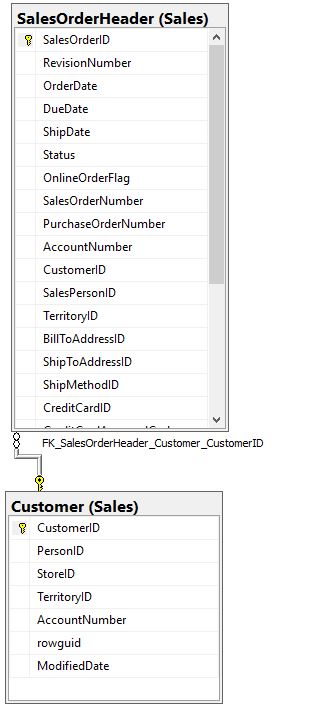
## What we need to solve this problem:

We want to show all customers that have made an order before 2012. We show all customers by CustomerID, then show the year an order was made. If that order was made before 2012, then the customer and year is shown. Order by CustomerID

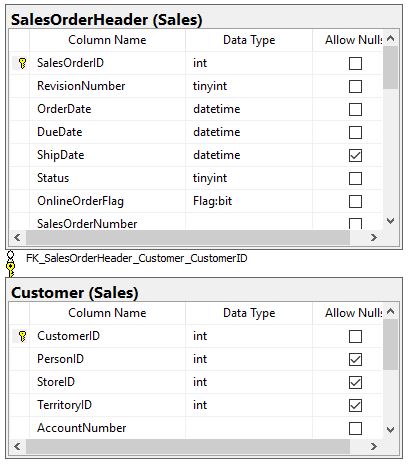
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table and the Customer table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | OrderYear |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customer | CustomerID | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID,

YEAR(Sales.SalesOrderHeader.OrderDate) as OrderYear

FROM Sales.Customer

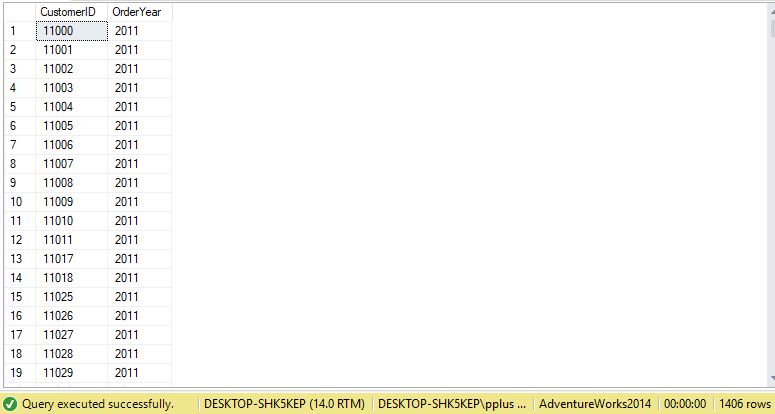
INNER JOIN Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID

WHERE YEAR(Sales.SalesOrderHeader.OrderDate) < 2012

GROUP BY Sales.Customer.CustomerID, YEAR(Sales.SalesOrderHeader.OrderDate)

ORDER BY Sales.Customer.CustomerID;

## Sample Output with (1406) rows



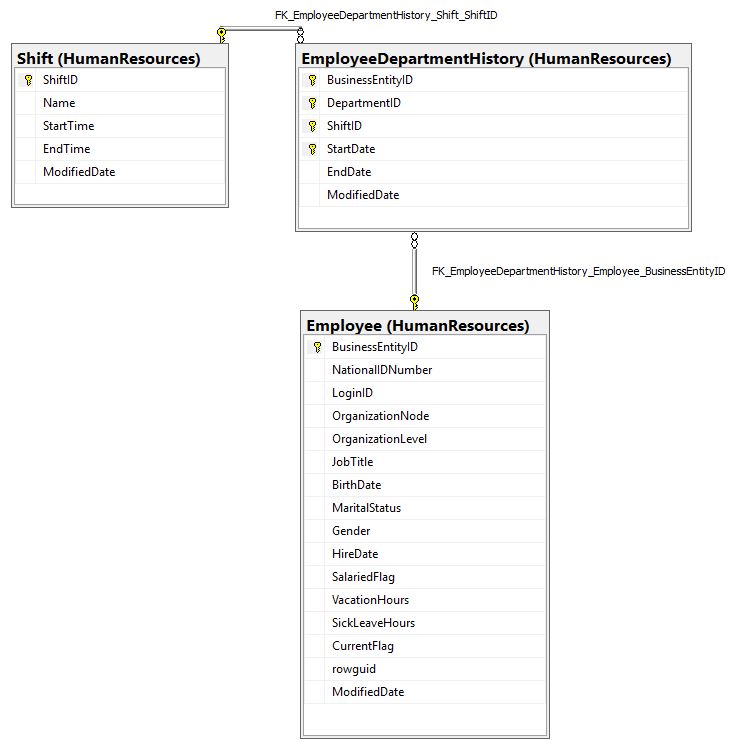
# Problem 11: Find total number workers in day, evening, and night shifts using AdventureWorks2014.

## What we need to solve this problem:

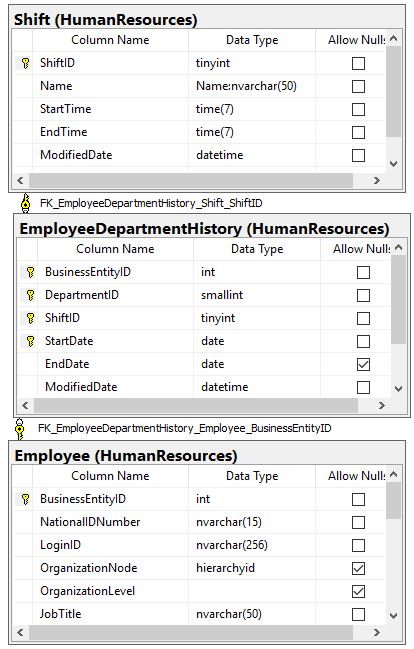
We want to show the total number of employees that work day, evening and night shifts. We show all shift schedules by ShiftID, then count all employees that have a ShiftID of 1, 2 or 3.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the Shift table, EmployeeDepartmentHistory table and the Employee table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Shift | ShiftID |
| DerivedColumn | DayEmployees  EveningEmployees  NightEmployees |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT HumanResources.Shift.ShiftID

,COUNT(CASE WHEN HumanResources.Shift.ShiftID = 1 THEN 1 END) AS DayEmployees

,COUNT(CASE WHEN HumanResources.Shift.ShiftID = 2 THEN 1 END) AS EveningEmployees

,COUNT(CASE WHEN HumanResources.Shift.ShiftID = 3 THEN 1 END) AS NightEmployees

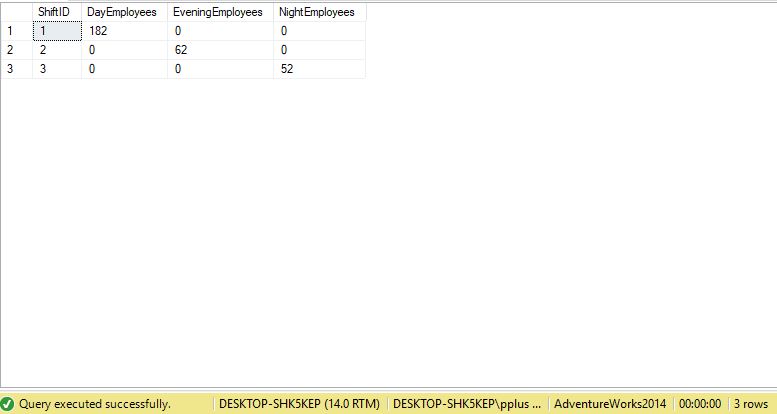
FROM HumanResources.Employee AS Employee\_1 INNER JOIN

HumanResources.EmployeeDepartmentHistory ON Employee\_1.BusinessEntityID = HumanResources.EmployeeDepartmentHistory.BusinessEntityID

INNER JOIN HumanResources.Shift ON HumanResources.EmployeeDepartmentHistory.ShiftID = HumanResources.Shift.ShiftID

GROUP BY HumanResources.Shift.ShiftID;

## Sample Output with (3) rows



# Problem 12: List all customers that have ordered lightweight bicycles using AdventureWorksDW2014.

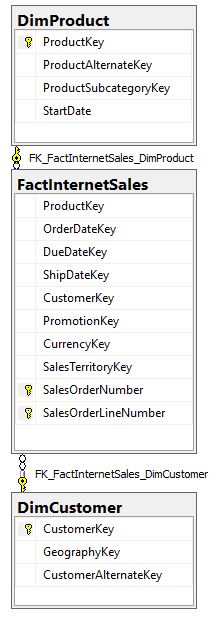
## What we need to solve this problem:

We want to show all customers that have ordered bicycle that weight less than 20 pounds. We will show all customers and group by CustomerID, and count all ordered bicycles that weight less than 20 pounds. We will sort the customers by most purchased bicycles to least purchased bicycles.

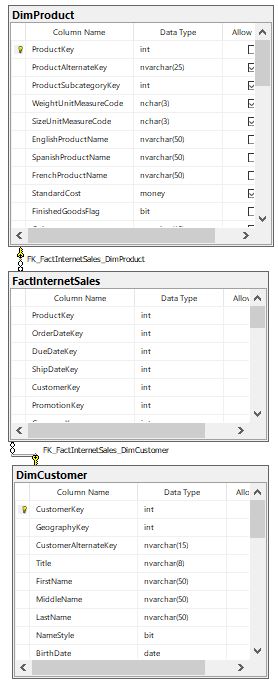
## Use AdventureWorksDW2014.

## Diagram(s) of tables

We will be using the DimProduct, the FactInternetSales and the DimCustomer table.



## Columns from Standard view



## 

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| DimCustomer | CustomerKey |
| Derived | LowWeightAmount |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Derived | LowWeightAmount | DESC |

## Query to solve this problem:

USE AdventureWorksDW2014;

GO

SELECT cust.CustomerKey, COUNT(DimProduct.ProductKey) AS LowWeightAmount

FROM DimProduct INNER JOIN

FactInternetSales ON DimProduct.ProductKey = FactInternetSales.ProductKey

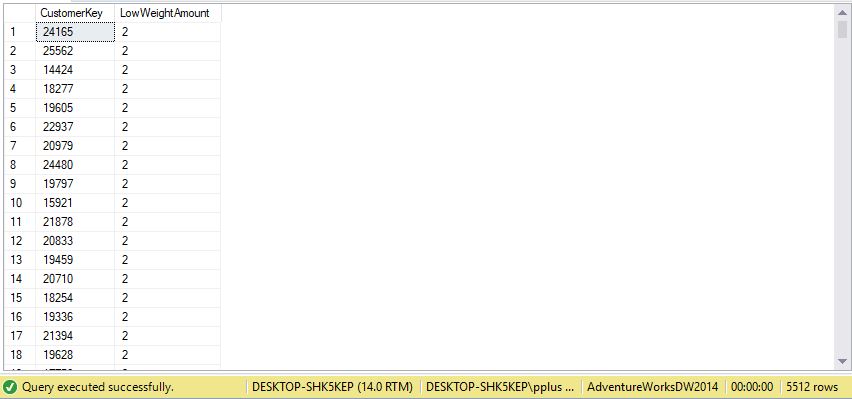
INNER JOIN DimCustomer AS cust On cust.CustomerKey = FactInternetSales.CustomerKey

WHERE DimProduct.Weight < 20

GROUP BY cust.CustomerKey

ORDER BY LowWeightAmount DESC;

## Sample Output with (5512) rows



# Problem 13: Find number of distinct currency rate ids per customer using AdventureWorks2014.

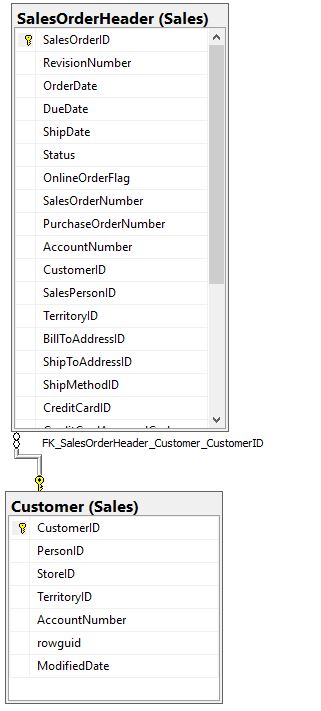
## What we need to solve this problem:

We want to show number of distinct currency rate ids per customer. We show all customers by CustomerID, then count all distinct current rate ids. We sort by customers who have the most currency rate ids.

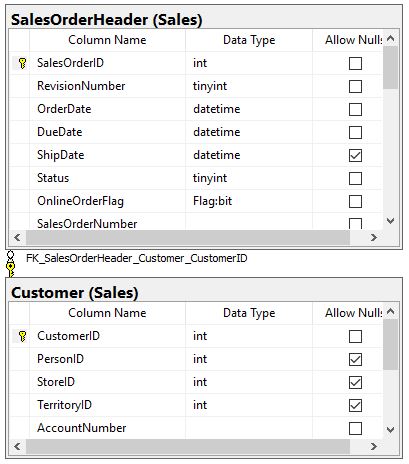
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader and the Customer table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | RateNums |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumn | RateNums | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Sales.Customer.CustomerID

,COUNT(DISTINCT Sales.SalesOrderHeader.CurrencyRateID) AS RateNums

FROM Sales.Customer INNER JOIN

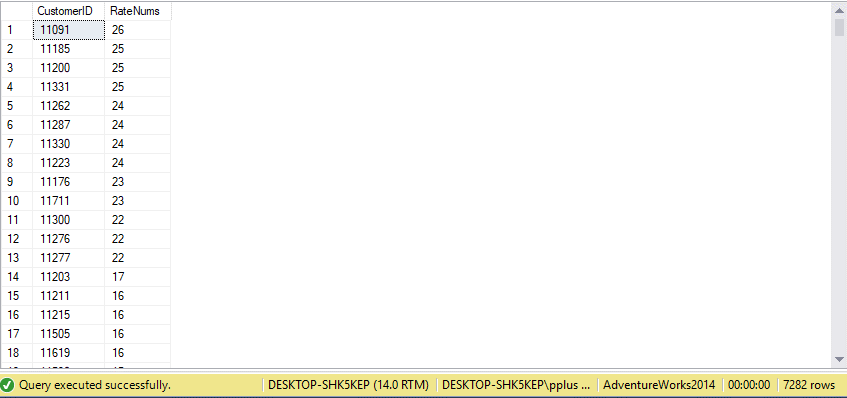
Sales.SalesOrderHeader ON Sales.Customer.CustomerID = Sales.SalesOrderHeader.CustomerID

GROUP BY Sales.Customer.CustomerID

HAVING COUNT(DISTINCT Sales.SalesOrderHeader.CurrencyRateID) > 0

ORDER BY Sales.Customer.CustomerID;

## Sample Output with (180) rows



# Problem 14: Show all the distinct departments employees belong to within an organization level in AdventureWorks2014.

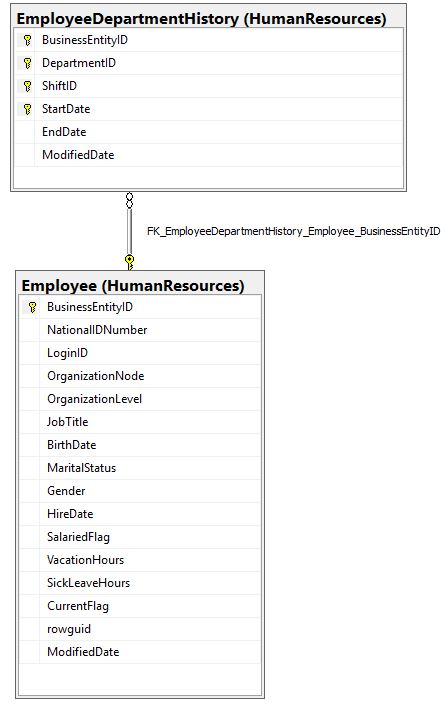
## What we need to solve this problem:

We want to show all department levels that work in a specific organization level. We display organization levels by OrganizationLevel, and count DepartmentID. We show the amount of DepartmentIDs per OrganizationLevel.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the EmployeeDepartmentHistory and the Employee table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Employee | OrganizationLevel |
| DerivedColumns | DistinctDepartments |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Emp.OrganizationLevel

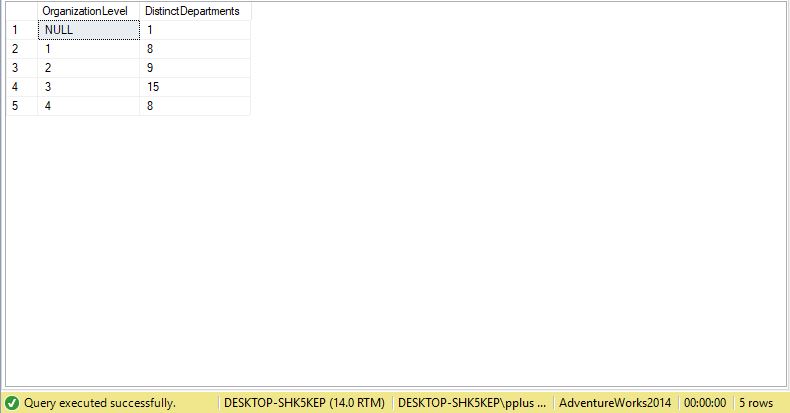
,COUNT(DISTINCT Hist.DepartmentID) AS DistinctDepartments

FROM HumanResources.Employee AS Emp

INNER JOIN HumanResources.EmployeeDepartmentHistory AS Hist ON Emp.BusinessEntityID = Hist.BusinessEntityID

GROUP BY Emp.OrganizationLevel;

## Sample Output with (5) rows



# Problem 15: Show top 50 customers by Total Spent ever using AdventureWorks2014.

## What we need to solve this problem:

We want to show top 50 customers who have spent the most money with our company. We select and group our results by CustomerID, take a sum of all they have ever spent, and the average spent per order. We sort our customers by TotalSpent.

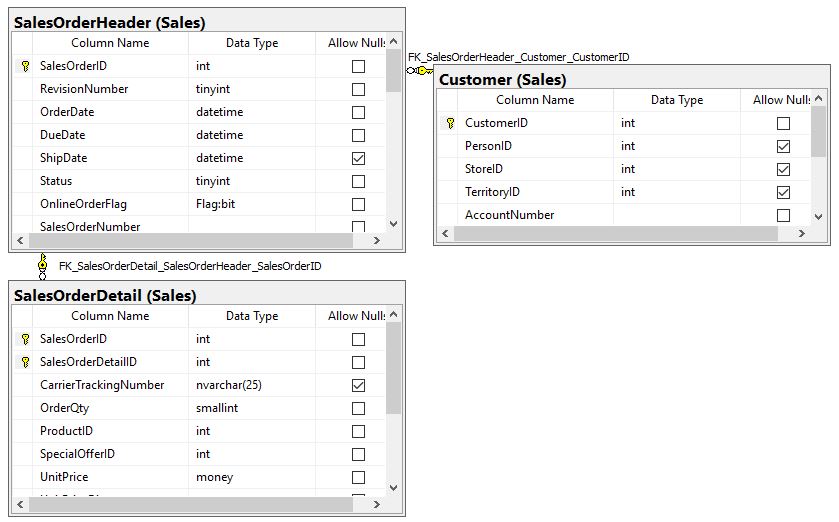
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | TotalSpent  AvgSpentPerOrder |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumn | TotalSpent | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT TOP(50) Cust.CustomerID

,SUM(DISTINCT Sod.LineTotal) AS TotalSpent

,AVG(DISTINCT Sod.LineTotal) AS AvgSpentPerOrders

FROM Sales.Customer AS Cust

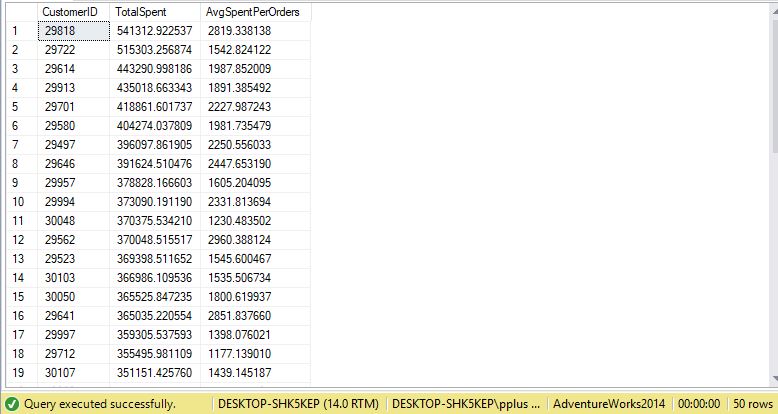
INNER JOIN Sales.SalesOrderHeader AS Soh ON Cust.CustomerID = Soh.CustomerID

INNER JOIN Sales.SalesOrderDetail AS Sod ON Sod.SalesOrderID = Soh.SalesOrderID

GROUP BY Cust.CustomerID

ORDER BY TotalSpent DESC

## Sample Output with (50) rows



# Problem 16: Show list of Top 100 Best Selling Products in 2013 using AdventureWorks2014.

## What we need to solve this problem:

We want to show top 100 best selling products in year 2013. We select products by ProductID, count the total amount of items sold by ProductID, and calculate the year it was sold. We filter all orders by year, and list Top 100 products by ProductID, and sort by the amount sold.

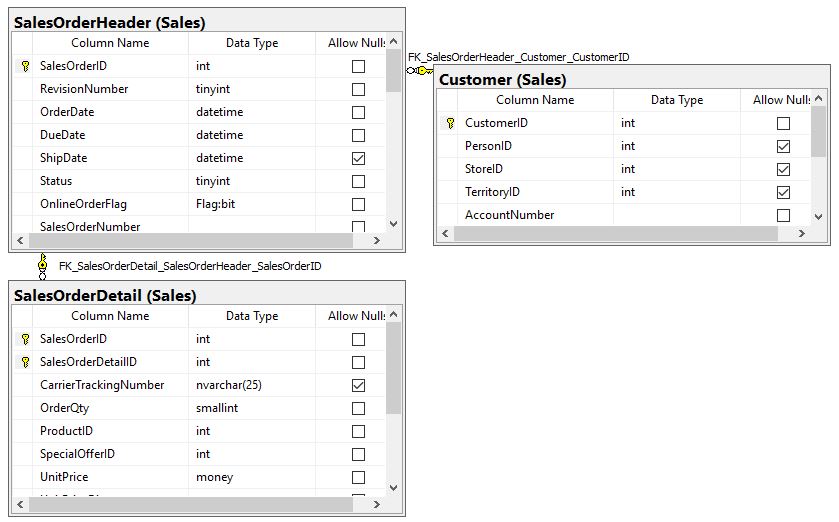
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.SalesOrderDetail | ProductID |
| DerivedColumn | QuantitySold  Year |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumn | QuantitySold | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT TOP(100) Sod.ProductID, COUNT(Sod.OrderQty) AS QuantitySold, YEAR(Soh.Orderdate) AS Year

FROM Sales.Customer As Cust

INNER JOIN Sales.SalesOrderHeader AS Soh ON Cust.CustomerID = Soh.CustomerID

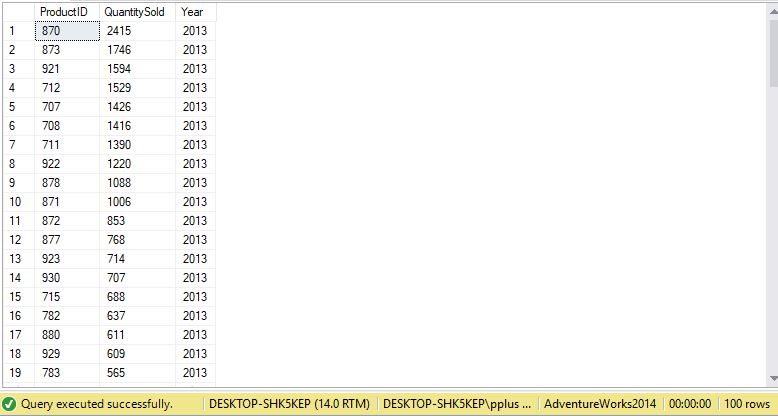
INNER JOIN Sales.SalesOrderDetail AS Sod ON Soh.SalesOrderID = Sod.SalesOrderID

WHERE YEAR(Soh.Orderdate) = 2013

GROUP BY Sod.ProductID, YEAR(Soh.Orderdate)

ORDER BY QuantitySold DESC;

## Sample Output with (50) rows



# Problem 17: Show list of Employees who have worked the longest with the company using AdventureWorks2014.

## What we need to solve this problem:

We want to list all employees according to how long they have been with the company, and how old they were when they started. We need to list the employees by EmployeeID, and calculate their ages when they started working with us as well as how long they have been working as of 2014. We sort the list of employees from longest to shortest amount of time.

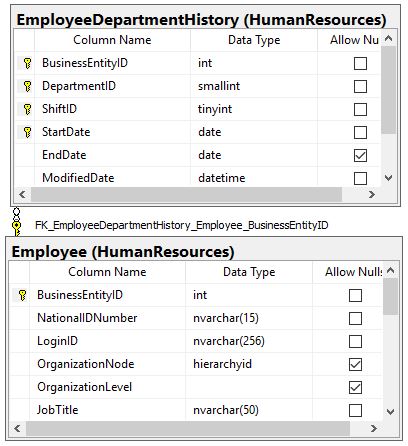
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the EmployeeDepartmentHistory and the Employee table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.EmployeeDepartmentHistory | BusinessEntityID |
| DerivedColumns | BirthYear  Start  StartAge  WorkYear |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumns | WorkYear | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Employee\_1.BusinessEntityID

, YEAR(Employee\_1.BirthDate) AS BirthYear

, YEAR(edh.StartDate) AS Start

, StartAge = (YEAR(edh.StartDate) - YEAR(Employee\_1.BirthDate))

, WorkYear = (2014 - YEAR(edh.StartDate))

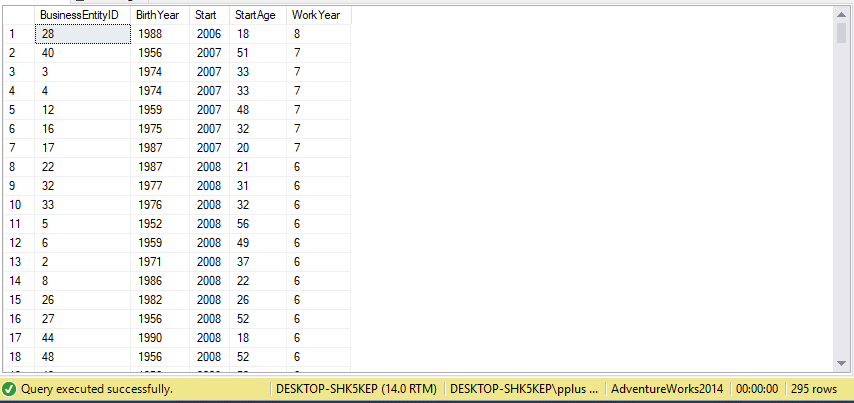
FROM HumanResources.Employee AS Employee\_1 INNER JOIN

HumanResources.EmployeeDepartmentHistory AS edh ON Employee\_1.BusinessEntityID = edh.BusinessEntityID

GROUP BY Employee\_1.BusinessEntityID, YEAR(Employee\_1.BirthDate), YEAR(edh.StartDate)

ORDER BY WorkYear DESC;

## Sample Output with (295) rows



# Problem 18: Show list of total sick leave hours per Department using AdventureWorks2014.

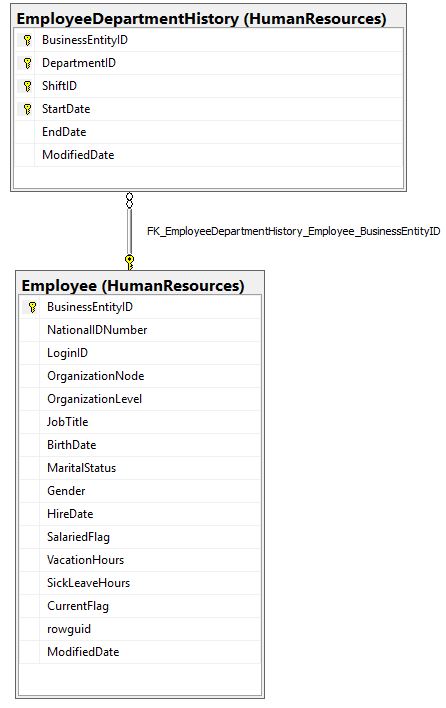
## What we need to solve this problem:

We want to show total amount of sick hours per Department level. We show departments by DepartmentID, and add up total amount of sick hours in each department. We sort by ascending number of sick leave hours.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the EmployeeDepartmentHistory and the Employee table.



## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.EmployeeDepartmentHistory | DepartmentID |
| DerivedColumns | TotalLeave |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumns | TotalLeave | ASC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT HR.DepartmentID, SUM(Employee\_1.SickLeaveHours) AS TotalLeave

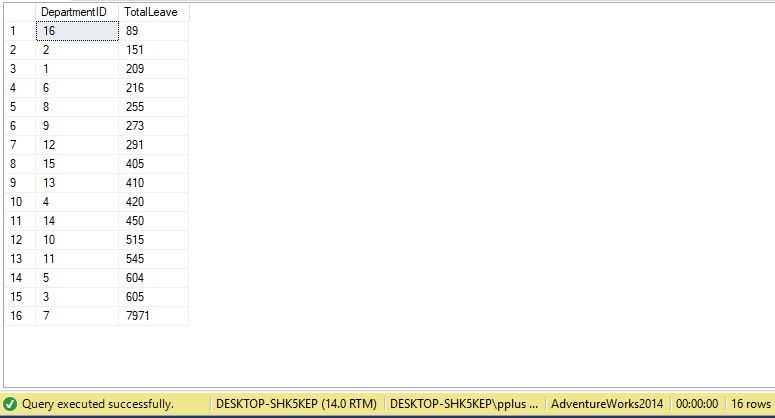
FROM HumanResources.Employee AS Employee\_1 INNER JOIN

HumanResources.EmployeeDepartmentHistory AS HR ON Employee\_1.BusinessEntityID = HR.BusinessEntityID

GROUP BY HR.DepartmentID

ORDER BY TotalLeave;

## Sample Output with (16) rows



# Problem 19: Show list of stores with most distinct products sold using AdventureWorks2014.

## What we need to solve this problem:

We want to show which stores have the greatest amount of distinct products. We show stores by StoreID, and make distinct rows per store by group by. Then, we count distinct ProductIDs in each store, and sort by stores which have the greatest number of distinct products.

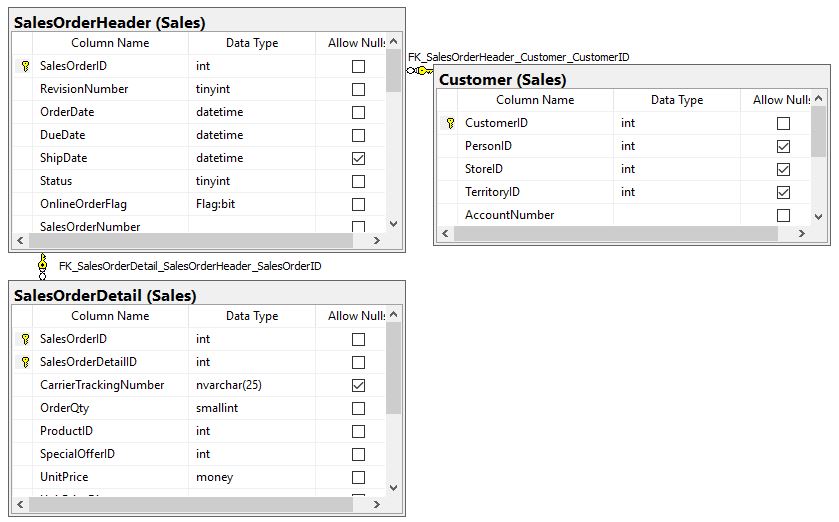
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | StoreID |
| DerivedColumn | ProductsInStore |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumn | ProductsInStore | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Cust.StoreID

,COUNT(DISTINCT Sod.ProductID) AS ProductsInStore

FROM Sales.Customer AS Cust INNER JOIN

Sales.SalesOrderHeader AS Soh ON Cust.CustomerID = Soh.CustomerID INNER JOIN

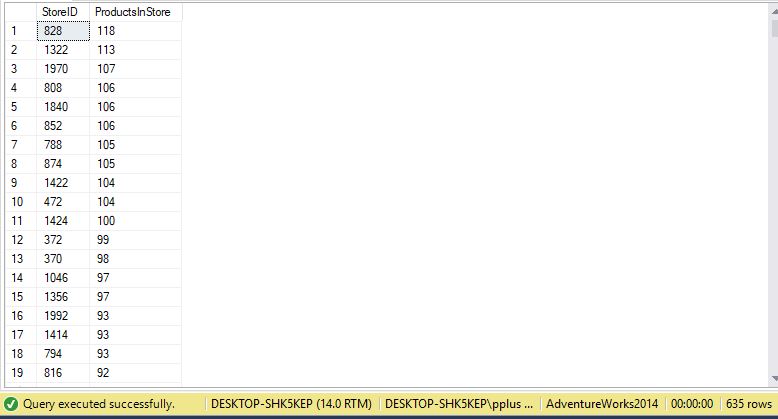
Sales.SalesOrderDetail AS Sod ON Soh.SalesOrderID = Sod.SalesOrderID

WHERE Cust.StoreID IS NOT NULL

GROUP BY Cust.StoreID

ORDER BY ProductsInStore DESC;

## Sample Output with (635) rows



# Problem 20: Show total amount of money made by a store using AdventureWorks2014.

## What we need to solve this problem:

We want to show which stores have made the most money. We show stores by StoreID, and take a sum of the amount of money they have made. We sort by stores that have made the most money.

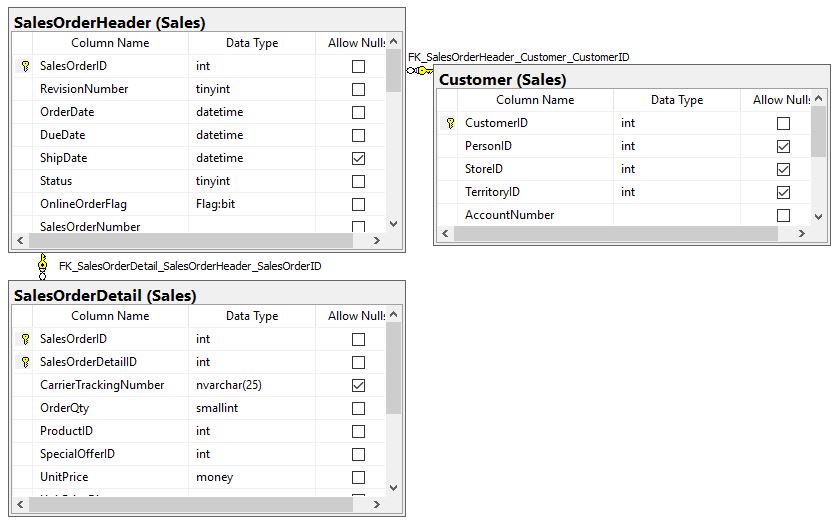
## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | StoreID |
| DerivedColumn | TotalMade |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DerivedColumn | TotalMade | DESC |

## Query to solve this problem:

USE AdventureWorks2014;

GO

SELECT Cust.StoreID

,SUM(Sod.LineTotal) AS TotalMade

FROM Sales.Customer AS Cust INNER JOIN

Sales.SalesOrderHeader AS Soh ON Cust.CustomerID = Soh.CustomerID INNER JOIN

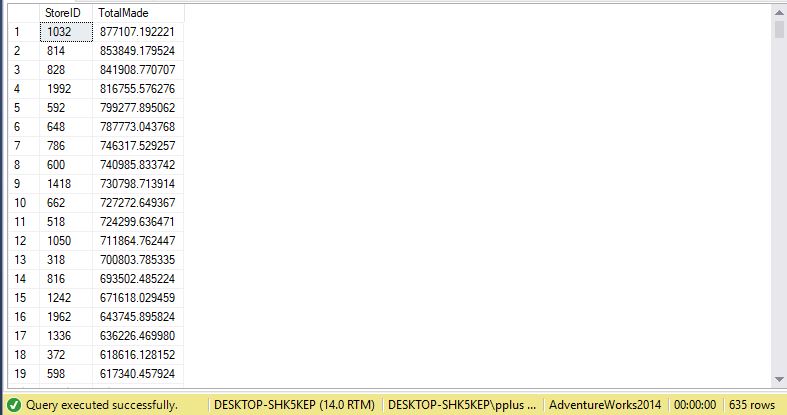
Sales.SalesOrderDetail AS Sod ON Soh.SalesOrderID = Sod.SalesOrderID

WHERE Cust.StoreID IS NOT NULL

GROUP BY Cust.StoreID

ORDER BY TotalMade DESC;

## Sample Output with (635) rows



# Problem 21: Show total employees in and not in management using AdventureWorks2014.

## What we need to solve this problem:

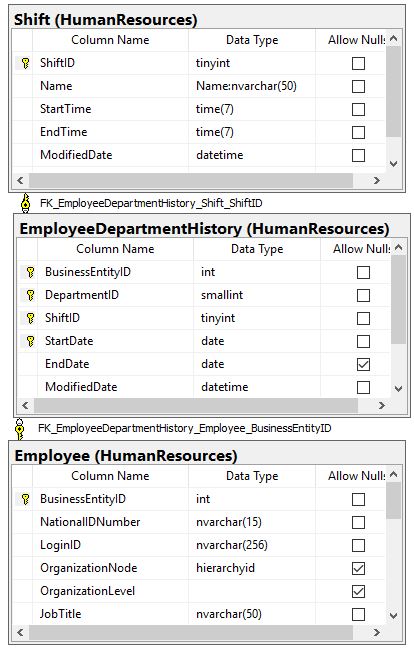
We want to show whether an employee is in, or is not in management. We use a function to determine if an employee is a manager, and count each determination. We show true or false for management and the amount of workers in each section.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the Shift table, EmployeeDepartmentHistory table and the Employee table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| DerivedColumn | isManager  NumberOfEmployees |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.isManager', N'FN') IS NOT NULL

DROP FUNCTION isManager;

GO

CREATE FUNCTION dbo.isManager(@EID int)

RETURNS bit

AS

BEGIN

DECLARE @ret AS bit;

IF (@EID > 3 )

SET @ret = 0;

ELSE SET @ret = 1;

RETURN @ret;

END;

USE AdventureWorks2014;

GO

SELECT dbo.isManager(Emp.OrganizationLevel) AS isManager, COUNT(HumanResources.EmployeeDepartmentHistory.BusinessEntityID) AS NumberOfEmployees

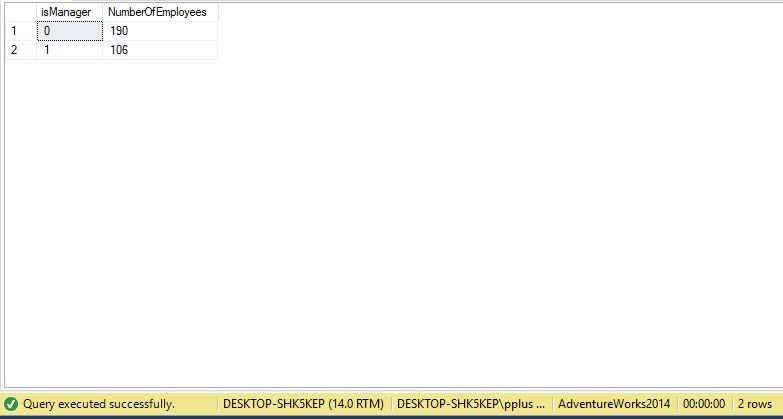
FROM HumanResources.Employee AS Emp

INNER JOIN HumanResources.EmployeeDepartmentHistory ON Emp.BusinessEntityID = HumanResources.EmployeeDepartmentHistory.BusinessEntityID

INNER JOIN HumanResources.Shift ON HumanResources.EmployeeDepartmentHistory.ShiftID = HumanResources.Shift.ShiftID

GROUP BY dbo.isManager(Emp.OrganizationLevel);

## Sample Output with (2) rows



# Problem 22: Show all customers that ordered a discontinued item using TSQLV4.

## What we need to solve this problem:

We want to show all customers that purchased a discontinued item. We show all customers by custid, and use a scalar function isDiscontinued to determine if a customer purchased a discontinued item, and count how many discontinued items there were. We then show the contact name, as well as the phone number and fax number associated with that customer. Sort by custid.

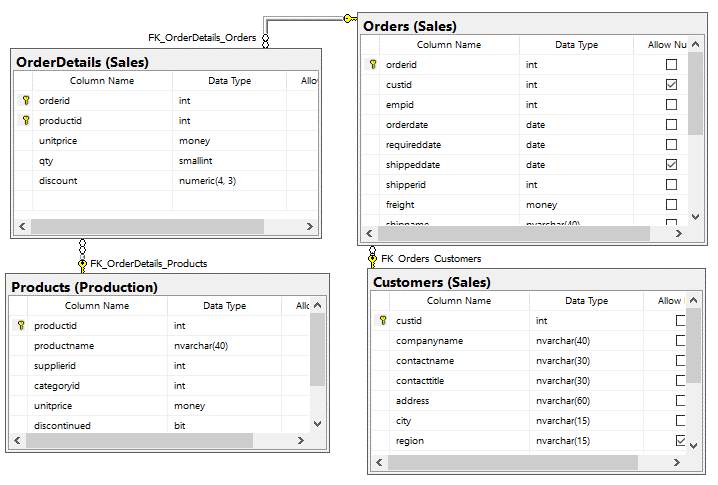
## Use TSQLV4.

## Diagram(s) of tables

We will be using the Products table, the Orders table, the OrderDetails table, and the Customers table.

## 

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customers | custid  Contactname  Phone  fax |
| DerivedColumn | NumDiscontinued |

## Group By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customers | custid | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.isDiscontinued', N'FN') IS NOT NULL

DROP FUNCTION isDiscontinued;

GO

CREATE FUNCTION dbo.isDiscontinued(@EID bit)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID > 0 )

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE TSQLV4;

GO

SELECT Cust.custid

, SUM(dbo.isDiscontinued(Pro.discontinued)) AS NumDiscontinued

, Cust.contactname

, Cust.phone

, Cust.fax

FROM Sales.Customers AS Cust

INNER JOIN Sales.Orders as So ON Cust.custid = So.custid

INNER JOIN Sales.OrderDetails as Sod ON Sod.orderid = So.orderid

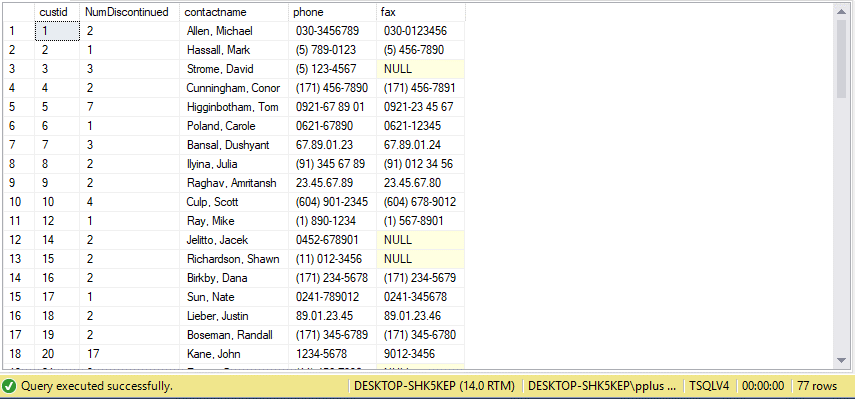
INNER JOIN Production.Products AS Pro ON Pro.productid = Sod.productid

GROUP BY Cust.custid, Cust.contactname, Cust.phone, Cust.fax

HAVING SUM(dbo.isDiscontinued(Pro.discontinued)) > 0

ORDER BY Cust.custid ASC;

## Sample Output with (77) rows



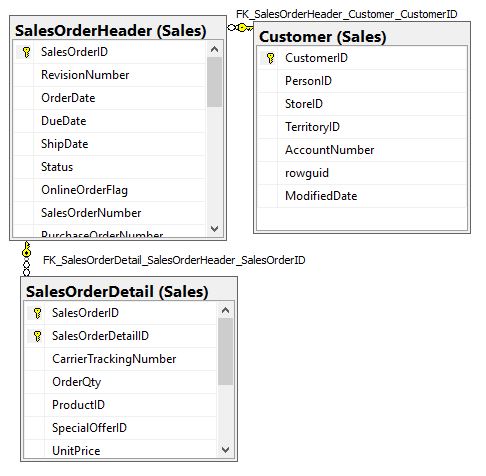
# Problem 23: Determine how many expensive orders a customer has made using AdventureWorks2014.

## What we need to solve this problem:

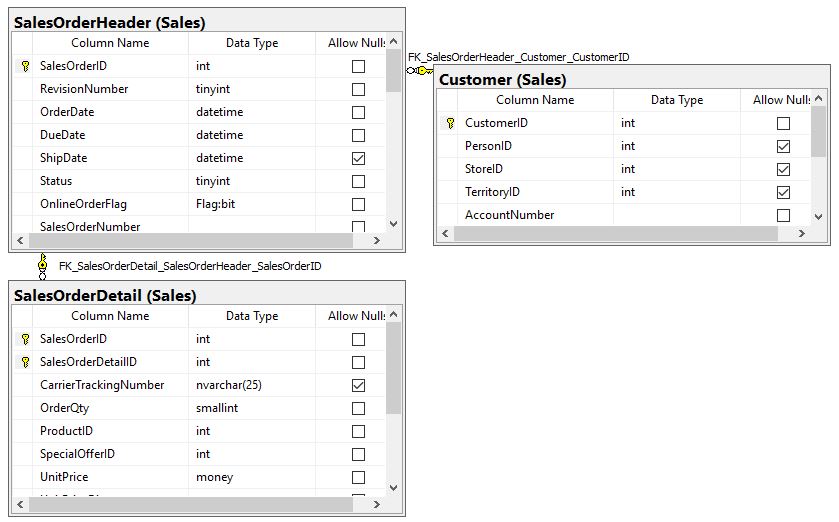
We want to show all customers that have made an expensive order. We show all customers by CustomerID, and use a scalar function isExpensive to determine if a customer order was expensive. We show how many expensive orders each customer has made.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | TotalExpensiveOrders |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.isExpensive', N'FN') IS NOT NULL

DROP FUNCTION isExpensive;

GO

CREATE FUNCTION dbo.isExpensive(@EID int)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID > 100 )

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE AdventureWorks2014;

GO

SELECT Cust.CustomerID, SUM(dbo.isExpensive(Sod.LineTotal)) AS TotalExpensiveOrders

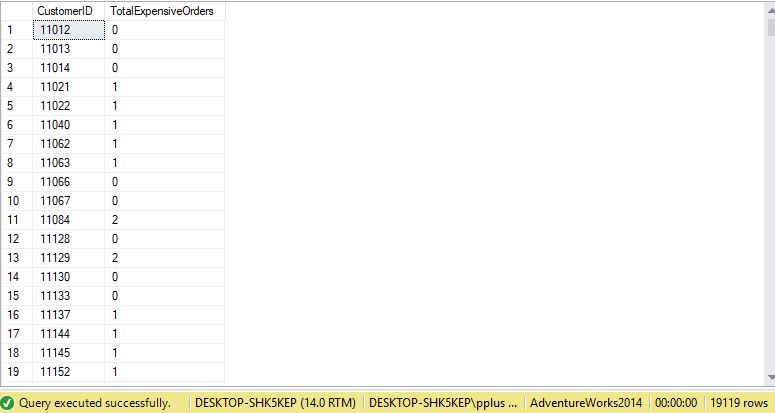
FROM Sales.Customer AS Cust

INNER JOIN Sales.SalesOrderHeader as Soh ON Cust.CustomerID = Soh.CustomerID

INNER JOIN Sales.SalesOrderDetail as Sod ON Sod.SalesOrderID = Soh.SalesOrderID

GROUP BY Cust.CustomerID

## Sample Output with (353) rows



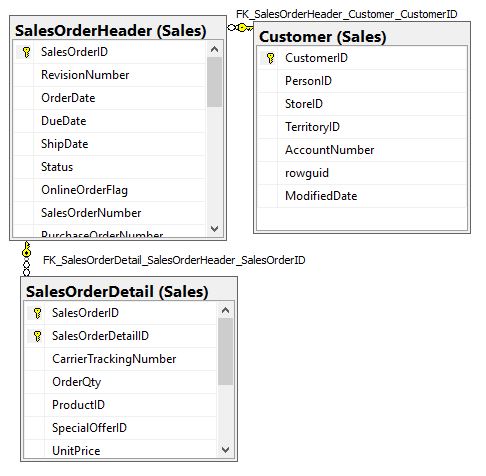
# Problem 24: Determine how many customer orders were picked up using AdventureWorks2014.

## What we need to solve this problem:

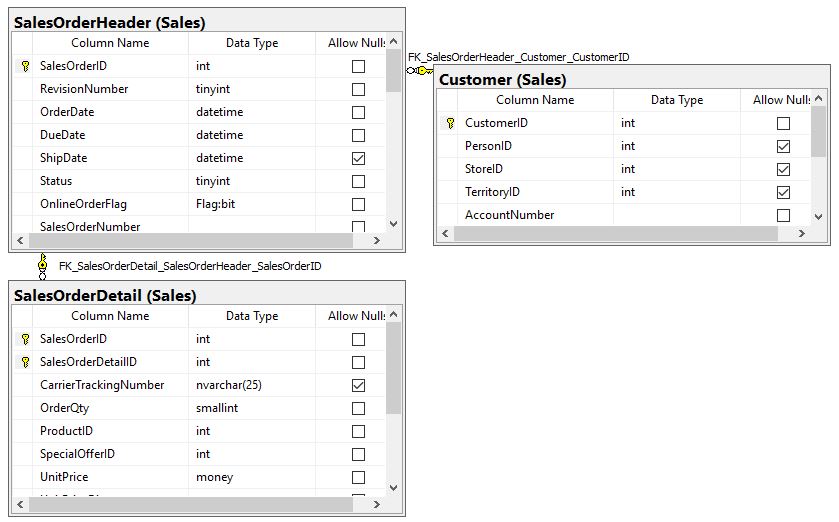
We want to show all orders that customers have picked up. We show all customers by CustomerID, and use a scalar function wasPicked to determine if a customer order was picked up. We show how many orders a customer has picked up.

## Use AdventureWorks2014.

## Diagram(s) of tables

We will be using the SalesOrderHeader table, SalesOrderDetail table, and the Customer table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | CustomerID |
| DerivedColumn | TotalPickedUp |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.wasPicked', N'FN') IS NOT NULL

DROP FUNCTION wasPicked;

GO

CREATE FUNCTION dbo.wasPicked(@EID nvarchar)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID = NULL )

SET @ret = 0;

ELSE SET @ret = 1;

RETURN @ret;

END;

USE AdventureWorks2014;

GO

SELECT Cust.CustomerID, SUM(dbo.wasPicked(Sod.CarrierTrackingNumber)) AS TotalPickedUp

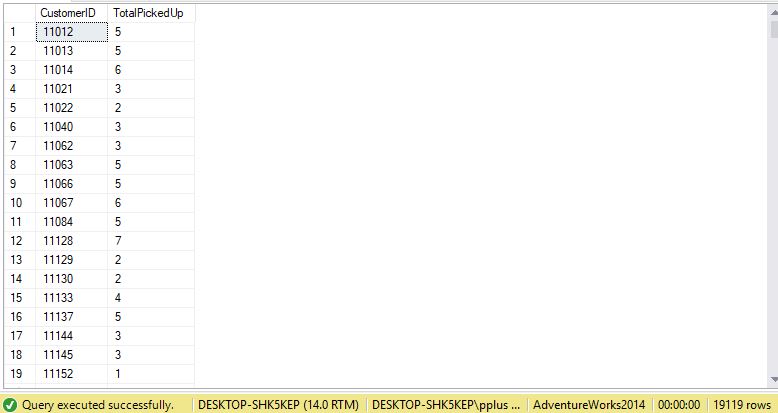
FROM Sales.Customer AS Cust

INNER JOIN Sales.SalesOrderHeader as Soh ON Cust.CustomerID = Soh.CustomerID

INNER JOIN Sales.SalesOrderDetail as Sod ON Sod.SalesOrderID = Soh.SalesOrderID

GROUP BY Cust.CustomerID

## Sample Output with (19119) rows



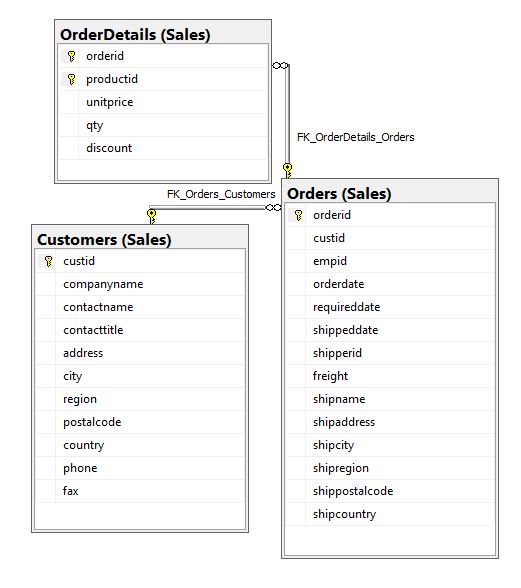
# Problem 25: Determine how much money customers have saved with discounts on orders using TSQLV4.

## What we need to solve this problem:

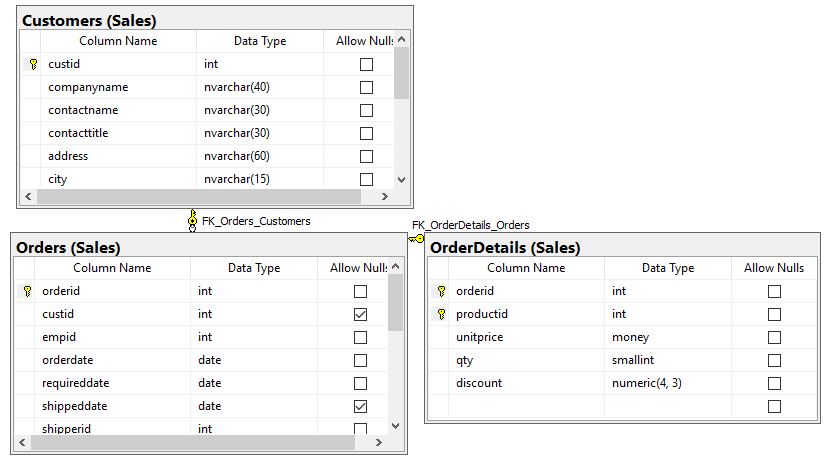
We want to show how much money customers have saved from discounts. We show all customers by custid, and use a scalar function saveMon to determine how much money has been saved. We show how all customers that have saved money, and the total they have saved. We sort results by custid.

## Use TSQLV4.

## Diagram(s) of tables

We will be using the Orders table, OrderDetails table, and the Customers table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customers | custid |
| DerivedColumn | SavedMoney |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customers | custid | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.saveMon', N'FN') IS NOT NULL

DROP FUNCTION saveMon;

GO

CREATE FUNCTION dbo.saveMon(@EID numeric(4,3), @qty smallint)

RETURNS MONEY

AS

BEGIN

DECLARE @ret AS MONEY;

IF (@EID > 0.000)

SET @ret = (@EID \* @qty);

ELSE SET @ret = 0;

RETURN @ret;

END;

USE TSQLV4;

GO

SELECT Cust.custid, SUM(dbo.saveMon(Sod.discount, sod.qty)) AS SavedMoney

FROM Sales.Customers AS Cust

INNER JOIN Sales.Orders as Soh ON Cust.custid = Soh.custid

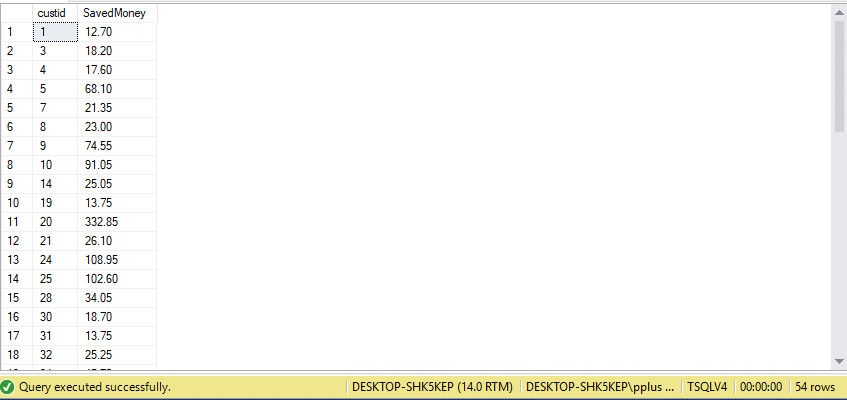
INNER JOIN Sales.OrderDetails as Sod ON Sod.orderid = Soh.orderid

GROUP BY Cust.custid

HAVING SUM(dbo.saveMon(Sod.discount, sod.qty)) > 0

ORDER BY Cust.custid;

## Sample Output with (54) rows



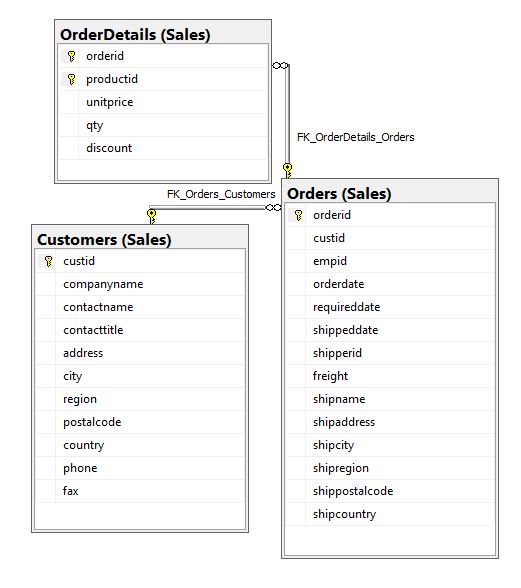
# Problem 26: Determine if a customer order was shipped late using TSQLV4.

## What we need to solve this problem:

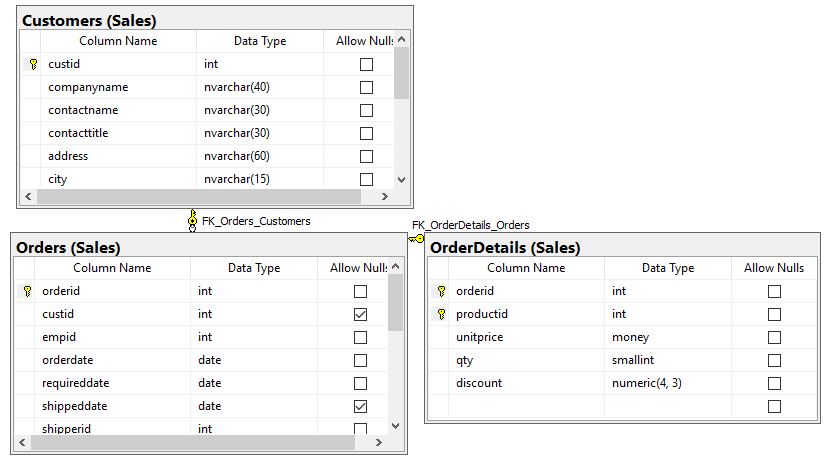
We want to show all customer orders that were shipped late. As per company policy, we apply a 0.20 discount in customers currency multiplied by quantity ordered We show all customers by custid, and use a scalar function isLate to determine if a customer order was shipped late. We show how many orders were late per customer, show current discount, determine late discount, and add current + late discount together.

## Use TSQLV4.

## Diagram(s) of tables

We will be using the Orders table, OrderDetails table, and the Customers table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customers | custid |
| DerivedColumn | ExpensiveProduct  ShippedLate  CurrentDiscount  AdditionalDiscount  NewDiscount |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customers | custid | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.isLate', N'FN') IS NOT NULL

DROP FUNCTION isLate;

GO

CREATE FUNCTION dbo.isLate(@req date, @shp date)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@shp > @req)

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE TSQLV4;

GO

SELECT Cust.custid

, SUM(dbo.isLate(Soh.requireddate, Soh.shippeddate)) AS ShippedLate

, SUM(Sod.discount) AS CurrentDiscount

, SUM((dbo.isLate(Soh.requireddate, Soh.shippeddate)) \* (Sod.qty \* 0.20)) AS AdditionalDiscount

, SUM((dbo.isLate(Soh.requireddate, Soh.shippeddate)) \* (Sod.qty \* 0.20) + (Sod.discount)) AS NewDiscount

FROM Sales.Customers AS Cust

INNER JOIN Sales.Orders as Soh ON Cust.custid = Soh.custid

INNER JOIN Sales.OrderDetails as Sod ON Sod.orderid = Soh.orderid

WHERE dbo.isLate(Soh.requireddate, Soh.shippeddate) > 0

GROUP BY Cust.custid

HAVING SUM(dbo.isLate(Soh.requireddate, Soh.shippeddate)) > 0

ORDER BY Cust.custid;

## Sample Output with (28) rows



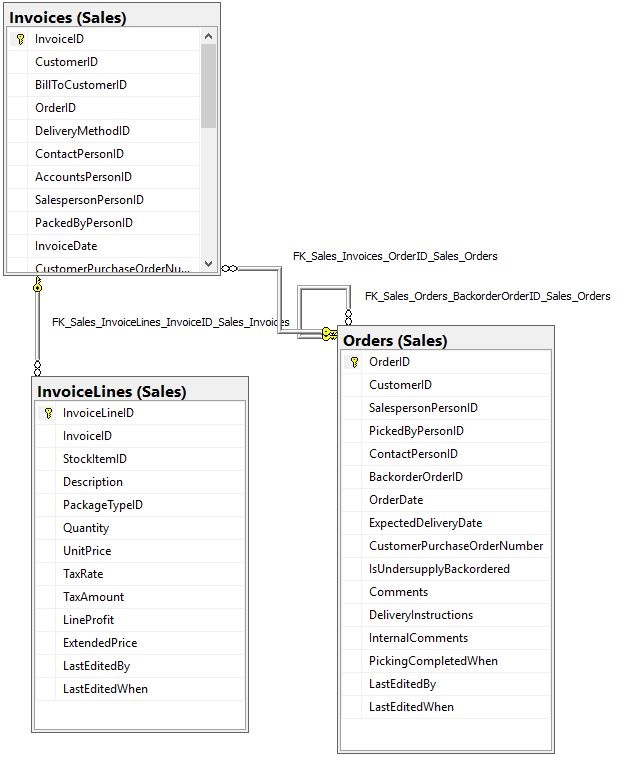
# Problem 27: Determine how many products need to be chilled using WideWorldImporters.

## What we need to solve this problem:

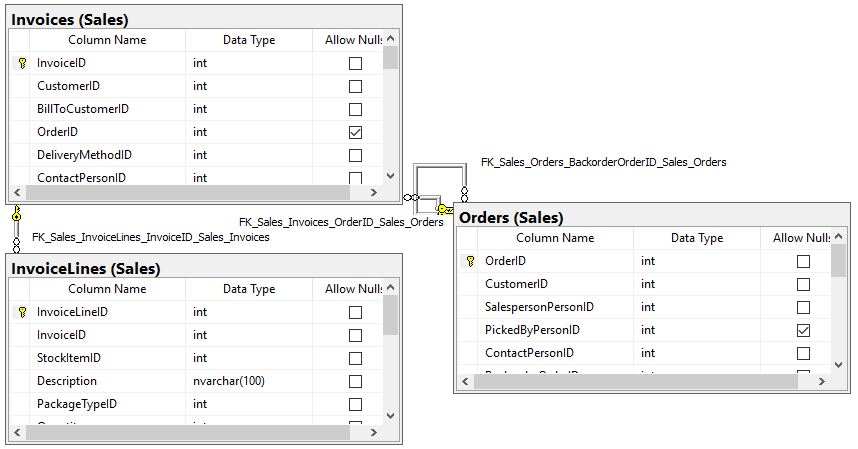
We want to show all customer orders that had items that need to be chilled. We show all orders by OrderID, and use a scalar function needChil to determine if a customer order has items that need chilling. We show how many items need chilling per order and sort by OrderID.

## Use WideWorldImporters.

## Diagram(s) of tables

We will be using the Orders table, Invoices table, and the InvoiceLines table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Orders | OrderID |
| DerivedColumn | ColdItems |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Orders | OrderID | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.needChil', N'FN') IS NOT NULL

DROP FUNCTION needChil;

GO

CREATE FUNCTION dbo.needChil(@EID int)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID > 0 )

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE WideWorldImporters;

GO

SELECT Cust.OrderID, SUM(dbo.needChil(Soh.TotalChillerItems)) AS ColdItems

FROM Sales.Orders AS Cust

INNER JOIN Sales.Invoices as Soh ON Cust.OrderID = Soh.OrderID

INNER JOIN Sales.InvoiceLines as Sod ON Sod.InvoiceID = Soh.InvoiceID

GROUP BY Cust.OrderID

HAVING SUM(dbo.needChil(Soh.TotalChillerItems)) > 0

ORDER BY Cust.OrderID;

## Sample Output with (991) rows



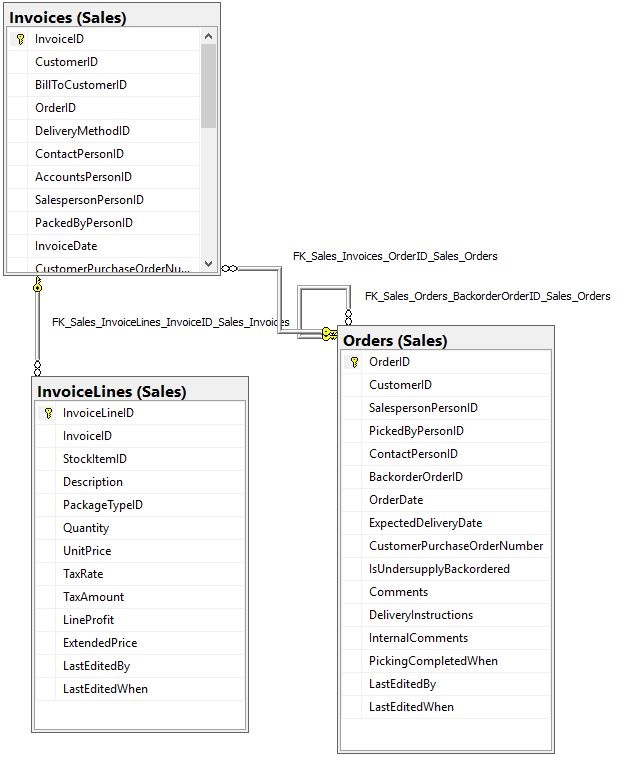
# Problem 28: Determine how many products are meant to be surprises using WideWorldImporters.

## What we need to solve this problem:

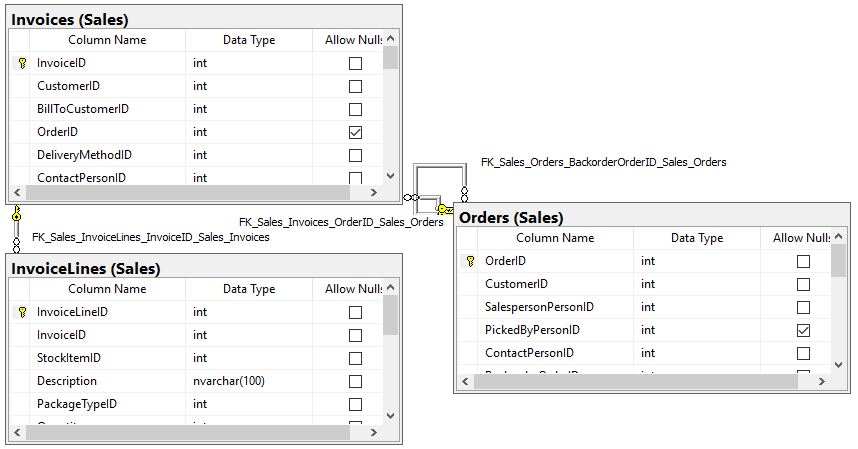
We want to show all customers that will get surprises. We show all customers by CustomerID, and use a scalar function surprise to determine if a customer will get a surprise. We show how many surprises are sent to a customer and sort by CustomerID.

## Use WideWorldImporters.

## Diagram(s) of tables

We will be using the Orders table, Invoices table, and the InvoiceLines table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Orders | CustomerID |
| DerivedColumn | SurpriseItems |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Orders | CustomerID | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.surprise', N'FN') IS NOT NULL

DROP FUNCTION surprise;

GO

CREATE FUNCTION dbo.surprise(@EID int)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID IS NULL )

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE WideWorldImporters;

GO

SELECT Soh.CustomerID, SUM(dbo.surprise(Cust.PickedByPersonID)) AS SurpriseItems

FROM Sales.Orders AS Cust

INNER JOIN Sales.Invoices as Soh ON Cust.OrderID = Soh.OrderID

INNER JOIN Sales.InvoiceLines as Sod ON Sod.InvoiceID = Soh.InvoiceID

GROUP BY Soh.CustomerID

ORDER BY Soh.CustomerID;

## Sample Output with (663) rows



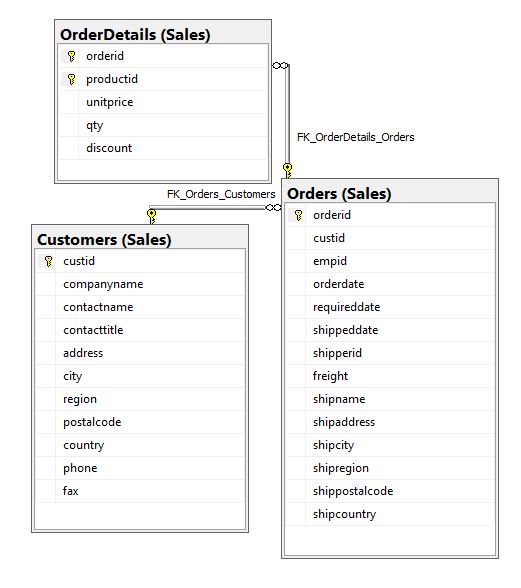
# Problem 29: Determine how many customer orders use shipper three using TSQLV4.

## What we need to solve this problem:

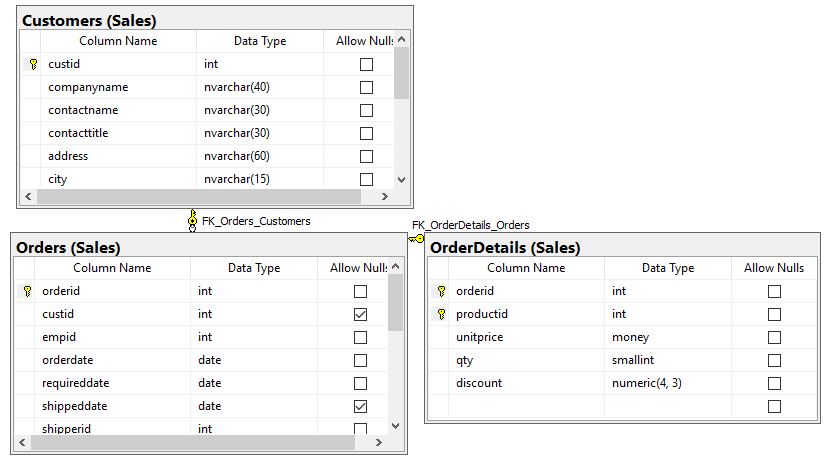
We want to show all customer orders that were shipped with shipper 3. We show all customers by custid, and use a scalar function shipThr to determine if a customer order used shipper 3. We show how many orders a customer had shipped by shipper 3, and sort by custid.

## Use TSQLV4.

## Diagram(s) of tables

We will be using the Orders table, OrderDetails table, and the Customers table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customers | custid |
| DerivedColumn | ShipperThree |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customers | custid | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.shipThr', N'FN') IS NOT NULL

DROP FUNCTION shipThr;

GO

CREATE FUNCTION dbo.shipThr(@EID int)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID = 3)

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE TSQLV4;

GO

SELECT Cust.custid, SUM(dbo.shipThr(Soh.shipperid)) AS ShipperThree

FROM Sales.Customers AS Cust

INNER JOIN Sales.Orders as Soh ON Cust.custid = Soh.custid

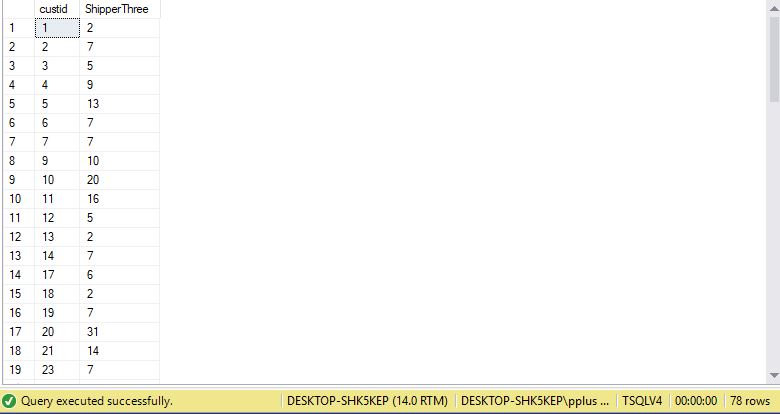
INNER JOIN Sales.OrderDetails as Sod ON Sod.orderid = Soh.orderid

GROUP BY Cust.custid

HAVING SUM(dbo.shipThr(Soh.shipperid)) > 0

ORDER BY Cust.custid;

## Sample Output with (78) rows



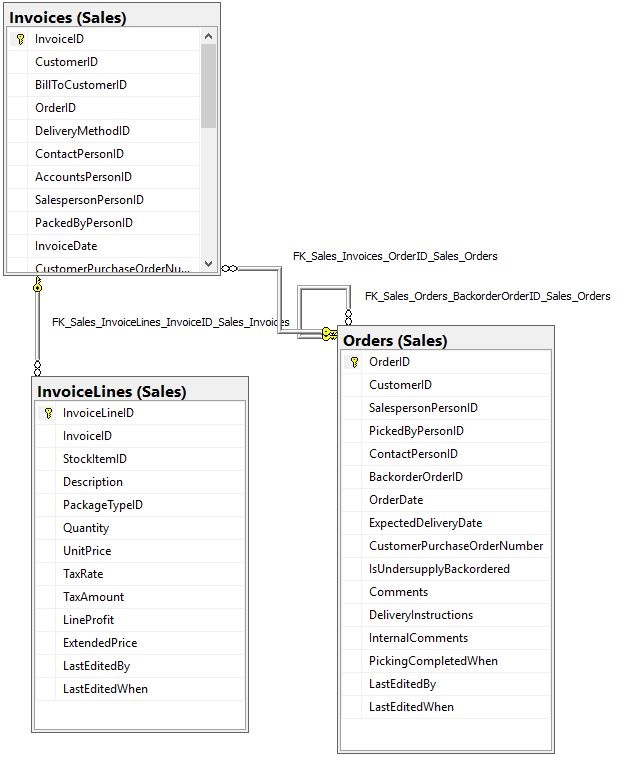
# Problem 30: Determine how many products were packed by person 14 using WideWorldImporters.

## What we need to solve this problem:

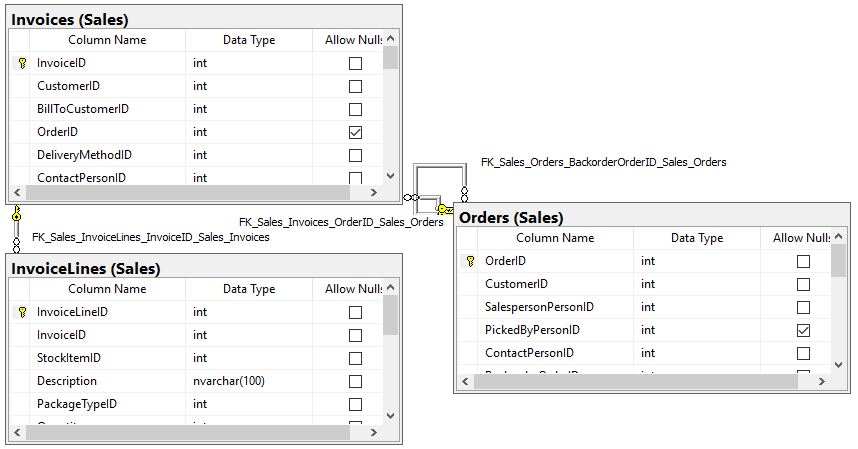
We want to show all customer order that were packed by person 14. We show all orders by OrderID, and use a scalar function packed to determine if a customer order was packed by person 14. We show how many items were packed by person 14 in an order, and sort by OrderID .

## Use WideWorldImporters.

## Diagram(s) of tables

We will be using the Orders table, Invoices table, and the InvoiceLines table.

## Columns from Standard view



## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Orders | OrderID |
| DerivedColumn | PackedBy14 |

## Order By

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Orders | OrderID | ASC |

## Query to solve this problem:

IF OBJECT\_ID (N'dbo.packed', N'FN') IS NOT NULL

DROP FUNCTION packed;

GO

CREATE FUNCTION dbo.packed(@EID int)

RETURNS int

AS

BEGIN

DECLARE @ret AS int;

IF (@EID = 14 )

SET @ret = 1;

ELSE SET @ret = 0;

RETURN @ret;

END;

USE WideWorldImporters;

GO

SELECT Cust.OrderID, SUM(dbo.packed(Soh.PackedByPersonID)) AS PackedBy14

FROM Sales.Orders AS Cust

INNER JOIN Sales.Invoices as Soh ON Cust.OrderID = Soh.OrderID

INNER JOIN Sales.InvoiceLines as Sod ON Sod.InvoiceID = Soh.InvoiceID

GROUP BY Cust.OrderID

ORDER BY Cust.OrderID;

## Sample Output with (70510) rows

