Individual Queries

CSCI 331 Database

Student: RYAN MOHAMED

Teacher: PETER HELLER
Group: 2

y: Ryan Mohamed **Date Prepared: 10/12**

BEST

SIMPLE

Example of Order sub-system in NorthWinds2022TSQLV7

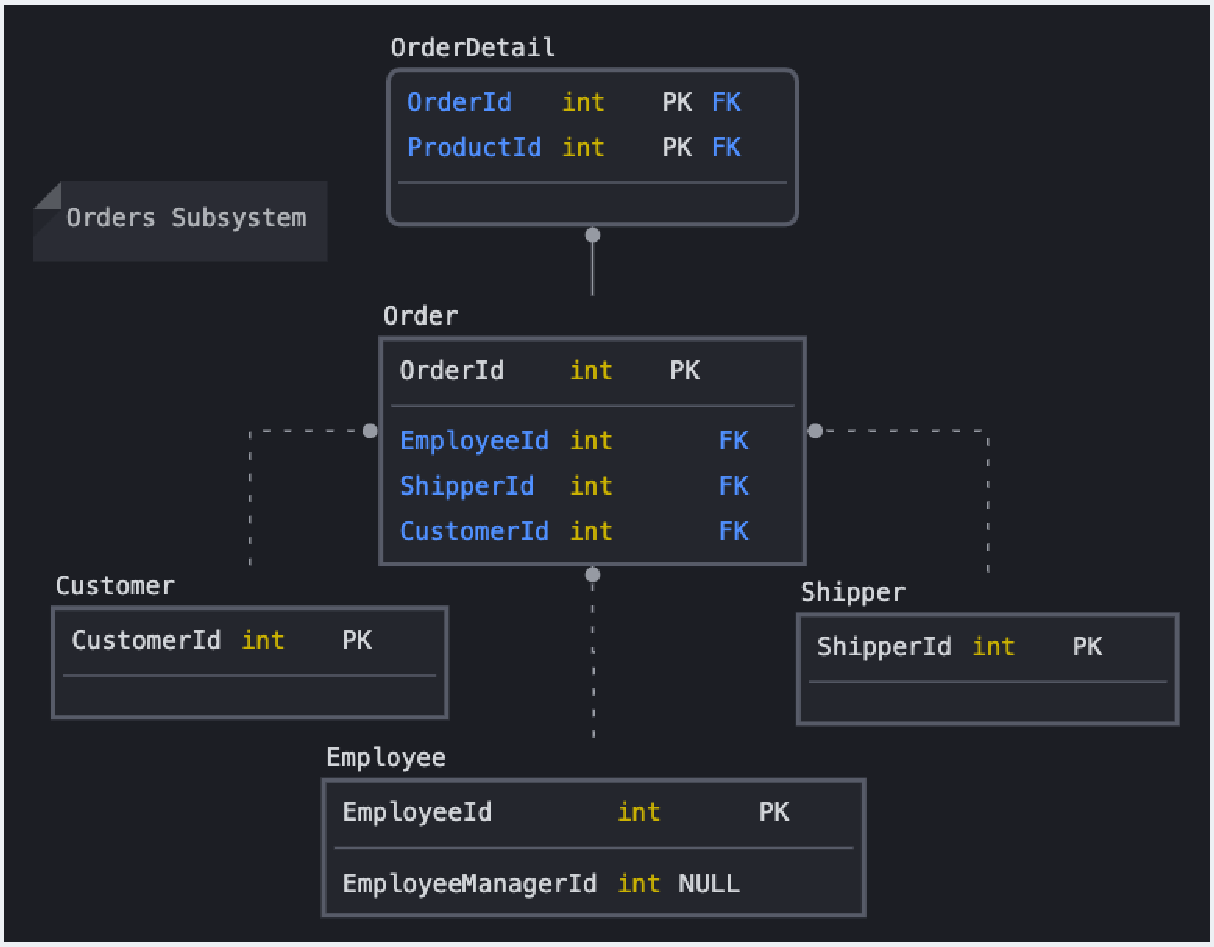
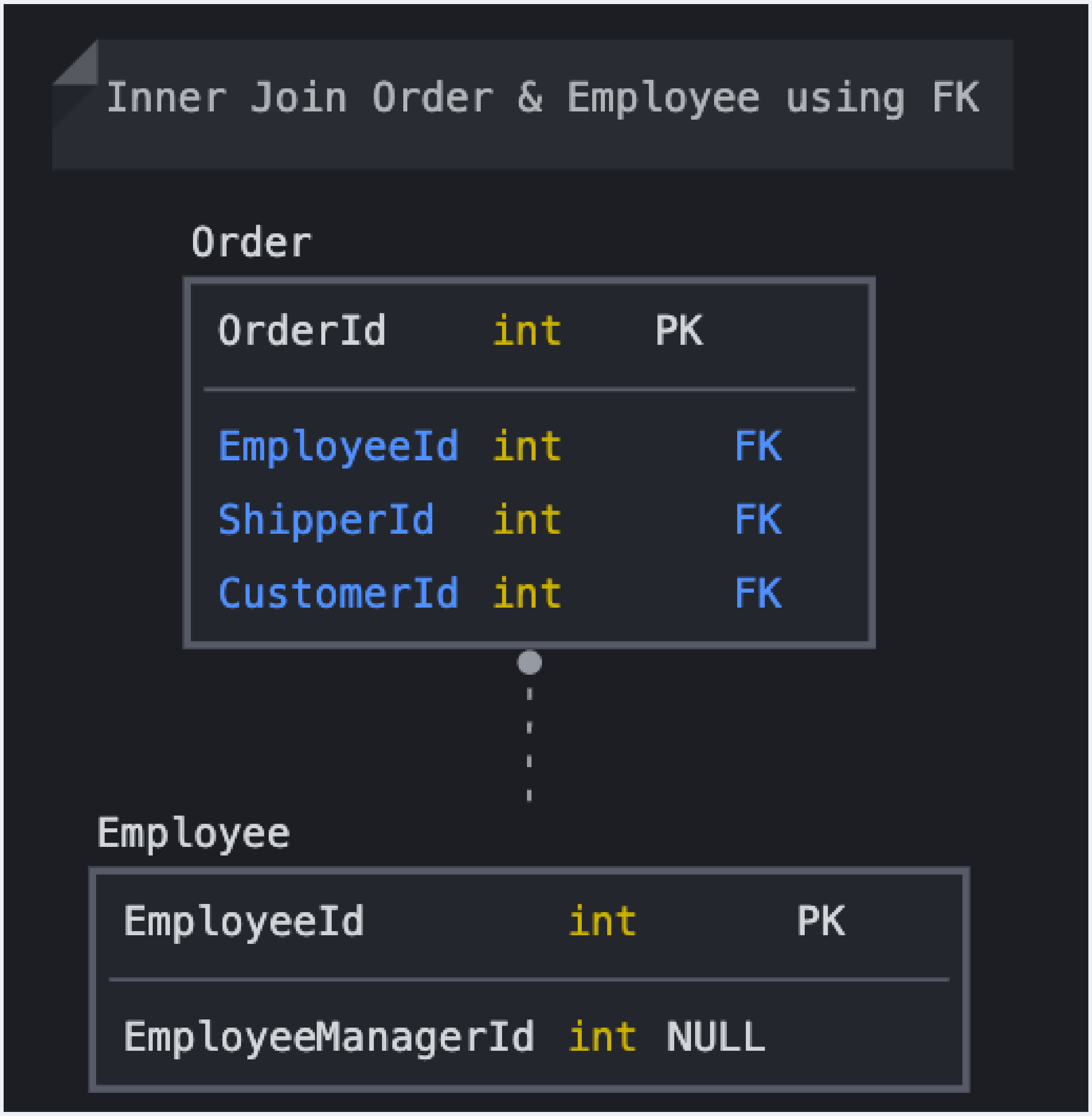
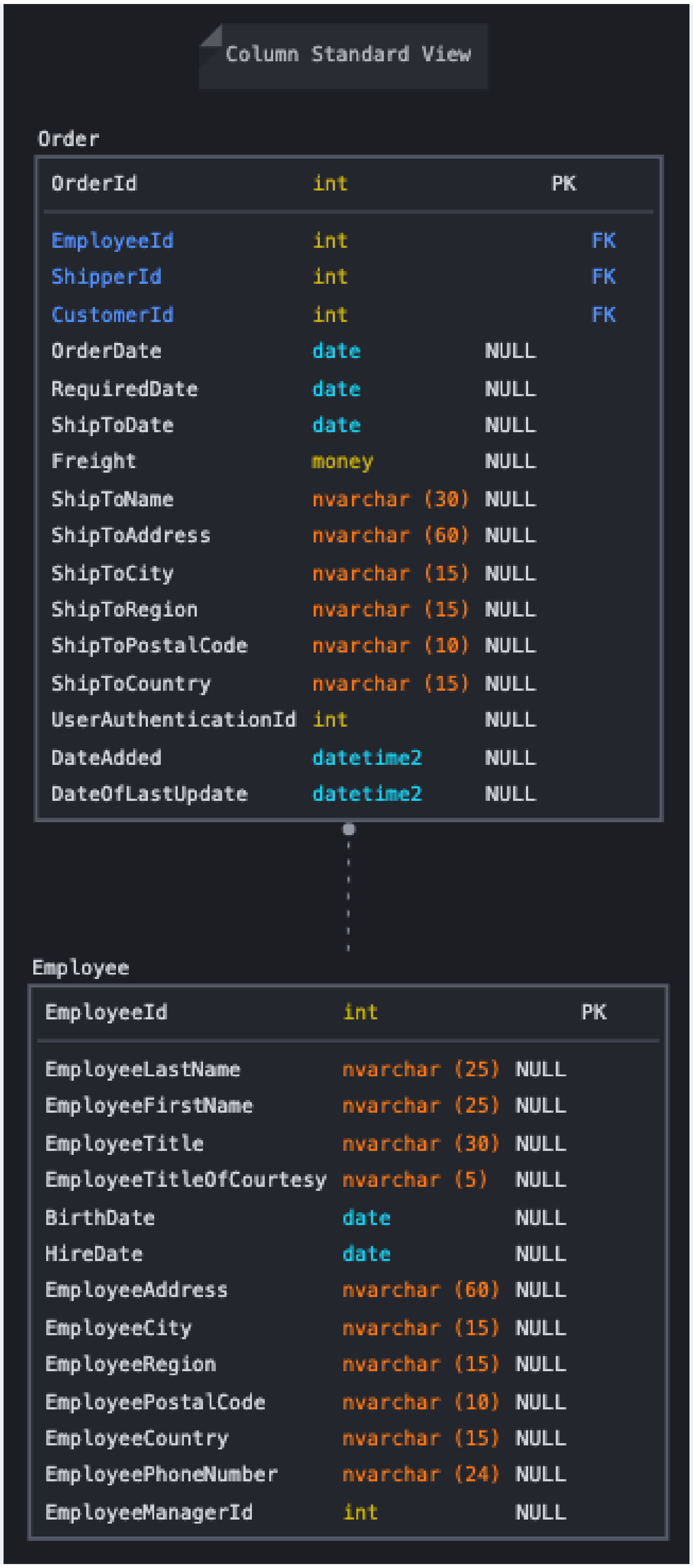


Diagram of Tables



Columns from Standard View


Column Standard View

Order

OrderId	int		PK
EmployeeId	int		FK
ShipperId	int		FK
CustomerId	int		FK
OrderDate	date	NULL	
RequiredDate	date	NULL	
ShipToDate	date	NULL	
Freight	money	NULL	
ShipToName	nvarchar (30)	NULL	
ShipToAddress	nvarchar (60)	NULL	
ShipToCity	nvarchar (15)	NULL	
ShipToRegion	nvarchar (15)	NULL	
ShipToPostalCode	nvarchar (10)	NULL	
ShipToCountry	nvarchar (15)	NULL	
UserAuthenticationId	int	NULL	
DateAdded	datetime2	NULL	
DateOfLastUpdate	datetime2	NULL	

Employee

EmployeeId	int		PK
EmployeeLastName	nvarchar (25)	NULL	
EmployeeFirstName	nvarchar (25)	NULL	
EmployeeTitle	nvarchar (30)	NULL	
EmployeeTitleOfCourtesy	nvarchar (5)	NULL	
BirthDate	date	NULL	
HireDate	date	NULL	
EmployeeAddress	nvarchar (60)	NULL	
EmployeeCity	nvarchar (15)	NULL	
EmployeeRegion	nvarchar (15)	NULL	
EmployeePostalCode	nvarchar (10)	NULL	
EmployeeCountry	nvarchar (15)	NULL	
EmployeePhoneNumber	nvarchar (24)	NULL	
EmployeeManagerId	int	NULL	

Proposition 01: Return the employee title responsible for each order, ordered by freight price.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

Each unique order in the order table is designated by its OrderId, with more information about the order such as the EmployeeId responsible. Accessing information about any Employee can be done by retrieving the row with the desired EmployeeId from the Employee table. EmployeeId should act as a link between the two tables to provide extra information (EmployeeTitle) for the order.

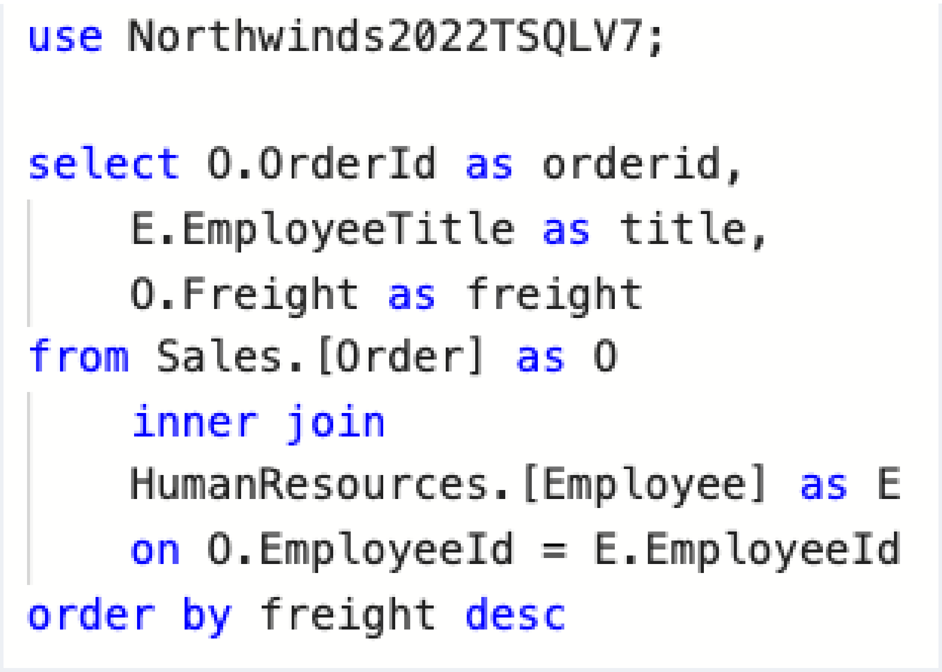
Project following columns from their respective tables in the select clause

Table Name	Column Name
Sales.Order	OrderId Freight
HumanResources.Employee	EmployeeTitle

Order By

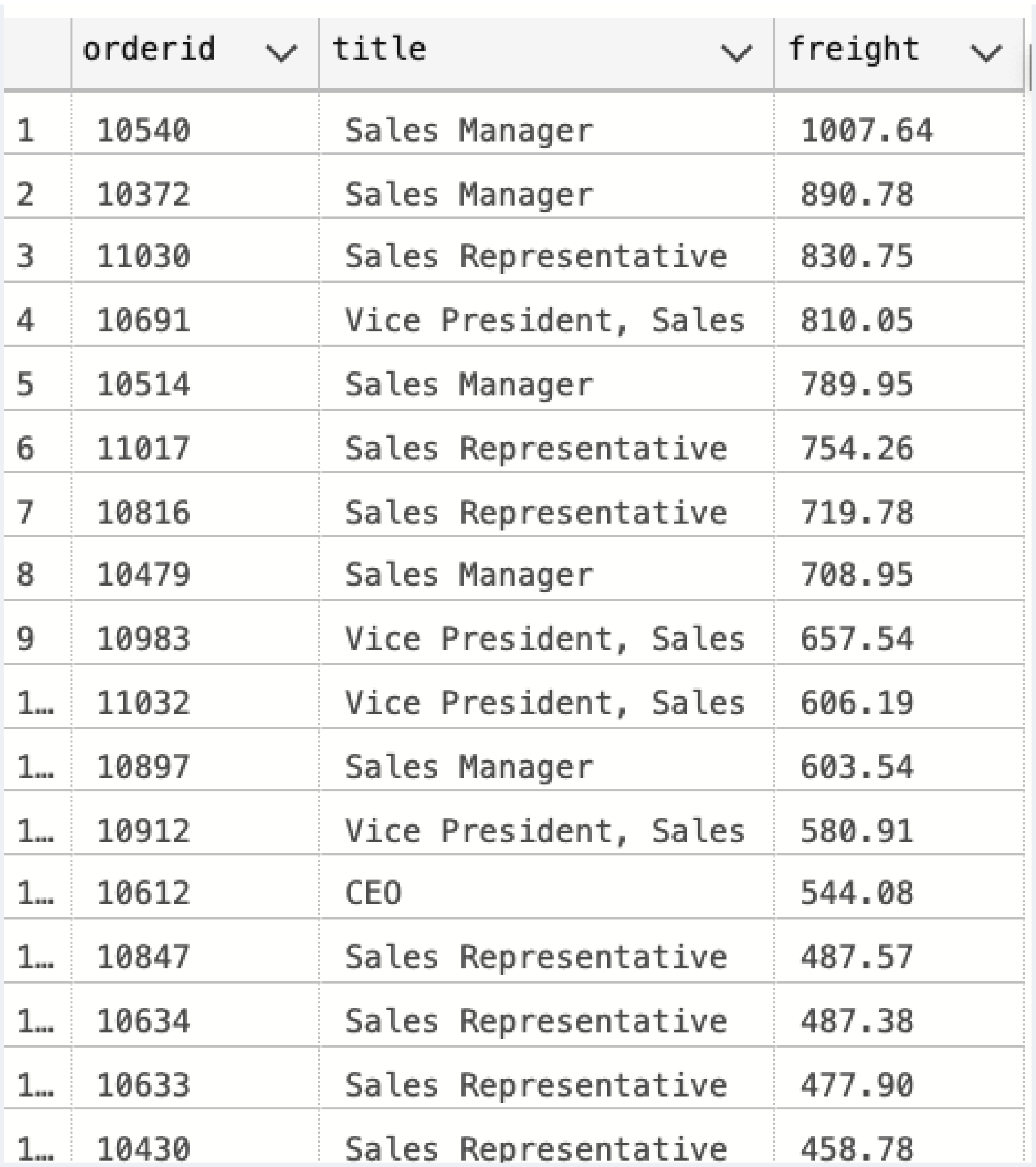
Table Name	Column Name	Sort Order
Sales.Order	Freight	DESC

Problem Solving Query



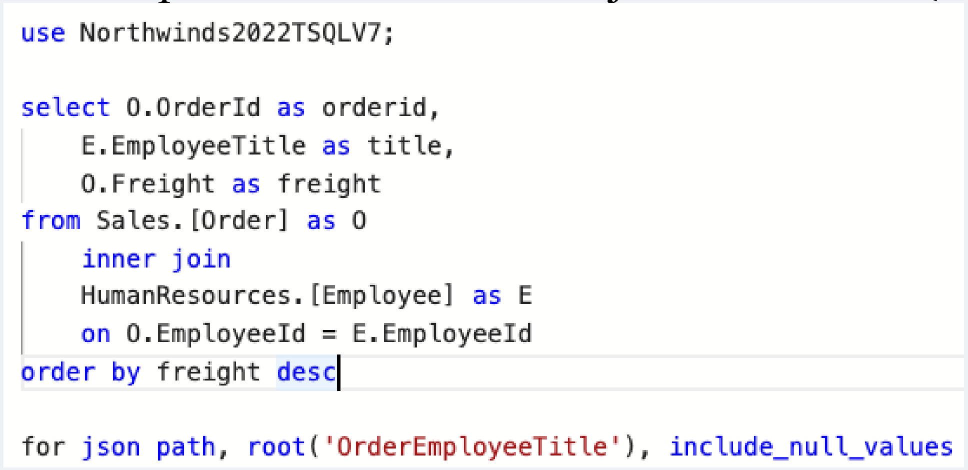
```sql
use Northwinds2022TSQLV7;

select O.OrderId as orderid,
    E.EmployeeTitle as title,
    O.Freight as freight
from Sales.[Order] as O
    inner join
    HumanResources.[Employee] as E
    on O.EmployeeId = E.EmployeeId
order by freight desc
```

Sample Relational Output with total number of rows returned (830)

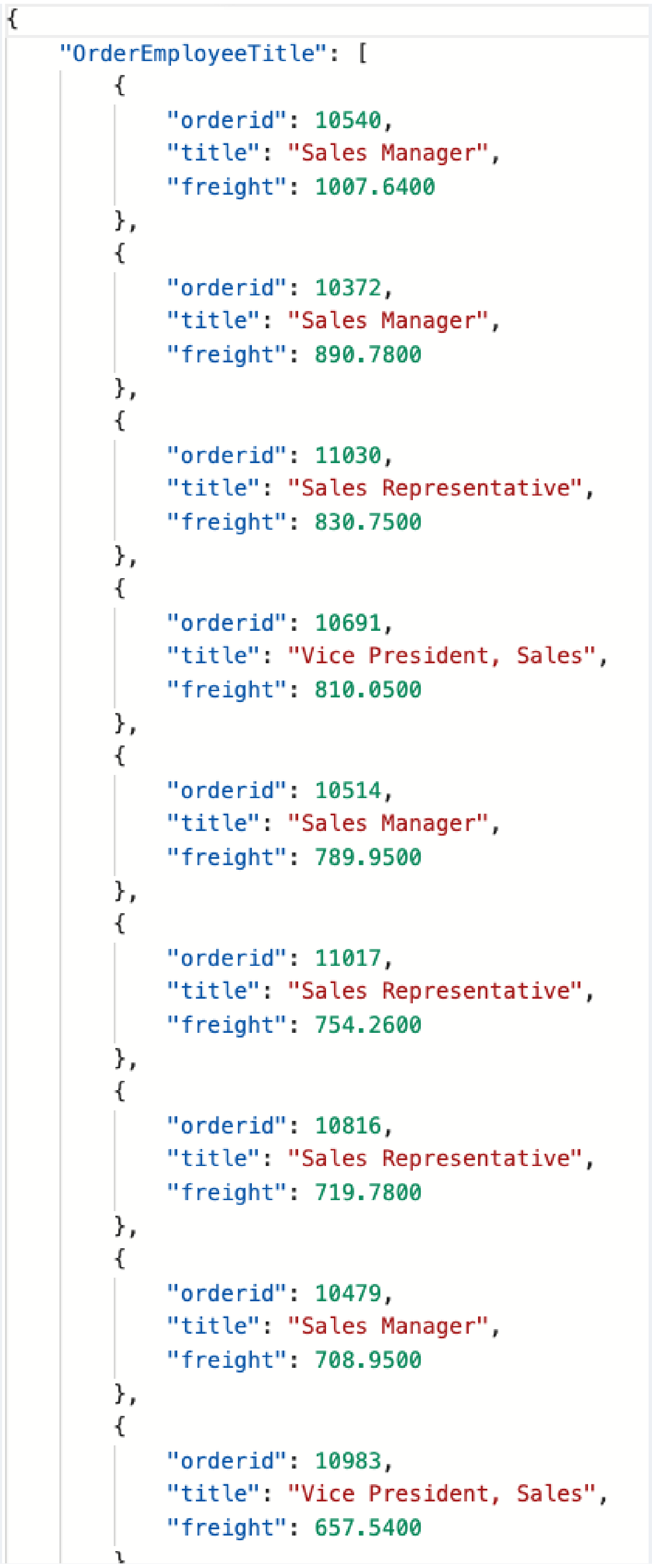


	orderid	title	freight
1	10540	Sales Manager	1007.64
2	10372	Sales Manager	890.78
3	11030	Sales Representative	830.75
4	10691	Vice President, Sales	810.05
5	10514	Sales Manager	789.95
6	11017	Sales Representative	754.26
7	10816	Sales Representative	719.78
8	10479	Sales Manager	708.95
9	10983	Vice President, Sales	657.54
1…	11032	Vice President, Sales	606.19
1…	10897	Sales Manager	603.54
1…	10912	Vice President, Sales	580.91
1…	10612	CEO	544.08
1…	10847	Sales Representative	487.57
1…	10634	Sales Representative	487.38
1…	10633	Sales Representative	477.90
1…	10430	Sales Representative	458.78

Sample JSON Output with total number of rows returned (830)



```
use Northwinds2022TSQLV7;

select O.OrderId as orderid,
    E.EmployeeTitle as title,
    O.Freight as freight
from Sales.[Order] as O
    inner join
    HumanResources.[Employee] as E
    on O.EmployeeId = E.EmployeeId
order by freight desc

for json path, root('OrderEmployeeTitle'), include_null_values
```



```
{
    "OrderEmployeeTitle": [
        {
            "orderid": 10540,
            "title": "Sales Manager",
            "freight": 1007.6400
        },
        {
            "orderid": 10372,
            "title": "Sales Manager",
            "freight": 890.7800
        },
        {
            "orderid": 11030,
            "title": "Sales Representative",
            "freight": 830.7500
        },
        {
            "orderid": 10691,
            "title": "Vice President, Sales",
            "freight": 810.0500
        },
        {
            "orderid": 10514,
            "title": "Sales Manager",
            "freight": 789.9500
        },
        {
            "orderid": 11017,
            "title": "Sales Representative",
            "freight": 754.2600
        },
        {
            "orderid": 10816,
            "title": "Sales Representative",
            "freight": 719.7800
        },
        {
            "orderid": 10479,
            "title": "Sales Manager",
            "freight": 708.9500
        },
        {
            "orderid": 10983,
            "title": "Vice President, Sales",
            "freight": 657.5400
```

MEDIUM

Example of OrderDetail sub-system in NorthWinds2022TSQLV7

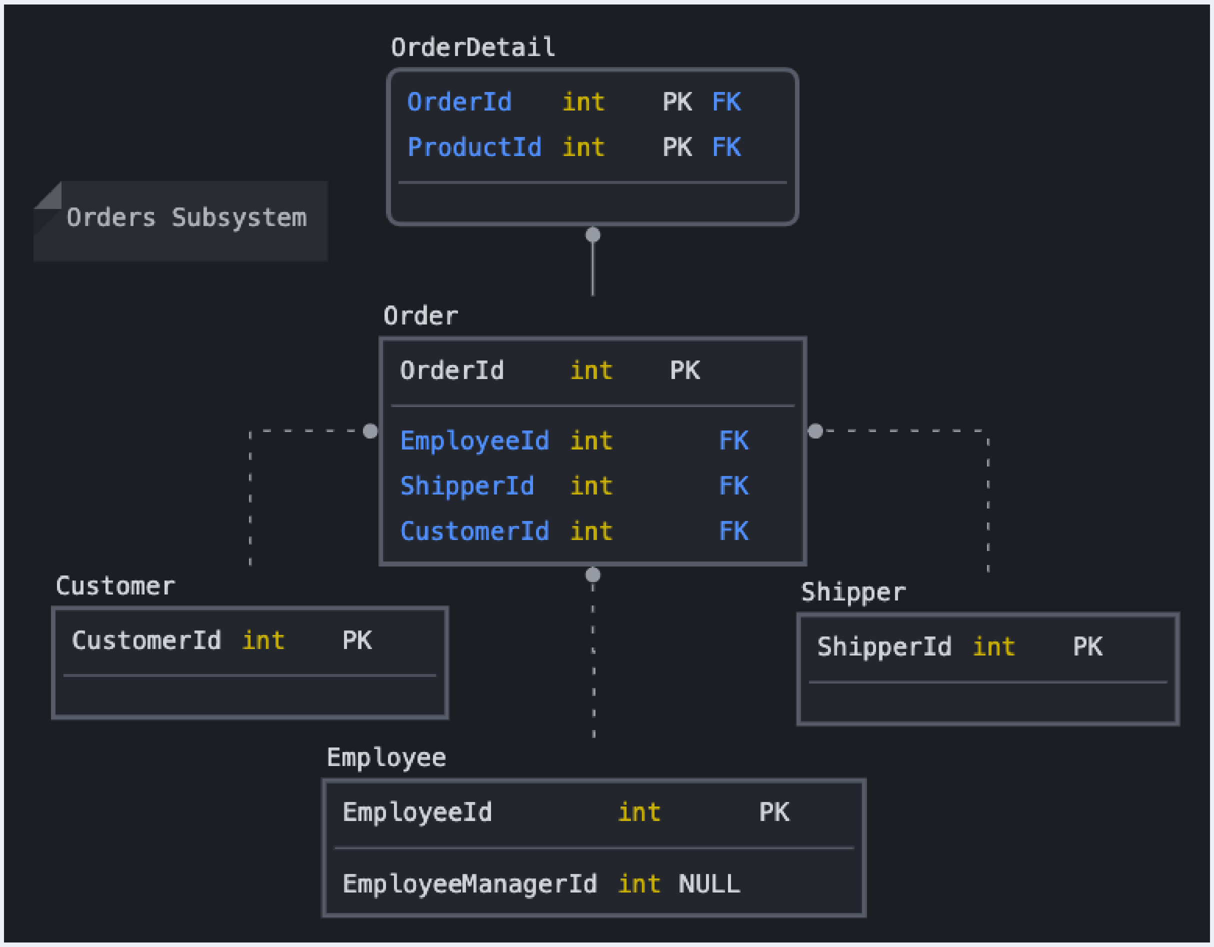
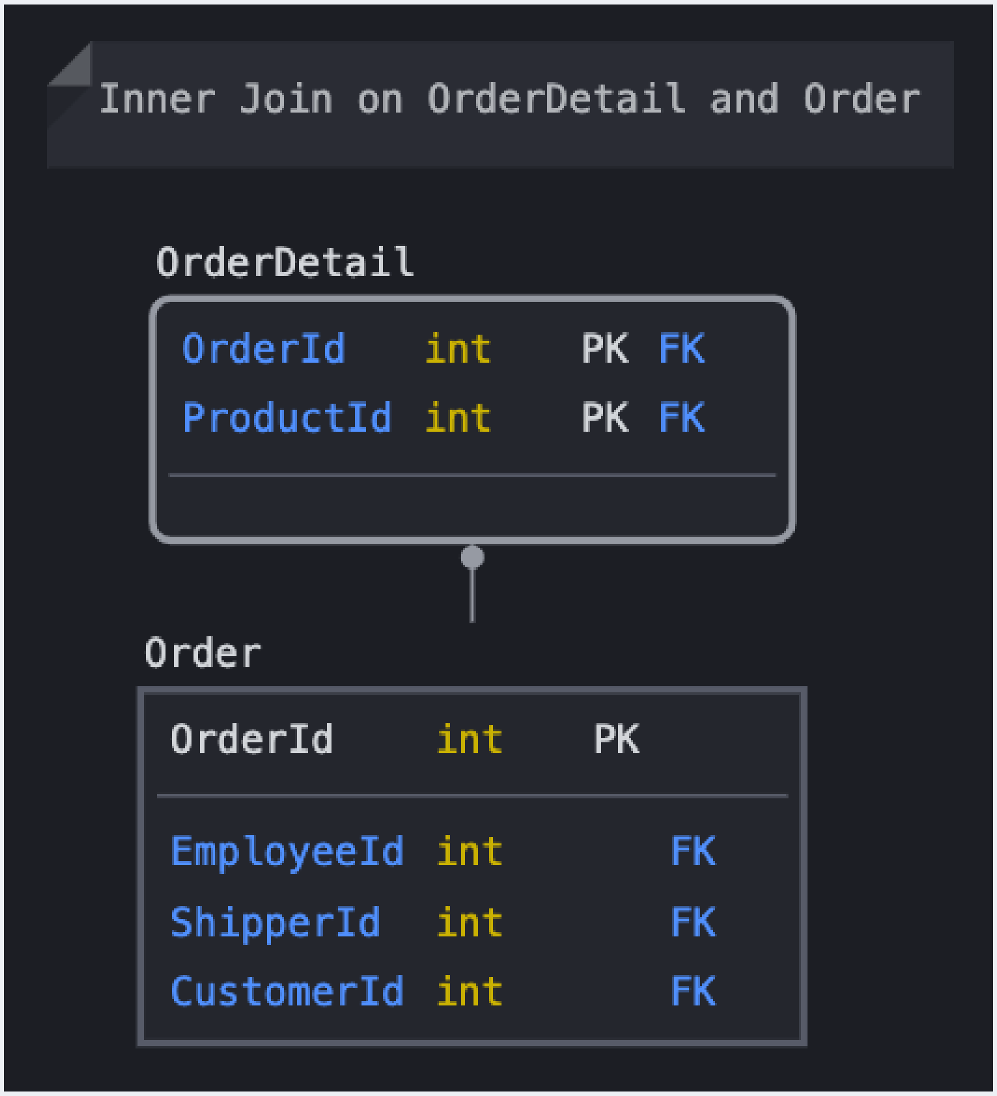
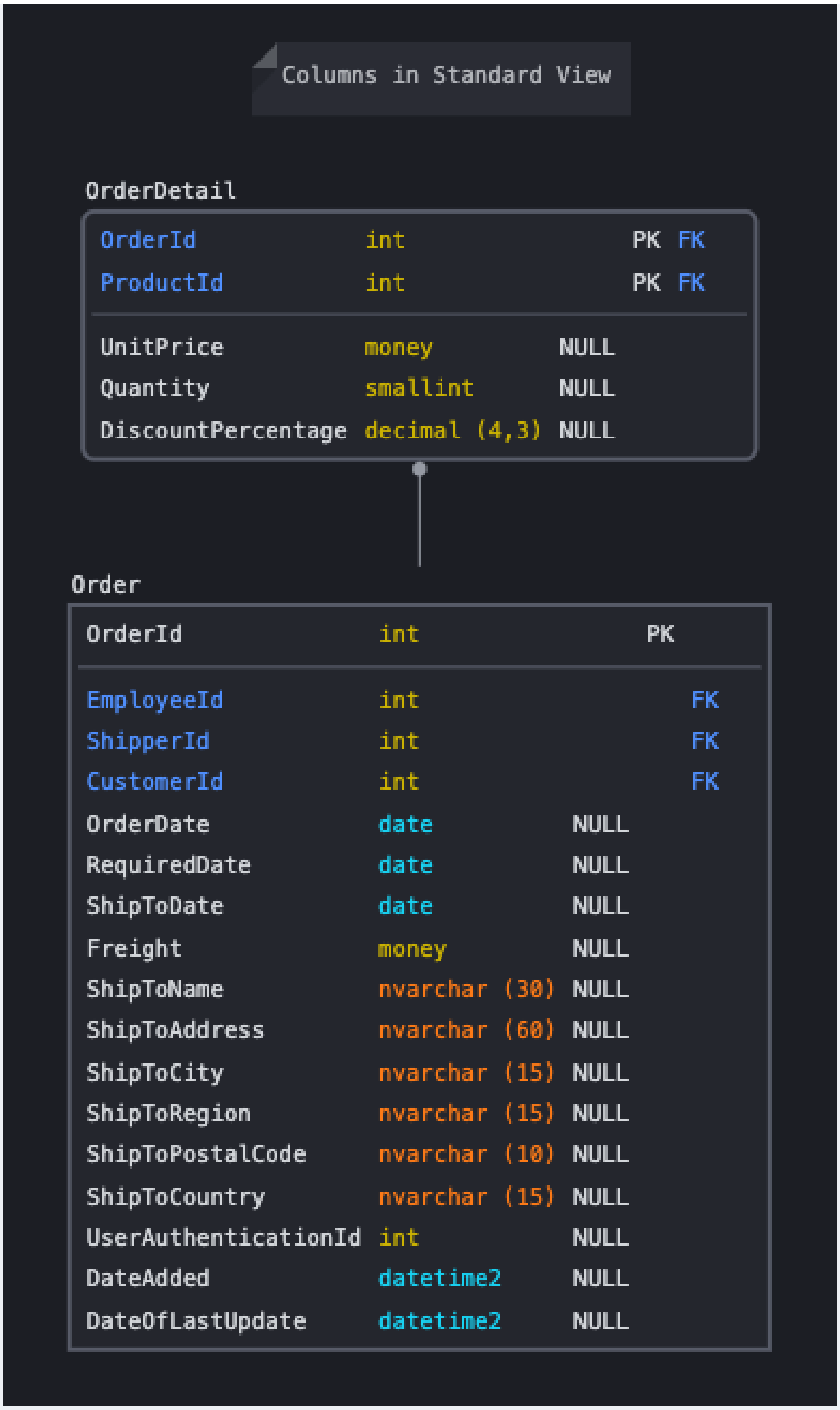


Diagram of Tables



Columns from Standard View



OrderDetail

OrderId	int		PK FK
ProductId	int		PK FK
UnitPrice	money	NULL	
Quantity	smallint	NULL	
DiscountPercentage	decimal (4,3)	NULL	

Order

OrderId	int		PK
EmployeeId	int		FK
ShipperId	int		FK
CustomerId	int		FK
OrderDate	date	NULL	
RequiredDate	date	NULL	
ShipToDate	date	NULL	
Freight	money	NULL	
ShipToName	nvarchar (30)	NULL	
ShipToAddress	nvarchar (60)	NULL	
ShipToCity	nvarchar (15)	NULL	
ShipToRegion	nvarchar (15)	NULL	
ShipToPostalCode	nvarchar (10)	NULL	
ShipToCountry	nvarchar (15)	NULL	
UserAuthenticationId	int	NULL	
DateAdded	datetime2	NULL	
DateOfLastUpdate	datetime2	NULL	

Proposition 02: Get the total price (w/ discount) for each order going to the UK.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

The total price for one product in an order can be calculated using the Unit Price, Quantity, and Discount Percentage. An order can consist of multiple products, each contributing to the total price of the order. Find the total for individual orders, then sum the totals for that specific order.

Project following columns from their respective tables in the select clause

Table Name	Column Name
Sales.Order	OrderId
UKOrder	orderid
Sales.OrderDetail	DiscountPercentage UnitPrice Quantity
UKProductTotal	total

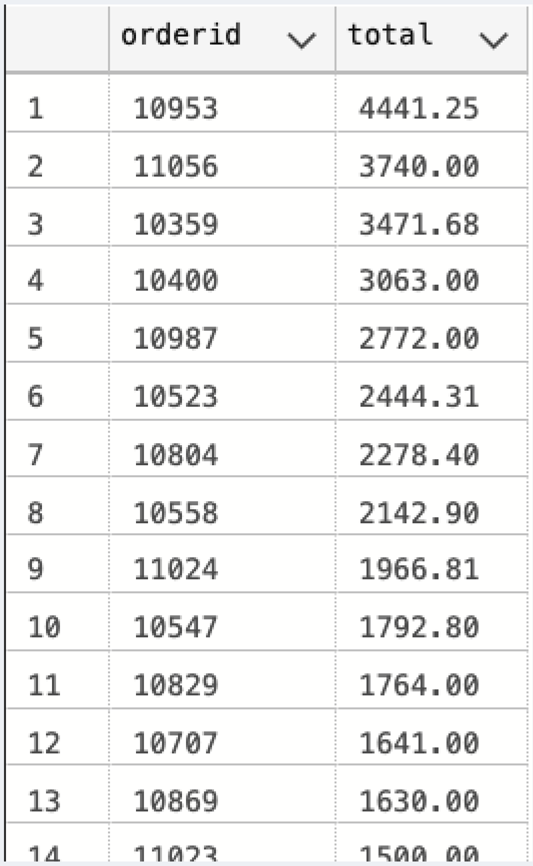
Order By

Table Name	Column Name	Sort Order
UKProductTotal	total	DESC

Problem Solving Query

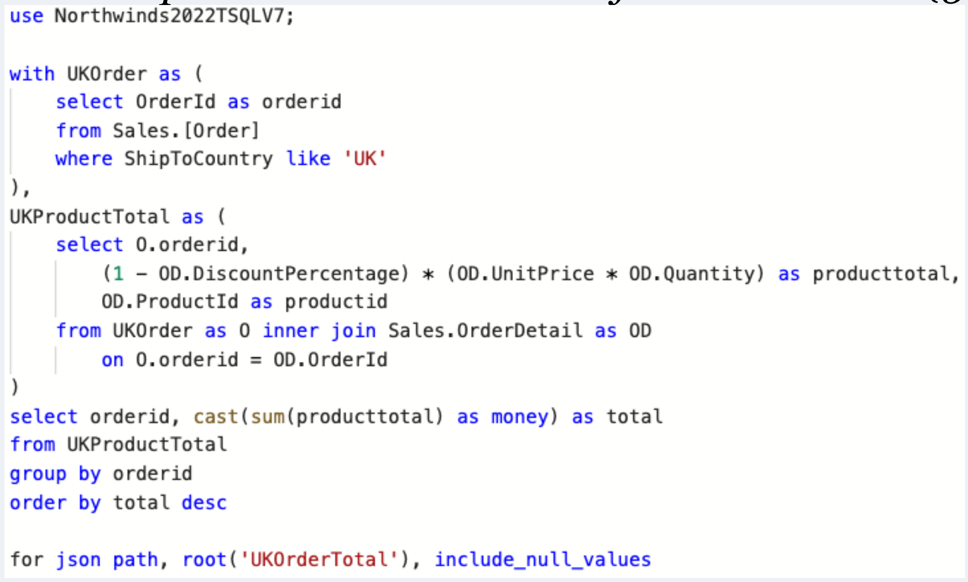
```
use Northwinds2022TSQLV7;

with UKOrder as (
    select OrderId as orderid
    from Sales.[Order]
    where ShipToCountry like 'UK'
),
UKProductTotal as (
    select O.orderid,
        (1 - OD.DiscountPercentage) * (OD.UnitPrice * OD.Quantity) as producttotal,
        OD.ProductId as productid
    from UKOrder as O inner join Sales.OrderDetail as OD
        on O.orderid = OD.OrderId
)
select orderid, cast(sum(producttotal) as money) as total
from UKProductTotal
group by orderid
order by total desc
```

Sample Relational Output with total number of rows returned (56)

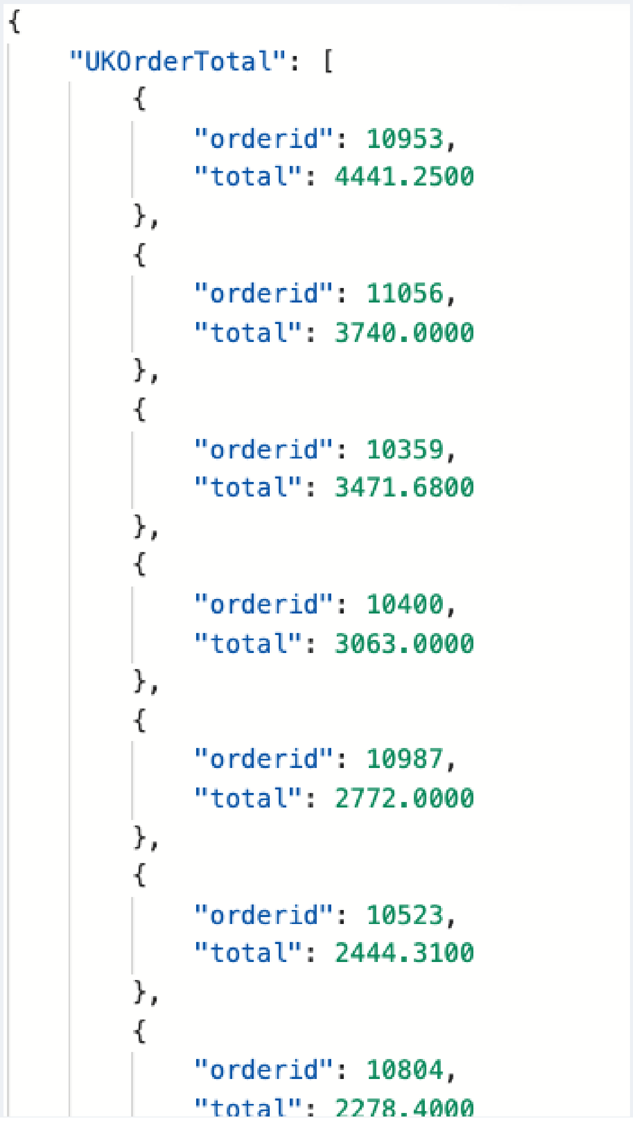


	orderid	total
1	10953	4441.25
2	11056	3740.00
3	10359	3471.68
4	10400	3063.00
5	10987	2772.00
6	10523	2444.31
7	10804	2278.40
8	10558	2142.90
9	11024	1966.81
10	10547	1792.80
11	10829	1764.00
12	10707	1641.00
13	10869	1630.00
14	11023	1500.00

Sample JSON Output with total number of rows returned (56)



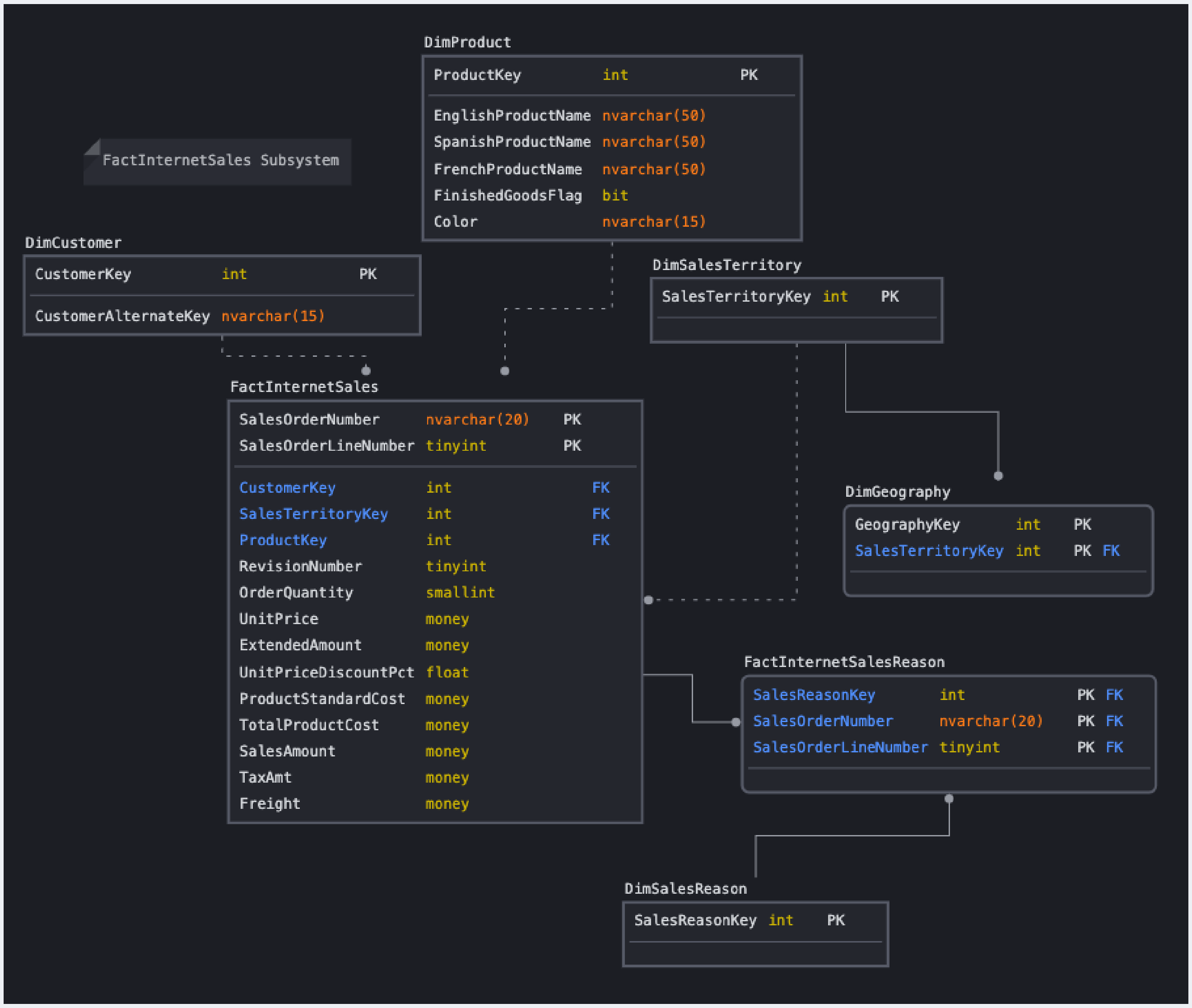
```
use Northwinds2022TSQLV7;

with UKOrder as (
    select OrderId as orderid
    from Sales.[Order]
    where ShipToCountry like 'UK'
),
UKProductTotal as (
    select O.orderid,
        (1 - OD.DiscountPercentage) * (OD.UnitPrice * OD.Quantity) as producttotal,
        OD.ProductId as productid
    from UKOrder as O inner join Sales.OrderDetail as OD
        on O.orderid = OD.OrderId
)
select orderid, cast(sum(producttotal) as money) as total
from UKProductTotal
group by orderid
order by total desc

for json path, root('UKOrderTotal'), include_null_values
```



```
{
    "UKOrderTotal": [
        {
            "orderid": 10953,
            "total": 4441.2500
        },
        {
            "orderid": 11056,
            "total": 3740.0000
        },
        {
            "orderid": 10359,
            "total": 3471.6800
        },
        {
            "orderid": 10400,
            "total": 3063.0000
        },
        {
            "orderid": 10987,
            "total": 2772.0000
        },
        {
            "orderid": 10523,
            "total": 2444.3100
        },
        {
            "orderid": 10804,
            "total": 2278.4000
```

COMPLEX

Example of FactInternetSales sub-system in AdventureWorksDW2017



DimProduct

ProductKey	int	PK
EnglishProductName	nvarchar(50)	
SpanishProductName	nvarchar(50)	
FrenchProductName	nvarchar(50)	
FinishedGoodsFlag	bit	
Color	nvarchar(15)	

DimCustomer

CustomerKey	int	PK
CustomerAlternateKey	nvarchar(15)	

DimSalesTerritory

SalesTerritoryKey	int	PK

FactInternetSales

SalesOrderNumber	nvarchar(20)	PK
SalesOrderLineNumber	tinyint	PK
CustomerKey	int	FK
SalesTerritoryKey	int	FK
ProductKey	int	FK
RevisionNumber	tinyint	
OrderQuantity	smallint	
UnitPrice	money	
ExtendedAmount	money	
UnitPriceDiscountPct	float	
ProductStandardCost	money	
TotalProductCost	money	
SalesAmount	money	
TaxAmt	money	
Freight	money	

DimGeography

GeographyKey	int	PK
SalesTerritoryKey	int	PK FK

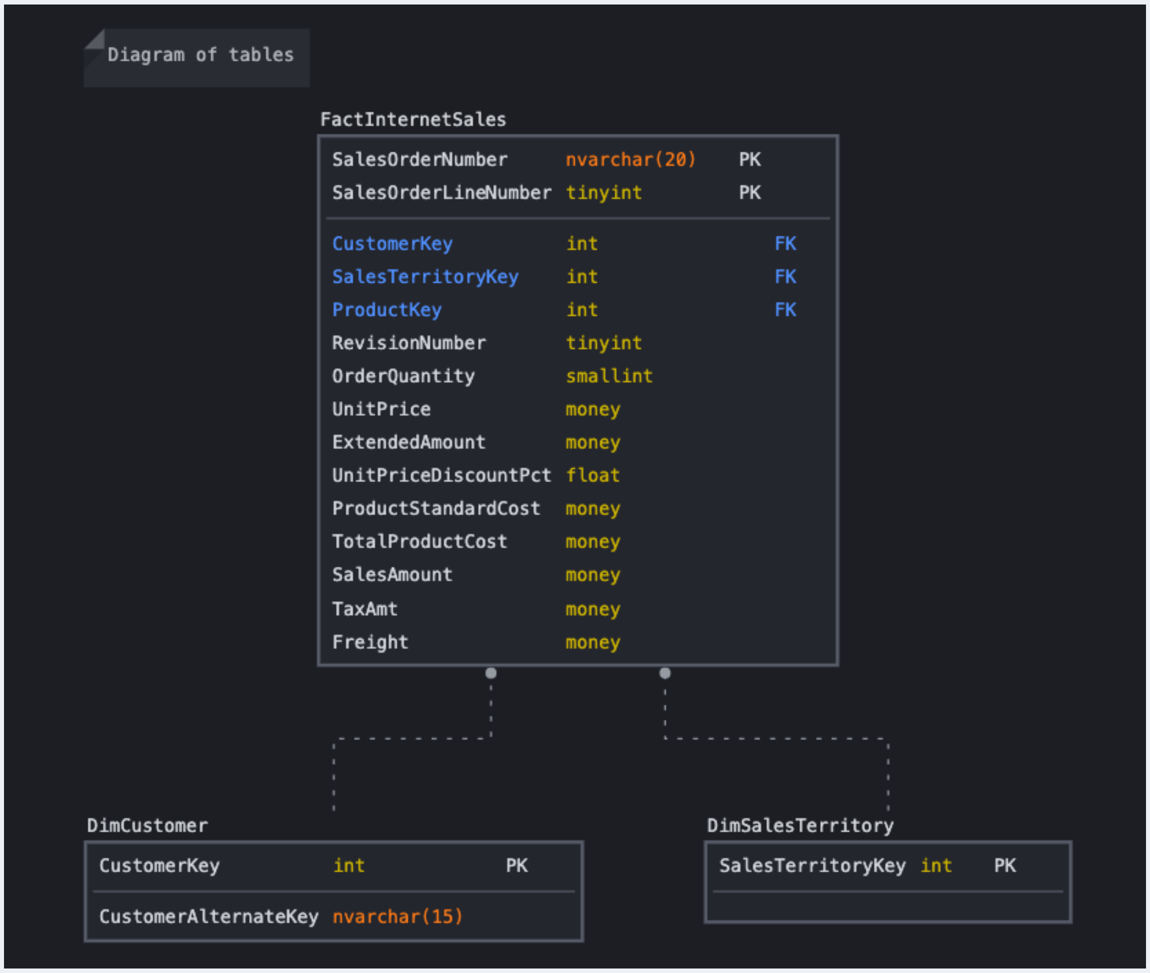
FactInternetSalesReason

SalesReasonKey	int	PK FK
SalesOrderNumber	nvarchar(20)	PK FK
SalesOrderLineNumber	tinyint	PK FK

DimSalesReason

SalesReasonKey	int	PK

Diagram of Tables



FactInternetSales

SalesOrderNumber	nvarchar(20)	PK
SalesOrderLineNumber	tinyint	PK
CustomerKey	int	FK
SalesTerritoryKey	int	FK
ProductKey	int	FK
RevisionNumber	tinyint	
OrderQuantity	smallint	
UnitPrice	money	
ExtendedAmount	money	
UnitPriceDiscountPct	float	
ProductStandardCost	money	
TotalProductCost	money	
SalesAmount	money	
TaxAmt	money	
Freight	money	

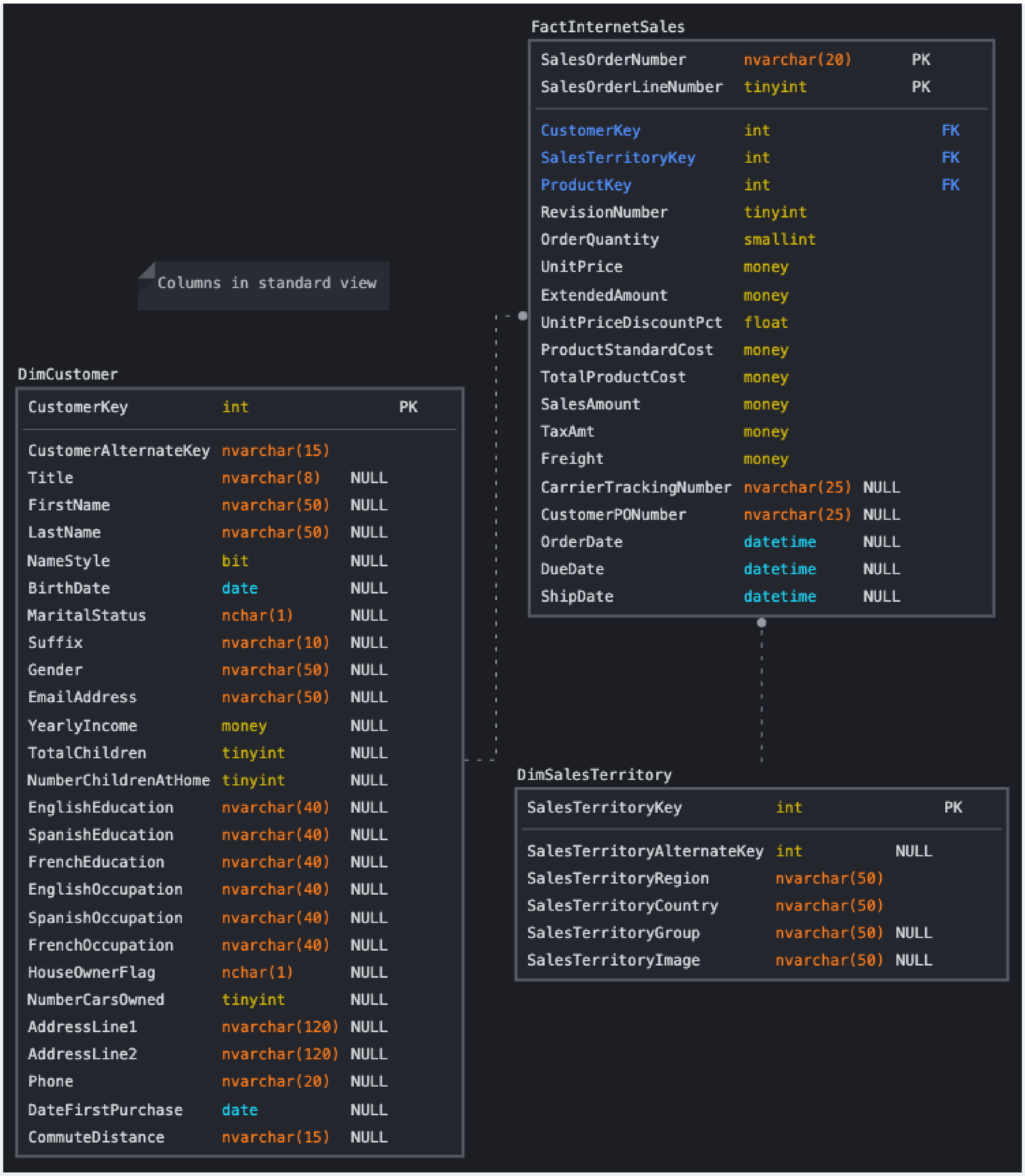
DimCustomer

CustomerKey	int	PK
CustomerAlternateKey	nvarchar(15)	

DimSalesTerritory

SalesTerritoryKey	int	PK

Columns from Standard View



Columns in standard view

FactInternetSales

SalesOrderNumber	nvarchar(20)		PK
SalesOrderLineNumber	tinyint		PK
CustomerKey	int		FK
SalesTerritoryKey	int		FK
ProductKey	int		FK
RevisionNumber	tinyint		
OrderQuantity	smallint		
UnitPrice	money		
ExtendedAmount	money		
UnitPriceDiscountPct	float		
ProductStandardCost	money		
TotalProductCost	money		
SalesAmount	money		
TaxAmt	money		
Freight	money		
CarrierTrackingNumber	nvarchar(25)	NULL	
CustomerPONumber	nvarchar(25)	NULL	
OrderDate	datetime	NULL	
DueDate	datetime	NULL	
ShipDate	datetime	NULL	

DimCustomer

CustomerKey	int		PK
CustomerAlternateKey	nvarchar(15)		
Title	nvarchar(8)	NULL	
FirstName	nvarchar(50)	NULL	
LastName	nvarchar(50)	NULL	
NameStyle	bit	NULL	
BirthDate	date	NULL	
MaritalStatus	nchar(1)	NULL	
Suffix	nvarchar(10)	NULL	
Gender	nvarchar(50)	NULL	
EmailAddress	nvarchar(50)	NULL	
YearlyIncome	money	NULL	
TotalChildren	tinyint	NULL	
NumberChildrenAtHome	tinyint	NULL	
EnglishEducation	nvarchar(40)	NULL	
SpanishEducation	nvarchar(40)	NULL	
FrenchEducation	nvarchar(40)	NULL	
EnglishOccupation	nvarchar(40)	NULL	
SpanishOccupation	nvarchar(40)	NULL	
FrenchOccupation	nvarchar(40)	NULL	
HouseOwnerFlag	nchar(1)	NULL	
NumberCarsOwned	tinyint	NULL	
AddressLine1	nvarchar(120)	NULL	
AddressLine2	nvarchar(120)	NULL	
Phone	nvarchar(20)	NULL	
DateFirstPurchase	date	NULL	
CommuteDistance	nvarchar(15)	NULL	

DimSalesTerritory

SalesTerritoryKey	int		PK
SalesTerritoryAlternateKey	int	NULL	
SalesTerritoryRegion	nvarchar(50)		
SalesTerritoryCountry	nvarchar(50)		
SalesTerritoryGroup	nvarchar(50)	NULL	
SalesTerritoryImage	nvarchar(50)	NULL	

Proposition 03: Rank the orders for Northwest internet sales, based on the customer's number of cars, using yearly income as a tie breaker.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

The first task is to find all internet sales in the Northwest region. First find the key value for the Northwest Region from DimSalesTerritory. Use this scalar value to filter through the FactInternetSales table for the desired orders. Next retrieve any given customer's number of cars and yearly income. Use cross apply to calculate a row (1 rowed table) for each customer from the Northwest orders. Finally use the number of cars and yearly income to create rows.

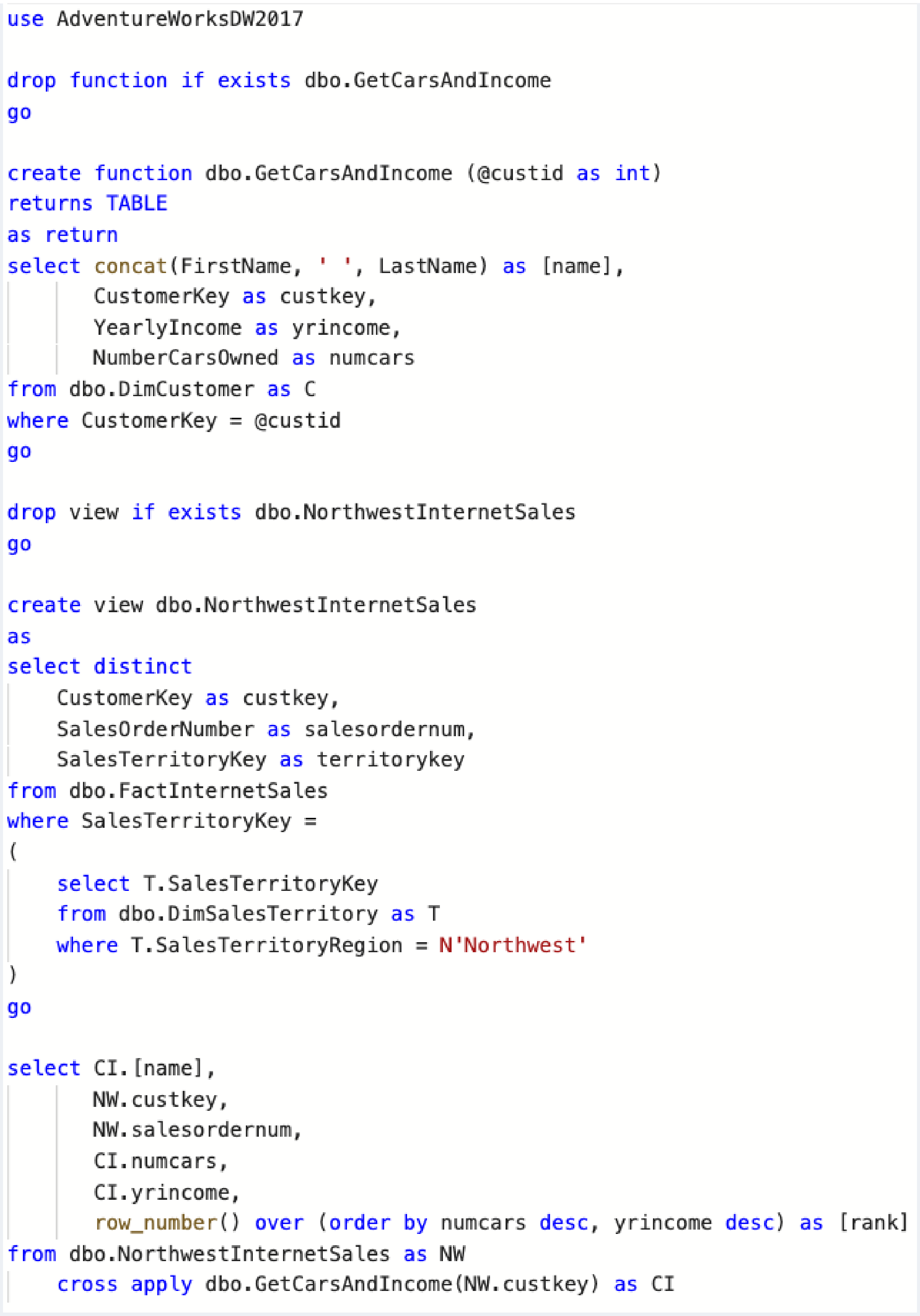
Project following columns from their respective tables in the select clause

Table Name	Column Name
dbo.DimCustomer	FirstName LastName CustomerKey NumberCarsOwned YearlyIncome
dbo.GetCarsAndIncome (function returns table)	name custkey yrincome numcars
dbo.FactInternetSales	CustomerKey SalesOrderNumber SalesTerritoryKey
dbo.DimSalesTerritory	SalesTerritoryKey
dbo.NorthWestInternetSales (view)	custkey salesordernum

Order By

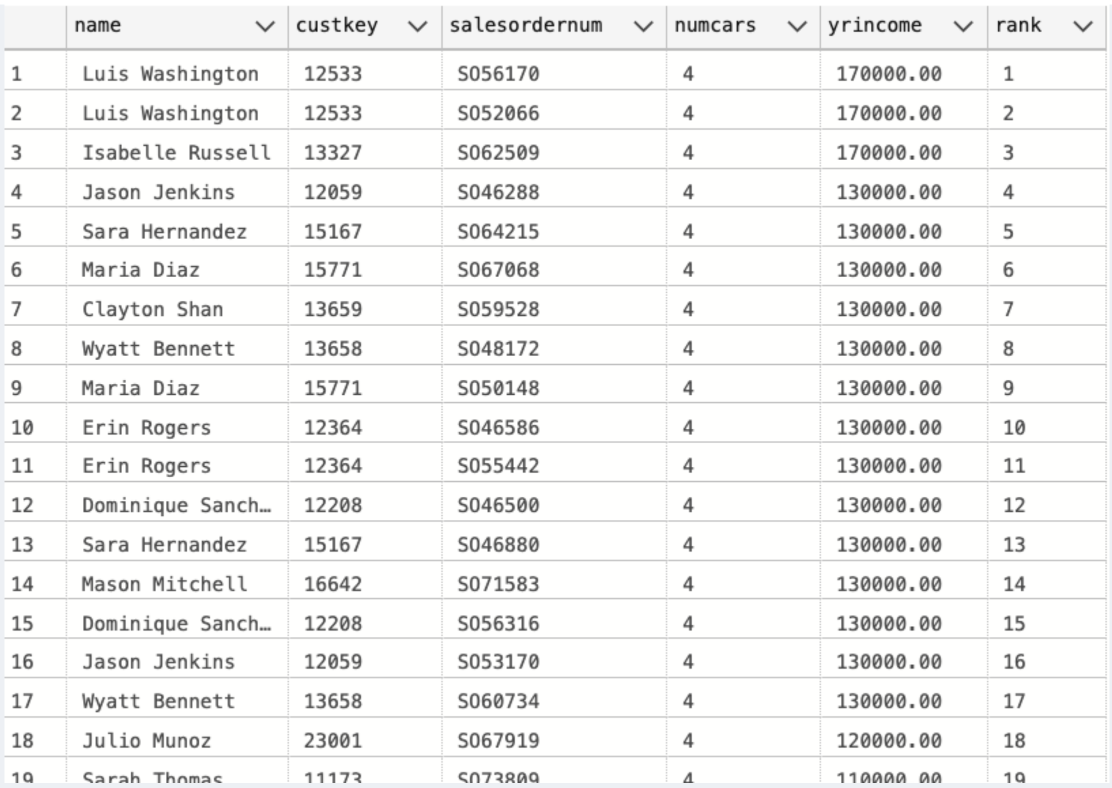
Table Name	Column Name	Sort Order
dbo.GetCarsAndIncome (function returns table)	numcars yrincome	DESC DESC

Problem Solving Query



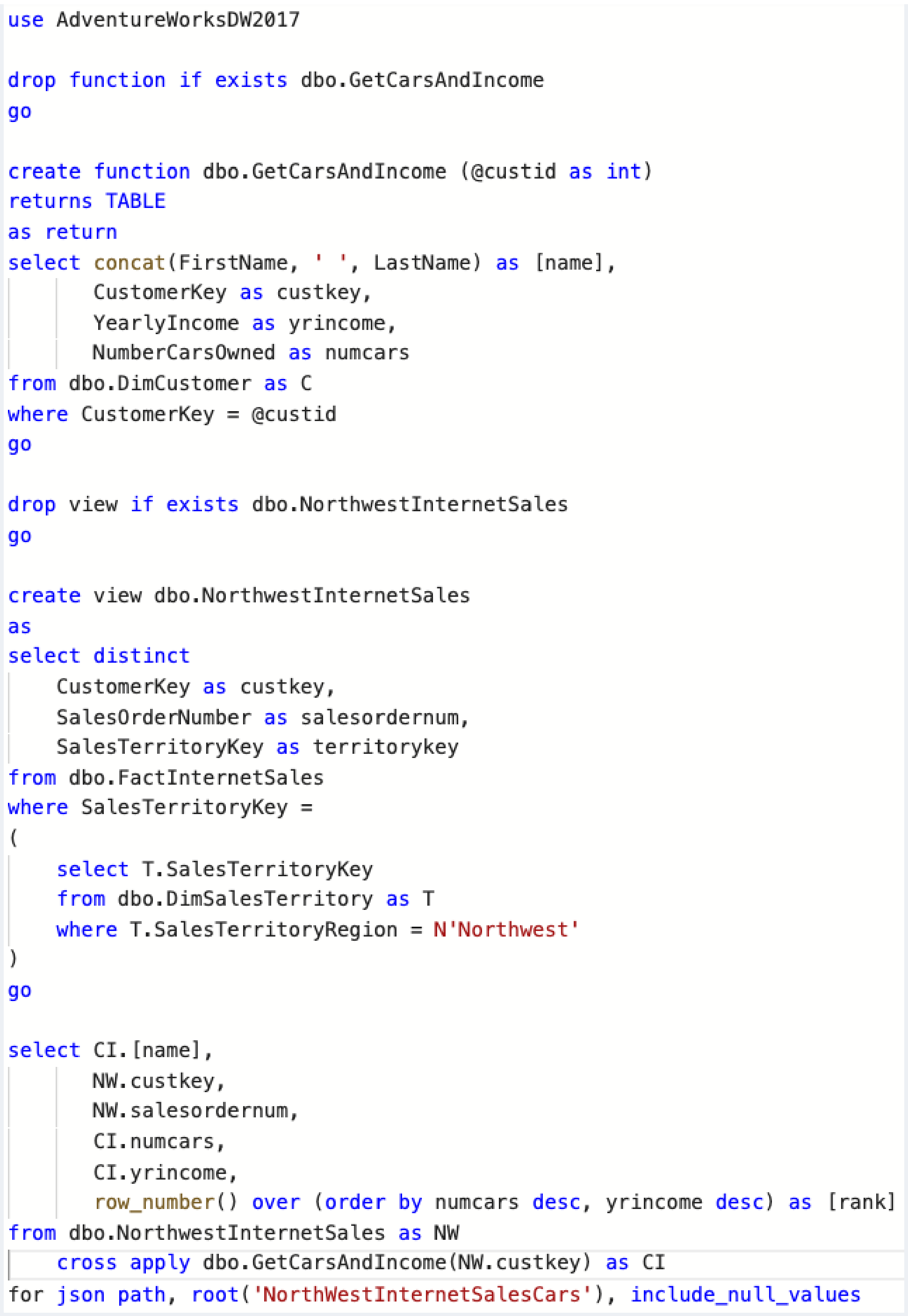
```sql
use AdventureWorksDW2017

drop function if exists dbo.GetCarsAndIncome
go

create function dbo.GetCarsAndIncome (@custid as int)
returns TABLE
as return
select concat(FirstName, ' ', LastName) as [name],
       CustomerKey as custkey,
       YearlyIncome as yrincome,
       NumberCarsOwned as numcars
from dbo.DimCustomer as C
where CustomerKey = @custid
go

drop view if exists dbo.NorthwestInternetSales
go

create view dbo.NorthwestInternetSales
as
select distinct
    CustomerKey as custkey,
    SalesOrderNumber as salesordernum,
    SalesTerritoryKey as territorykey
from dbo.FactInternetSales
where SalesTerritoryKey =
(
    select T.SalesTerritoryKey
    from dbo.DimSalesTerritory as T
    where T.SalesTerritoryRegion = N'Northwest'
)
go

select CI.[name],
       NW.custkey,
       NW.salesordernum,
       CI.numcars,
       CI.yrincome,
       row_number() over (order by numcars desc, yrincome desc) as [rank]
from dbo.NorthwestInternetSales as NW
    cross apply dbo.GetCarsAndIncome(NW.custkey) as CI
```

Sample Relational Output with total number of rows returned (4058)

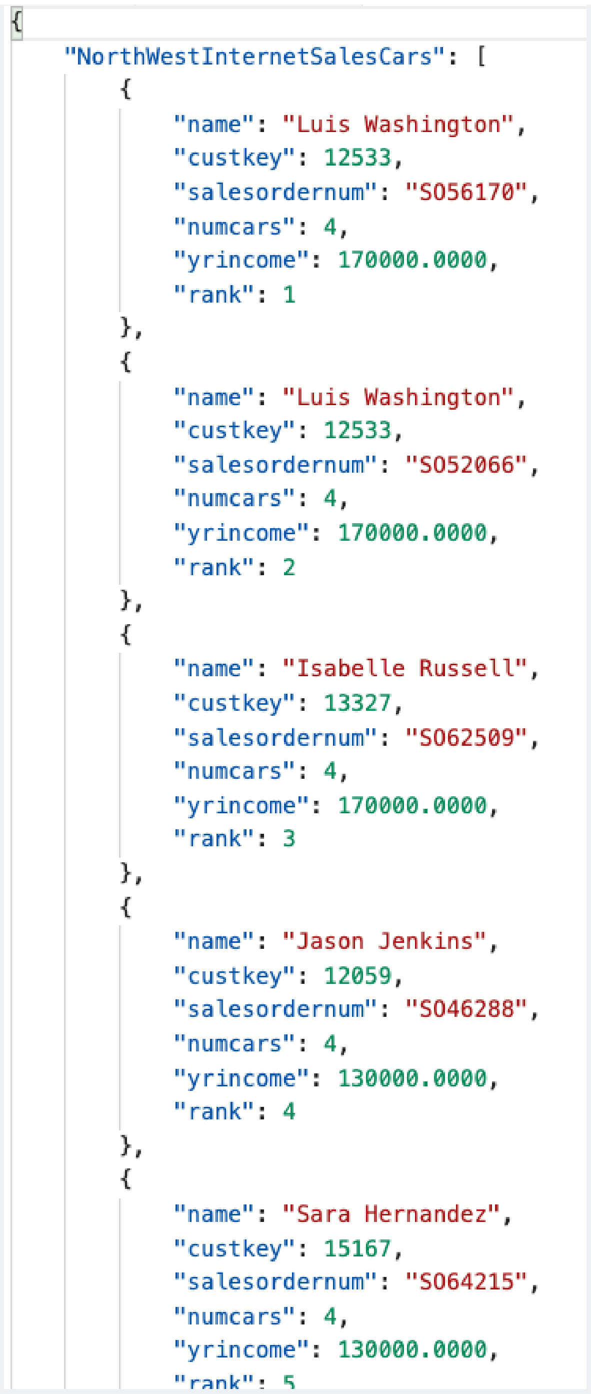


	name	custkey	salesordernum	numcars	yrincome	rank
1	Luis Washington	12533	SO56170	4	170000.00	1
2	Luis Washington	12533	SO52066	4	170000.00	2
3	Isabelle Russell	13327	SO62509	4	170000.00	3
4	Jason Jenkins	12059	SO46288	4	130000.00	4
5	Sara Hernandez	15167	SO64215	4	130000.00	5
6	Maria Diaz	15771	SO67068	4	130000.00	6
7	Clayton Shan	13659	SO59528	4	130000.00	7
8	Wyatt Bennett	13658	SO48172	4	130000.00	8
9	Maria Diaz	15771	SO50148	4	130000.00	9
10	Erin Rogers	12364	SO46586	4	130000.00	10
11	Erin Rogers	12364	SO55442	4	130000.00	11
12	Dominique Sanch…	12208	SO46500	4	130000.00	12
13	Sara Hernandez	15167	SO46880	4	130000.00	13
14	Mason Mitchell	16642	SO71583	4	130000.00	14
15	Dominique Sanch…	12208	SO56316	4	130000.00	15
16	Jason Jenkins	12059	SO53170	4	130000.00	16
17	Wyatt Bennett	13658	SO60734	4	130000.00	17
18	Julio Munoz	23001	SO67919	4	120000.00	18
19	Sarah Thomas	11173	SO73809	4	110000.00	19

Sample JSON Output with total number of rows returned (4058)



```sql
use AdventureWorksDW2017

drop function if exists dbo.GetCarsAndIncome
go

create function dbo.GetCarsAndIncome (@custid as int)
returns TABLE
as return
select concat(FirstName, ' ', LastName) as [name],
       CustomerKey as custkey,
       YearlyIncome as yrincome,
       NumberCarsOwned as numcars
from dbo.DimCustomer as C
where CustomerKey = @custid
go

drop view if exists dbo.NorthwestInternetSales
go

create view dbo.NorthwestInternetSales
as
select distinct
    CustomerKey as custkey,
    SalesOrderNumber as salesordernum,
    SalesTerritoryKey as territorykey
from dbo.FactInternetSales
where SalesTerritoryKey =
(
    select T.SalesTerritoryKey
    from dbo.DimSalesTerritory as T
    where T.SalesTerritoryRegion = N'Northwest'
)
go

select CI.[name],
       NW.custkey,
       NW.salesordernum,
       CI.numcars,
       CI.yrincome,
       row_number() over (order by numcars desc, yrincome desc) as [rank]
from dbo.NorthwestInternetSales as NW
    cross apply dbo.GetCarsAndIncome(NW.custkey) as CI
for json path, root('NorthWestInternetSalesCars'), include_null_values
```



```
{
    "NorthWestInternetSalesCars": [
        {
            "name": "Luis Washington",
            "custkey": 12533,
            "salesordernum": "SO56170",
            "numcars": 4,
            "yrincome": 170000.0000,
            "rank": 1
        },
        {
            "name": "Luis Washington",
            "custkey": 12533,
            "salesordernum": "SO52066",
            "numcars": 4,
            "yrincome": 170000.0000,
            "rank": 2
        },
        {
            "name": "Isabelle Russell",
            "custkey": 13327,
            "salesordernum": "SO62509",
            "numcars": 4,
            "yrincome": 170000.0000,
            "rank": 3
        },
        {
            "name": "Jason Jenkins",
            "custkey": 12059,
            "salesordernum": "SO46288",
            "numcars": 4,
            "yrincome": 130000.0000,
            "rank": 4
        },
        {
            "name": "Sara Hernandez",
            "custkey": 15167,
            "salesordernum": "SO64215",
            "numcars": 4,
            "yrincome": 130000.0000,
            "rank": 5
```

WORST

SIMPLE

Example of Sales2018 sub-system in PrestigeCarsOriginal

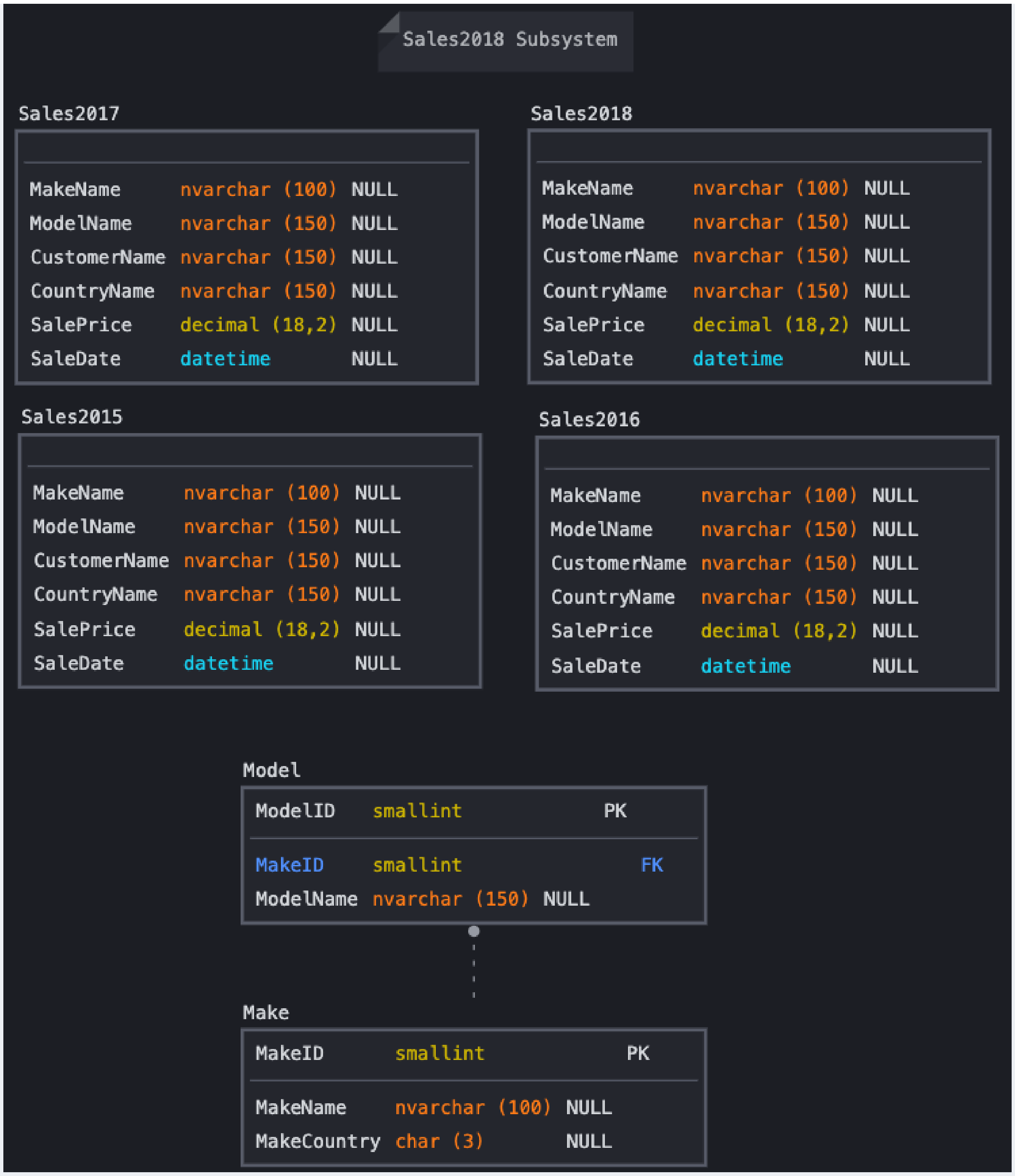
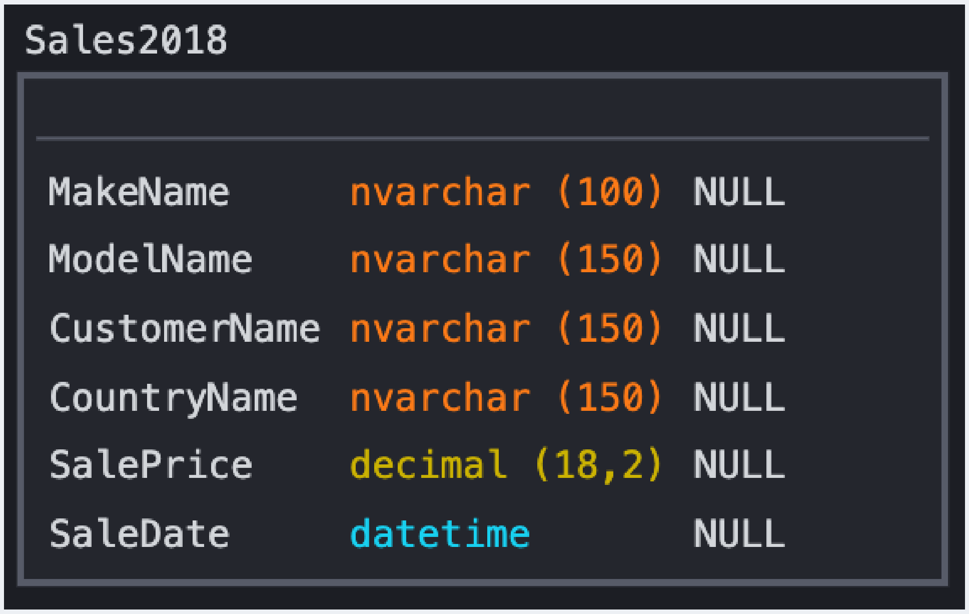
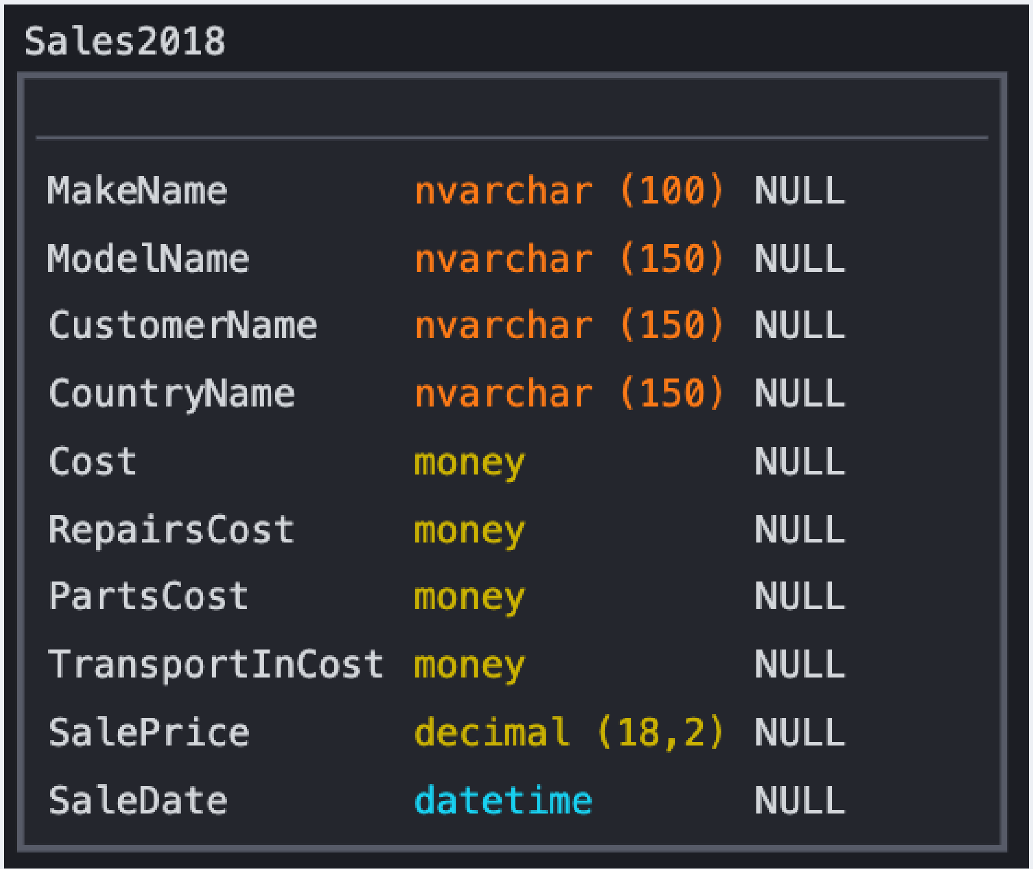

Sales2018 Subsystem
Sales2017
MakeName nvarchar (100) NULL
ModelName nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName nvarchar (150) NULL
SalePrice decimal (18,2) NULL
SaleDate datetime NULL
Sales2018
MakeName nvarchar (100) NULL
ModelName nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName nvarchar (150) NULL
SalePrice decimal (18,2) NULL
SaleDate datetime NULL
Sales2015
MakeName nvarchar (100) NULL
ModelName nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName nvarchar (150) NULL
SalePrice decimal (18,2) NULL
SaleDate datetime NULL
Sales2016
MakeName nvarchar (100) NULL
ModelName nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName nvarchar (150) NULL
SalePrice decimal (18,2) NULL
SaleDate datetime NULL
Model
ModelID smallint PK
MakeID smallint FK
ModelName nvarchar (150) NULL
Make
MakeID smallint PK
MakeName nvarchar (100) NULL
MakeCountry char (3) NULL

Diagram of Tables



```
Sales2018

MakeName     nvarchar (100) NULL
ModelName    nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName  nvarchar (150) NULL
SalePrice    decimal (18,2) NULL
SaleDate     datetime       NULL
```

Columns from Standard View



```
Sales2018

MakeName        nvarchar (100) NULL
ModelName       nvarchar (150) NULL
CustomerName    nvarchar (150) NULL
CountryName     nvarchar (150) NULL
Cost            money          NULL
RepairsCost     money          NULL
PartsCost       money          NULL
TransportInCost money          NULL
SalePrice       decimal (18,2) NULL
SaleDate        datetime       NULL
```

Proposition 04: Return the total sales price for each make of cars sold in 2018.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

An example of a make of car would be "BMW" or "Bugatti". The total sales price would be the sum of all sales prices in 2018 for the specific make. Necessary to use a group by.

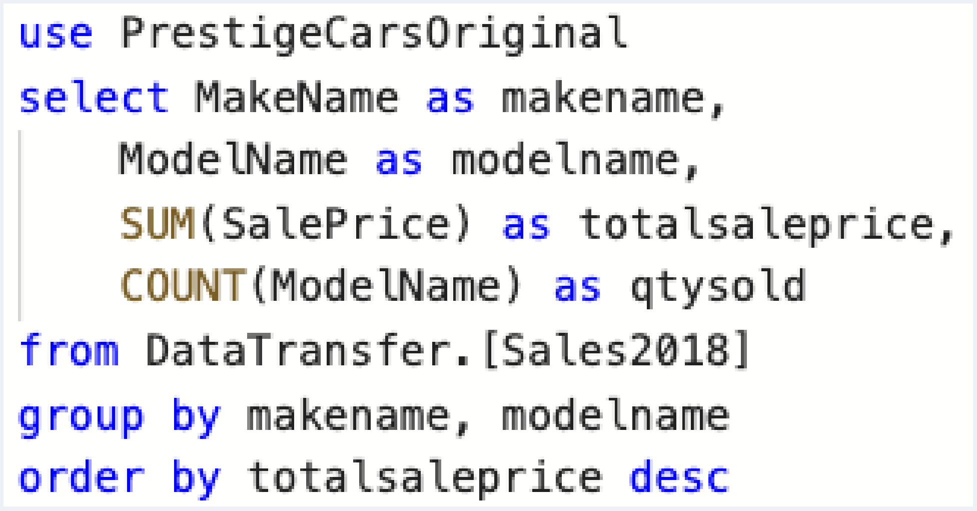
Project following columns from their respective tables in the select clause

Table Name	Column Name
DataTransfer.Sales2018	MakeName ModelName SalePrice

Order By

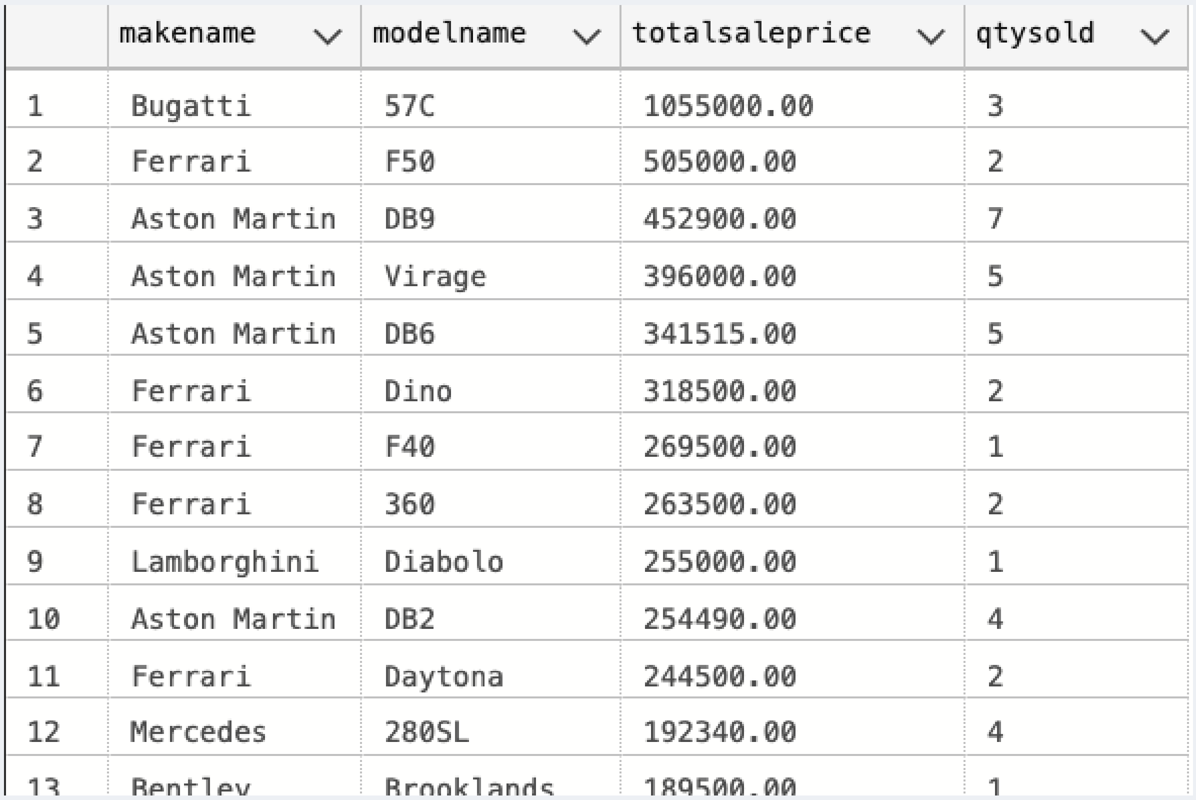
Table Name	Column Name	Sort Order
DataTransfer.Sales2018	totalsalesprice	DESC

Problem Solving Query



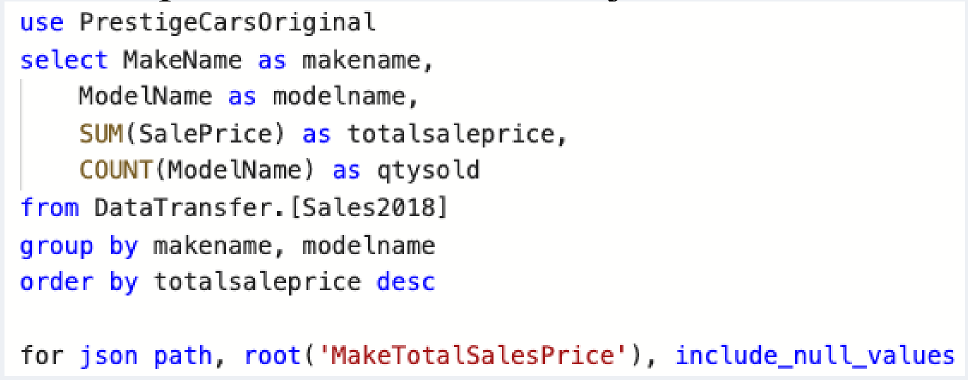
```
use PrestigeCarsOriginal
select MakeName as makename,
    ModelName as modelname,
    SUM(SalePrice) as totalsaleprice,
    COUNT(ModelName) as qtysold
from DataTransfer.[Sales2018]
group by makename, modelname
order by totalsaleprice desc
```

Sample Relational Output with total number of rows returned (61)

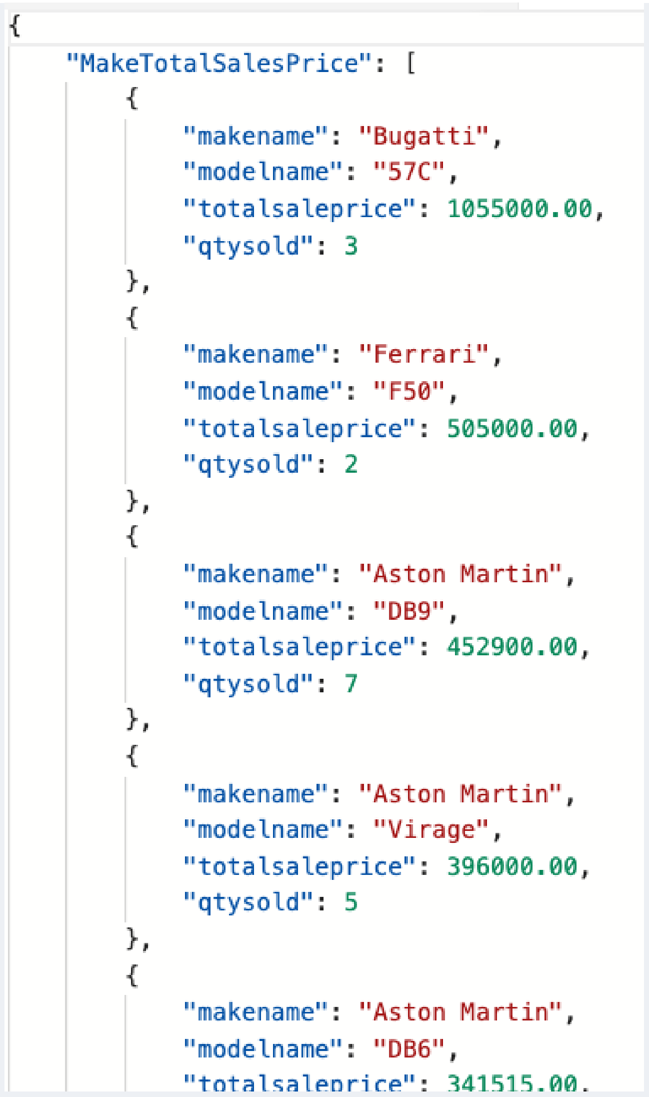


	makename	modelname	totalsaleprice	qtysold
1	Bugatti	57C	1055000.00	3
2	Ferrari	F50	505000.00	2
3	Aston Martin	DB9	452900.00	7
4	Aston Martin	Virage	396000.00	5
5	Aston Martin	DB6	341515.00	5
6	Ferrari	Dino	318500.00	2
7	Ferrari	F40	269500.00	1
8	Ferrari	360	263500.00	2
9	Lamborghini	Diabolo	255000.00	1
10	Aston Martin	DB2	254490.00	4
11	Ferrari	Daytona	244500.00	2
12	Mercedes	280SL	192340.00	4
13	Bentley	Brooklands	189500.00	1

Sample JSON Output with total number of rows returned (61)



```
use PrestigeCarsOriginal
select MakeName as makename,
    ModelName as modelname,
    SUM(SalePrice) as totalsaleprice,
    COUNT(ModelName) as qtysold
from DataTransfer.[Sales2018]
group by makename, modelname
order by totalsaleprice desc

for json path, root('MakeTotalSalesPrice'), include_null_values
```



```
{
    "MakeTotalSalesPrice": [
        {
            "makename": "Bugatti",
            "modelname": "57C",
            "totalsaleprice": 1055000.00,
            "qtysold": 3
        },
        {
            "makename": "Ferrari",
            "modelname": "F50",
            "totalsaleprice": 505000.00,
            "qtysold": 2
        },
        {
            "makename": "Aston Martin",
            "modelname": "DB9",
            "totalsaleprice": 452900.00,
            "qtysold": 7
        },
        {
            "makename": "Aston Martin",
            "modelname": "Virage",
            "totalsaleprice": 396000.00,
            "qtysold": 5
        },
        {
            "makename": "Aston Martin",
            "modelname": "DB6",
            "totalsaleprice": 341515.00,
```

MEDIUM

Example of WorkOrder sub-system in AdventureWorks2017

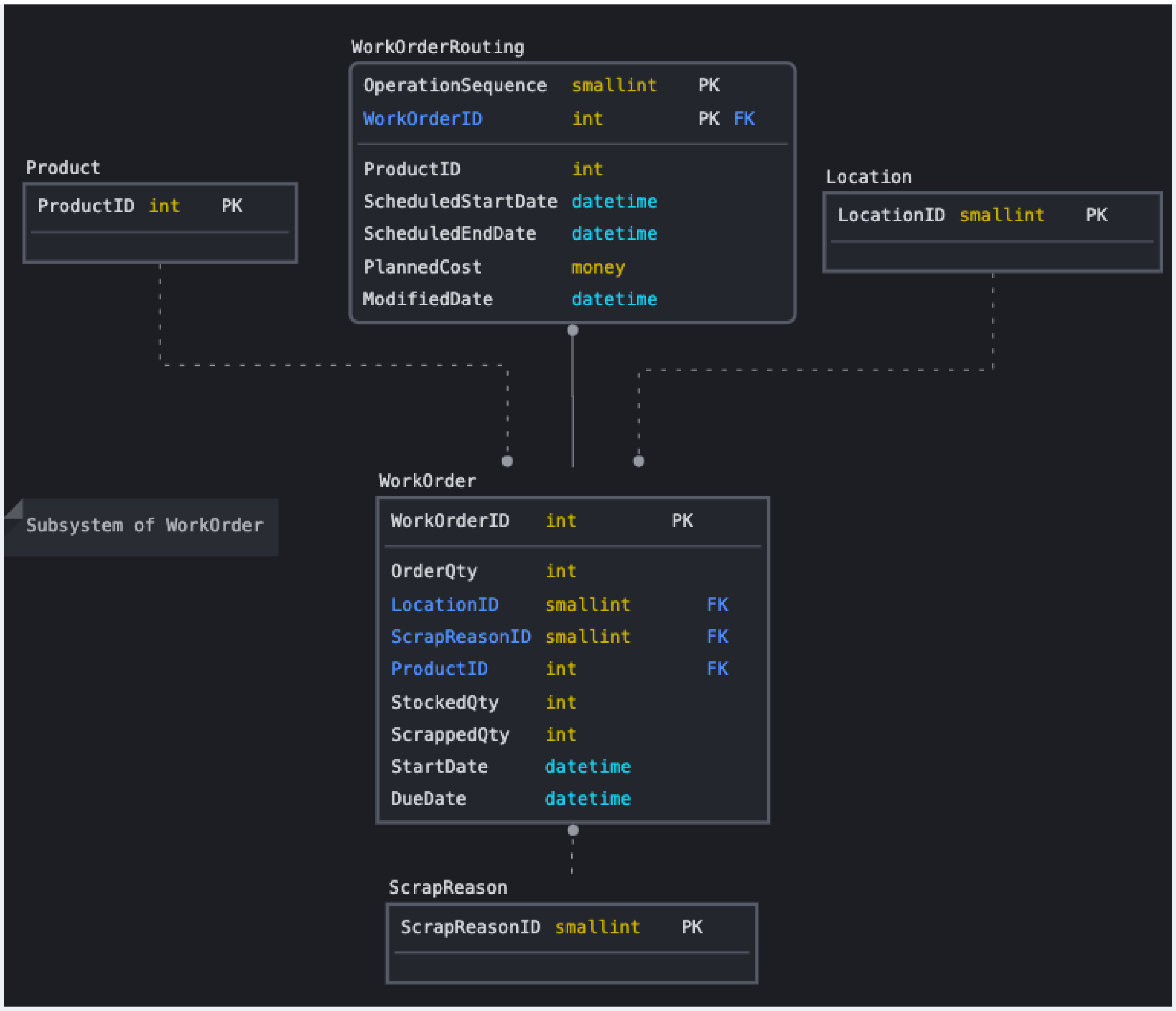
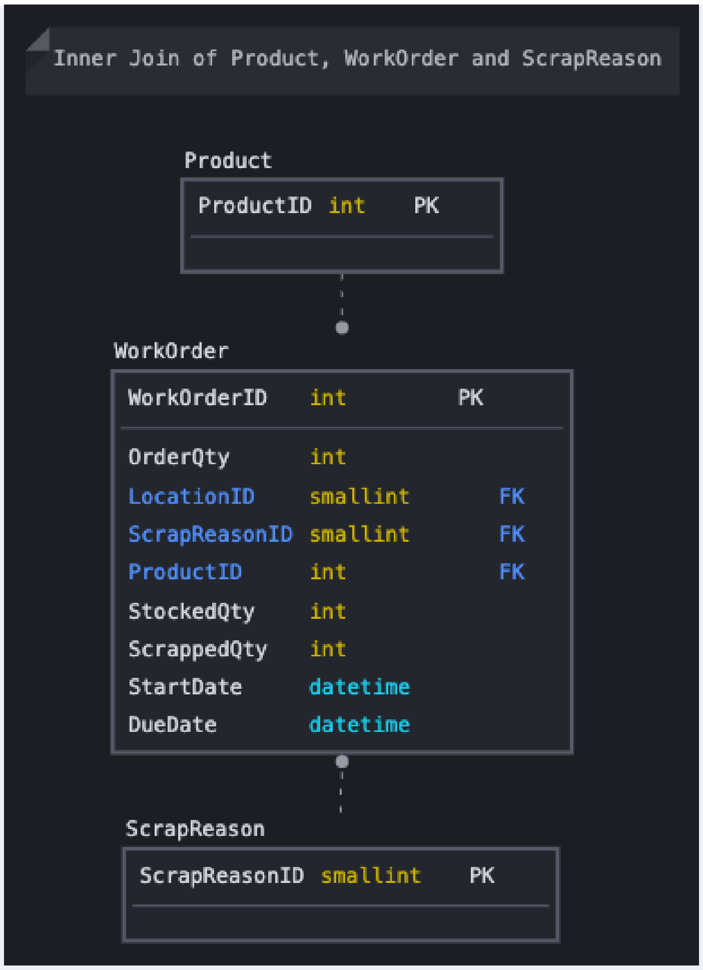
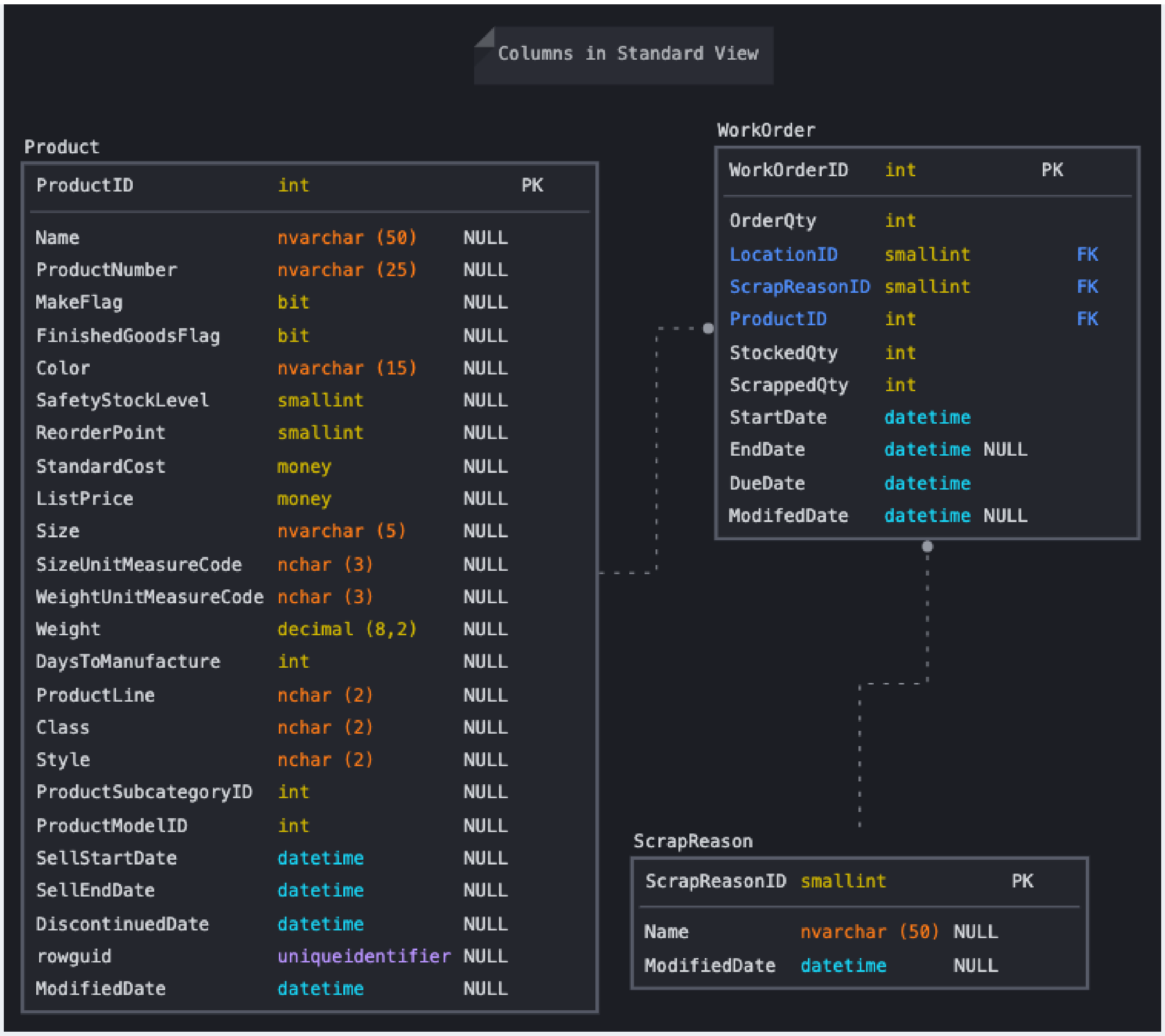


Diagram of Tables



Columns from Standard View



Product

ProductID	int	PK
Name	nvarchar (50)	NULL
ProductNumber	nvarchar (25)	NULL
MakeFlag	bit	NULL
FinishedGoodsFlag	bit	NULL
Color	nvarchar (15)	NULL
SafetyStockLevel	smallint	NULL
ReorderPoint	smallint	NULL
StandardCost	money	NULL
ListPrice	money	NULL
Size	nvarchar (5)	NULL
SizeUnitMeasureCode	nchar (3)	NULL
WeightUnitMeasureCode	nchar (3)	NULL
Weight	decimal (8,2)	NULL
DaysToManufacture	int	NULL
ProductLine	nchar (2)	NULL
Class	nchar (2)	NULL
Style	nchar (2)	NULL
ProductSubcategoryID	int	NULL
ProductModelID	int	NULL
SellStartDate	datetime	NULL
SellEndDate	datetime	NULL
DiscontinuedDate	datetime	NULL
rowguid	uniqueidentifier	NULL
ModifiedDate	datetime	NULL

WorkOrder

WorkOrderID	int		PK
OrderQty	int		
LocationID	smallint		FK
ScrapReasonID	smallint		FK
ProductID	int		FK
StockedQty	int		
ScrappedQty	int		
StartDate	datetime		
EndDate	datetime	NULL	
DueDate	datetime		
ModifedDate	datetime	NULL	

ScrapReason

ScrapReasonID	smallint	PK
Name	nvarchar (50)	NULL
ModifiedDate	datetime	NULL

Proposition 05: Return all scrapped product names with the reason they were scrapped.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

A "scrapped" product is a product part of a business work order. A work order for a given product has a quantity of the expected merchandise, and a quantity of goods that were scrapped in the process to fulfillment. Find all orders where the scrapped quantity is more than 0. Orders with a scrapped quantity greater than 0 have a reason ID and product ID paired with them. These foreign keys can be used to access information from the Product and ScrapReason tables like the reason description and product name.

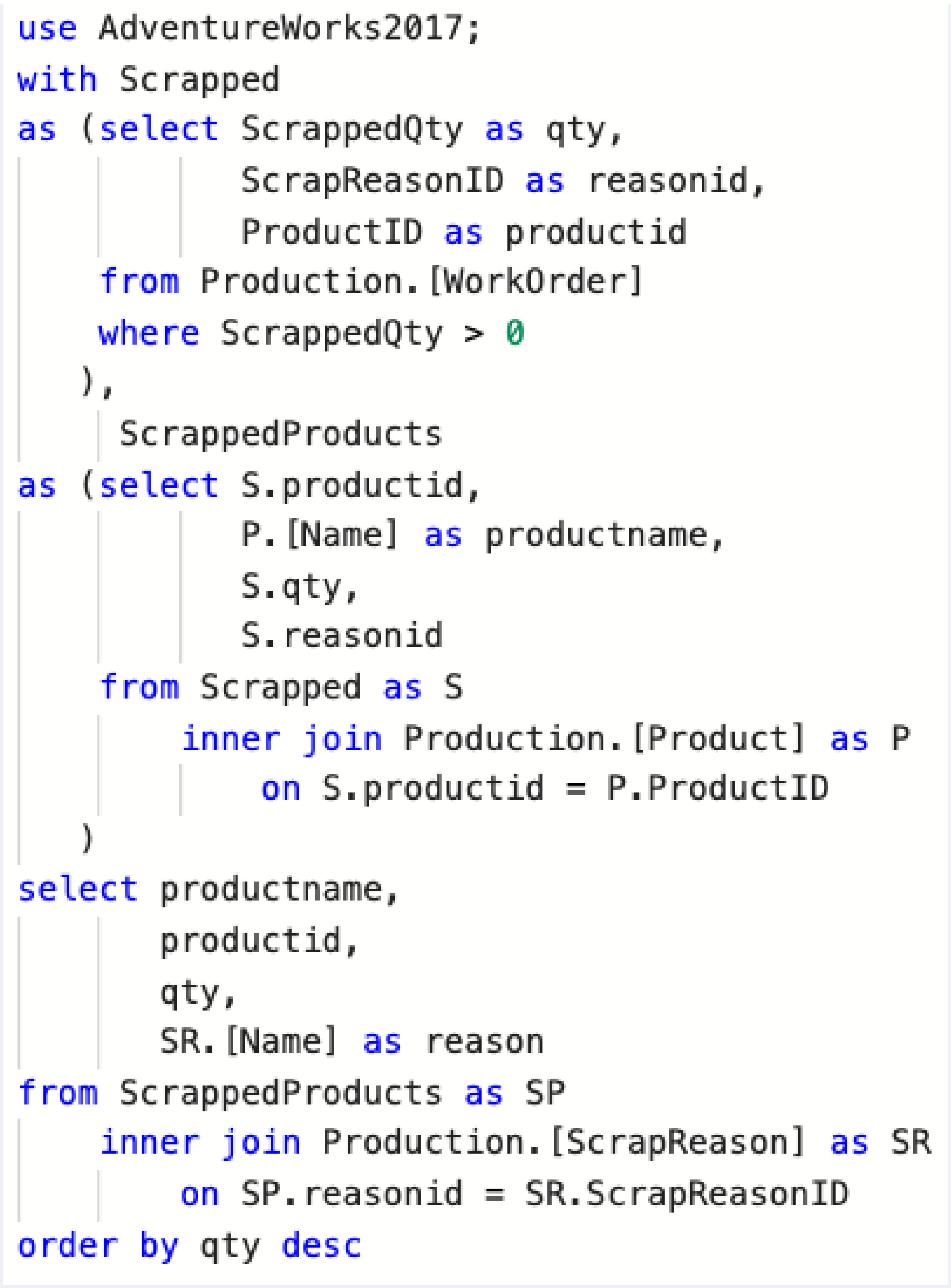
Project following columns from their respective tables in the select clause

Table Name	Column Name
Production.WorkOrder	ScrappedQty ScrapReasonID ProductID
Scrapped	qty reasonid productid
Production.Product	Name
ScrappedProducts	productname
Production.[ScrapReason]	Name

Order By

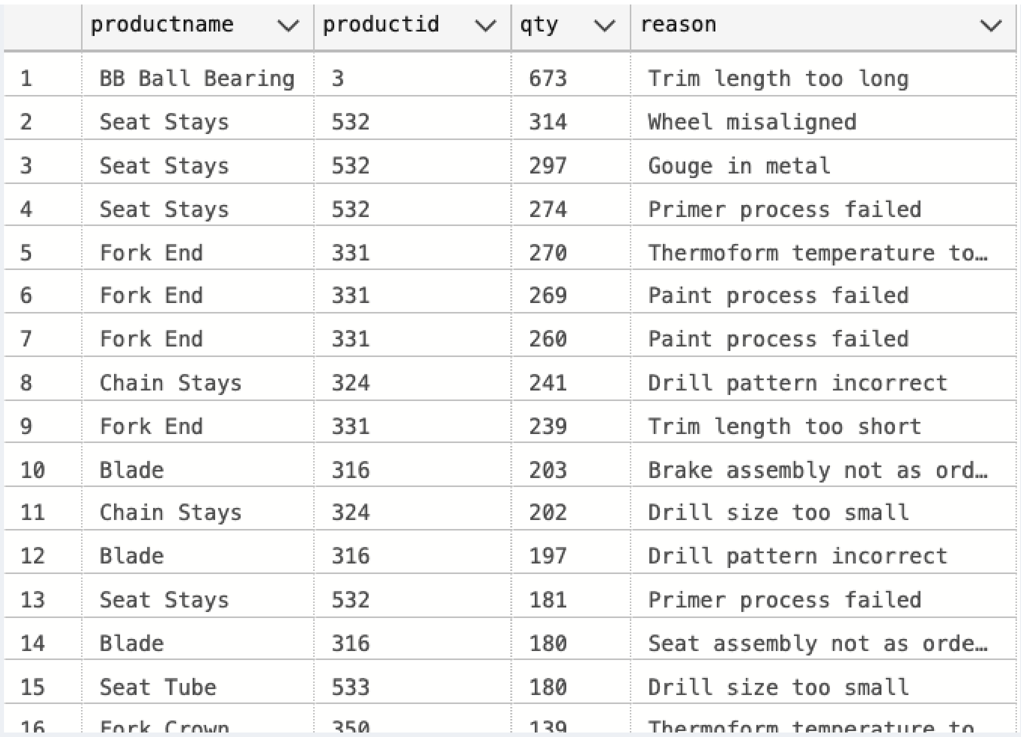
Table Name	Column Name	Sort Order
Production.WorkOrder	ScrappedQty	DESC

Problem Solving Query



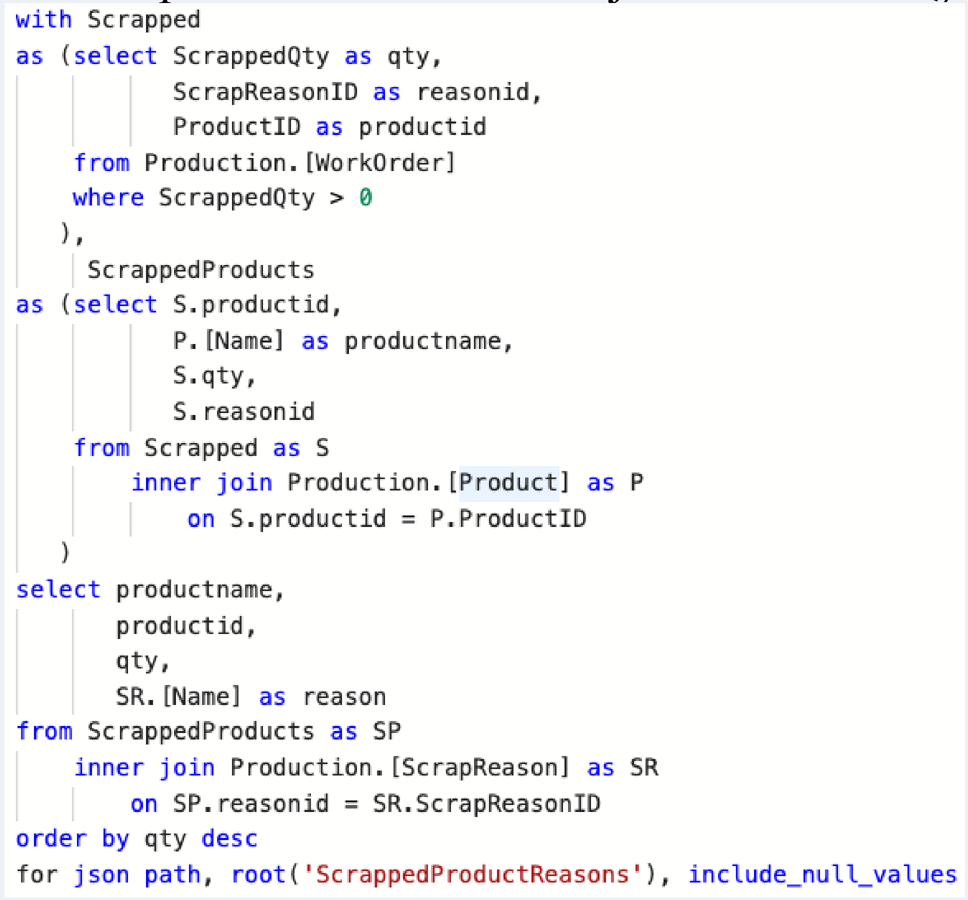
```sql
use AdventureWorks2017;
with Scrapped
as (select ScrappedQty as qty,
           ScrapReasonID as reasonid,
           ProductID as productid
    from Production.[WorkOrder]
    where ScrappedQty > 0
   ),
    ScrappedProducts
as (select S.productid,
           P.[Name] as productname,
           S.qty,
           S.reasonid
    from Scrapped as S
        inner join Production.[Product] as P
            on S.productid = P.ProductID
   )
select productname,
       productid,
       qty,
       SR.[Name] as reason
from ScrappedProducts as SP
    inner join Production.[ScrapReason] as SR
        on SP.reasonid = SR.ScrapReasonID
order by qty desc
```

Sample Relational Output with total number of rows returned (729)

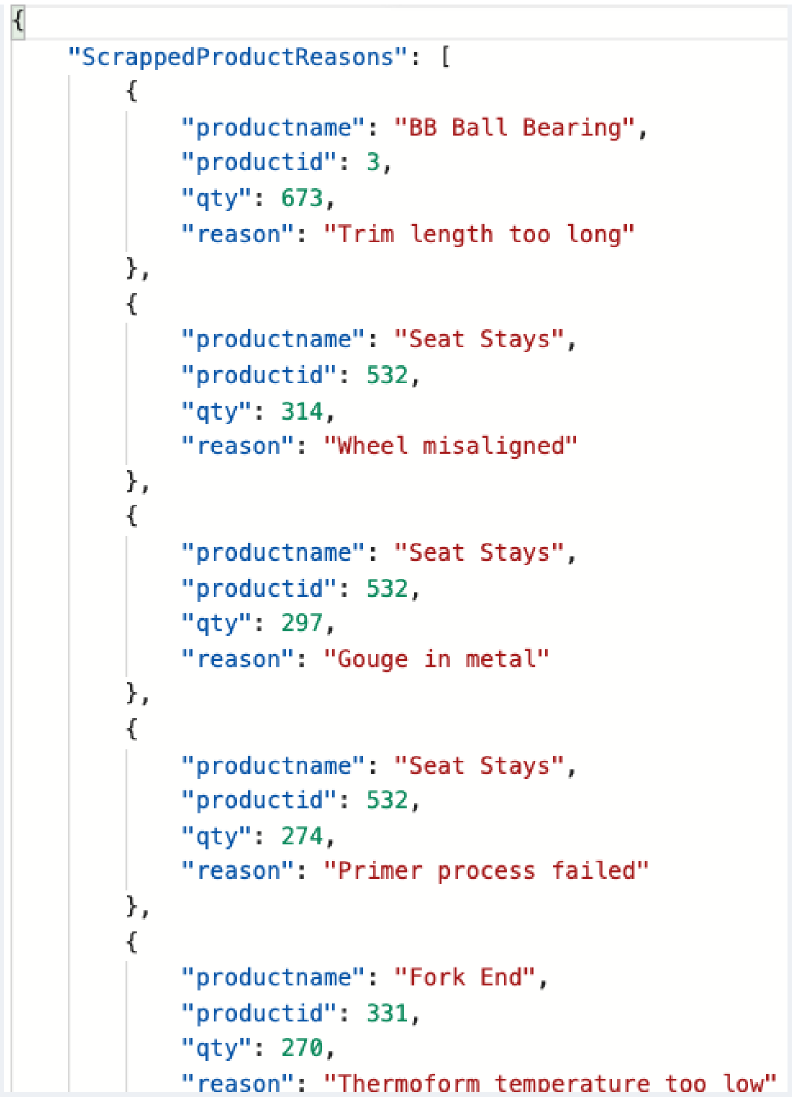


	productname	productid	qty	reason
1	BB Ball Bearing	3	673	Trim length too long
2	Seat Stays	532	314	Wheel misaligned
3	Seat Stays	532	297	Gouge in metal
4	Seat Stays	532	274	Primer process failed
5	Fork End	331	270	Thermoform temperature to…
6	Fork End	331	269	Paint process failed
7	Fork End	331	260	Paint process failed
8	Chain Stays	324	241	Drill pattern incorrect
9	Fork End	331	239	Trim length too short
10	Blade	316	203	Brake assembly not as ord…
11	Chain Stays	324	202	Drill size too small
12	Blade	316	197	Drill pattern incorrect
13	Seat Stays	532	181	Primer process failed
14	Blade	316	180	Seat assembly not as orde…
15	Seat Tube	533	180	Drill size too small
16	Fork Crown	350	139	Thermoform temperature to

Sample JSON Output with total number of rows returned (729)



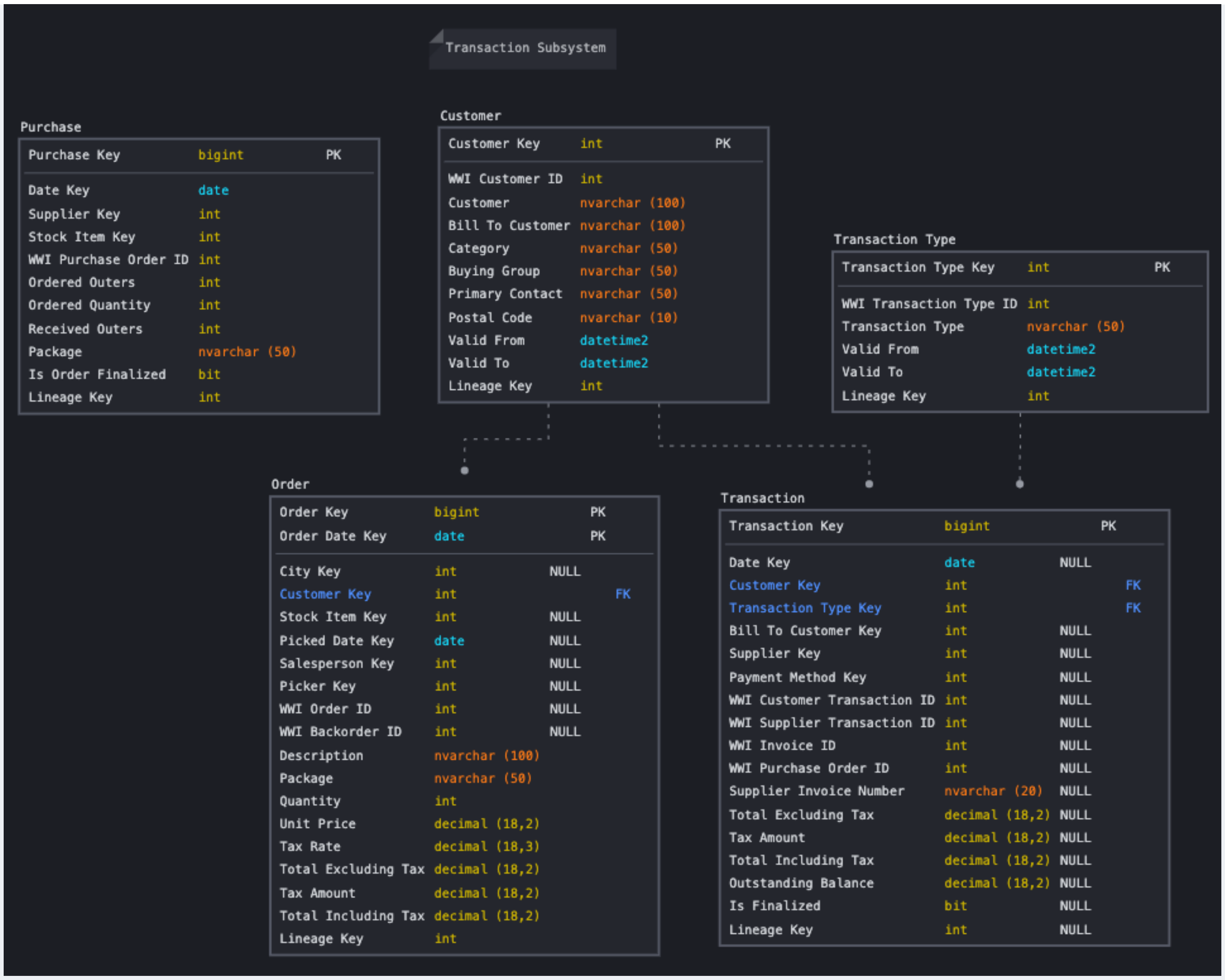
```sql
with Scrapped
as (select ScrappedQty as qty,
           ScrapReasonID as reasonid,
           ProductID as productid
    from Production.[WorkOrder]
    where ScrappedQty > 0
   ),
     ScrappedProducts
as (select S.productid,
           P.[Name] as productname,
           S.qty,
           S.reasonid
    from Scrapped as S
        inner join Production.[Product] as P
            on S.productid = P.ProductID
   )
select productname,
       productid,
       qty,
       SR.[Name] as reason
from ScrappedProducts as SP
    inner join Production.[ScrapReason] as SR
        on SP.reasonid = SR.ScrapReasonID
order by qty desc
for json path, root('ScrappedProductReasons'), include_null_values
```



```
{
    "ScrappedProductReasons": [
        {
            "productname": "BB Ball Bearing",
            "productid": 3,
            "qty": 673,
            "reason": "Trim length too long"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 314,
            "reason": "Wheel misaligned"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 297,
            "reason": "Gouge in metal"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 274,
            "reason": "Primer process failed"
        },
        {
            "productname": "Fork End",
            "productid": 331,
            "qty": 270,
            "reason": "Thermoform temperature too low"
```

COMPLEX

Example of Transaction sub-system in WideWorldImportersDW



Purchase

Purchase Key	bigint	PK
Date Key	date	
Supplier Key	int	
Stock Item Key	int	
WWI Purchase Order ID	int	
Ordered Outers	int	
Ordered Quantity	int	
Received Outers	int	
Package	nvarchar (50)	
Is Order Finalized	bit	
Lineage Key	int	

Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

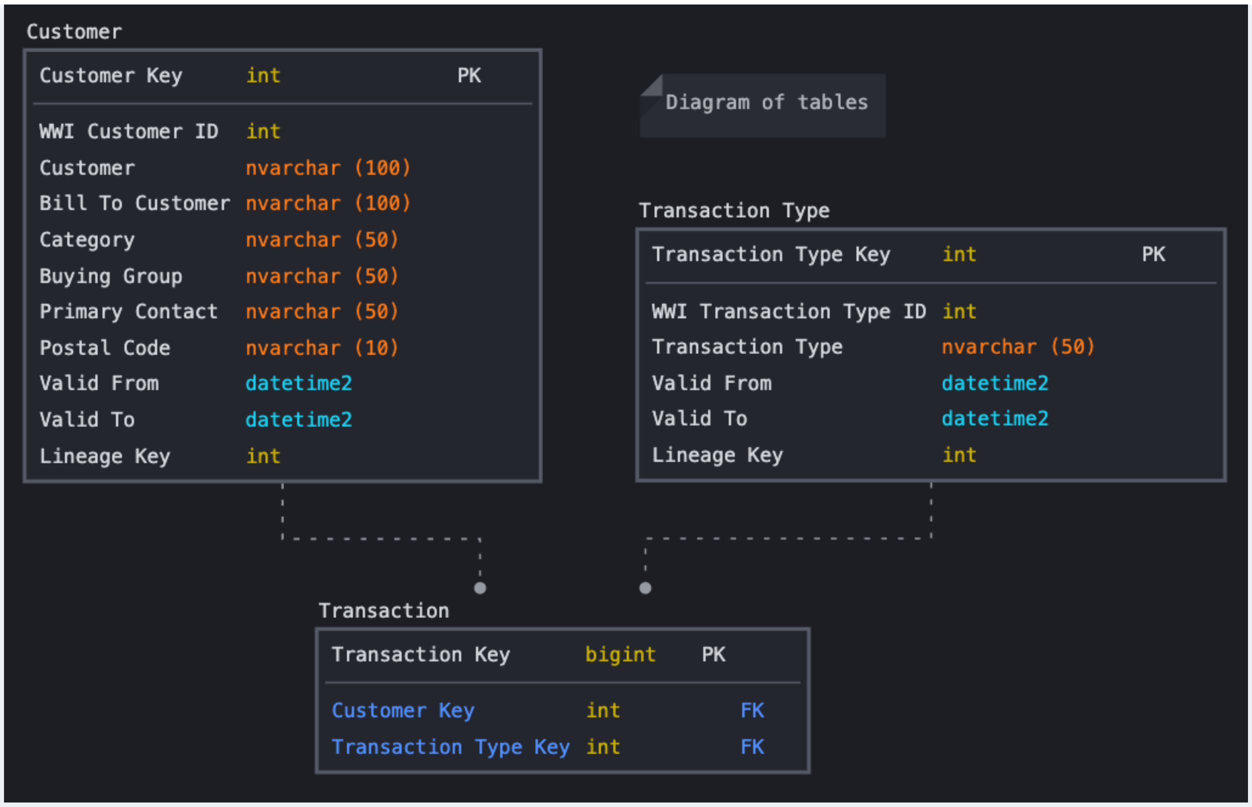
Order

Order Key	bigint		PK
Order Date Key	date		PK
City Key	int	NULL	
Customer Key	int		FK
Stock Item Key	int	NULL	
Picked Date Key	date	NULL	
Salesperson Key	int	NULL	
Picker Key	int	NULL	
WWI Order ID	int	NULL	
WWI Backorder ID	int	NULL	
Description	nvarchar (100)		
Package	nvarchar (50)		
Quantity	int		
Unit Price	decimal (18,2)		
Tax Rate	decimal (18,3)		
Total Excluding Tax	decimal (18,2)		
Tax Amount	decimal (18,2)		
Total Including Tax	decimal (18,2)		
Lineage Key	int		

Transaction

Transaction Key	bigint		PK
Date Key	date	NULL	
Customer Key	int		FK
Transaction Type Key	int		FK
Bill To Customer Key	int	NULL	
Supplier Key	int	NULL	
Payment Method Key	int	NULL	
WWI Customer Transaction ID	int	NULL	
WWI Supplier Transaction ID	int	NULL	
WWI Invoice ID	int	NULL	
WWI Purchase Order ID	int	NULL	
Supplier Invoice Number	nvarchar (20)	NULL	
Total Excluding Tax	decimal (18,2)	NULL	
Tax Amount	decimal (18,2)	NULL	
Total Including Tax	decimal (18,2)	NULL	
Outstanding Balance	decimal (18,2)	NULL	
Is Finalized	bit	NULL	
Lineage Key	int	NULL	

Diagram of Tables



Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

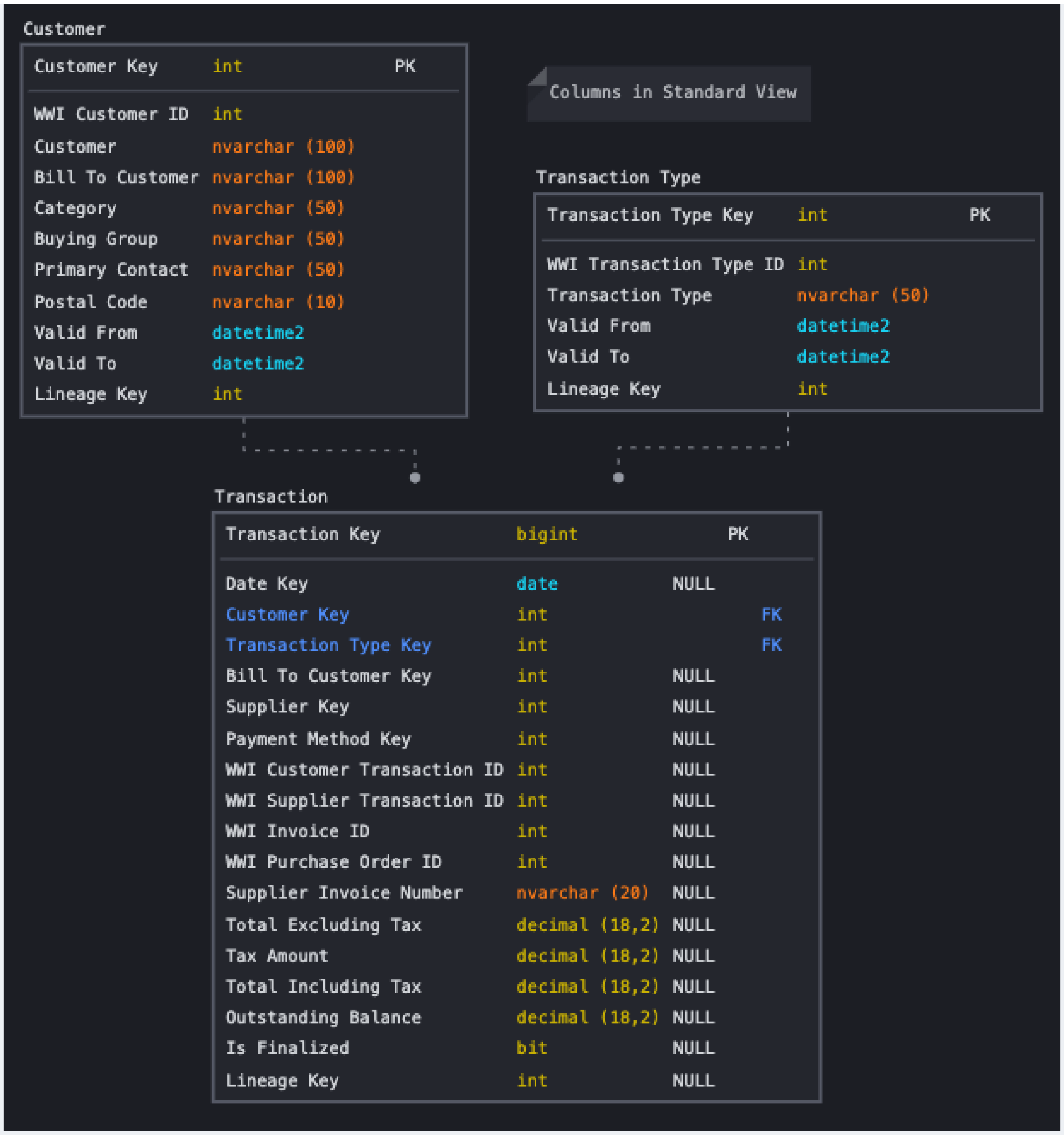
Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction

Transaction Key	bigint	PK
Customer Key	int	FK
Transaction Type Key	int	FK

Columns from Standard View



Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Columns in Standard View

Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction

Transaction Key	bigint	PK	
Date Key	date	NULL	
Customer Key	int		FK
Transaction Type Key	int		FK
Bill To Customer Key	int	NULL	
Supplier Key	int	NULL	
Payment Method Key	int	NULL	
WWI Customer Transaction ID	int	NULL	
WWI Supplier Transaction ID	int	NULL	
WWI Invoice ID	int	NULL	
WWI Purchase Order ID	int	NULL	
Supplier Invoice Number	nvarchar (20)	NULL	
Total Excluding Tax	decimal (18,2)	NULL	
Tax Amount	decimal (18,2)	NULL	
Total Including Tax	decimal (18,2)	NULL	
Outstanding Balance	decimal (18,2)	NULL	
Is Finalized	bit	NULL	
Lineage Key	int	NULL	

Proposition 06: Return each customers 3 most recent received payments.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

First find the keys of transaction types for Received Customer Payments. Use these values to filter Transactions by utilizing their Transaction Type Key, in order to find all Received Customer Payment Transactions. Return the top 3 for the specific customer based on the Date Key (most recent). Repeat this process for all customers.

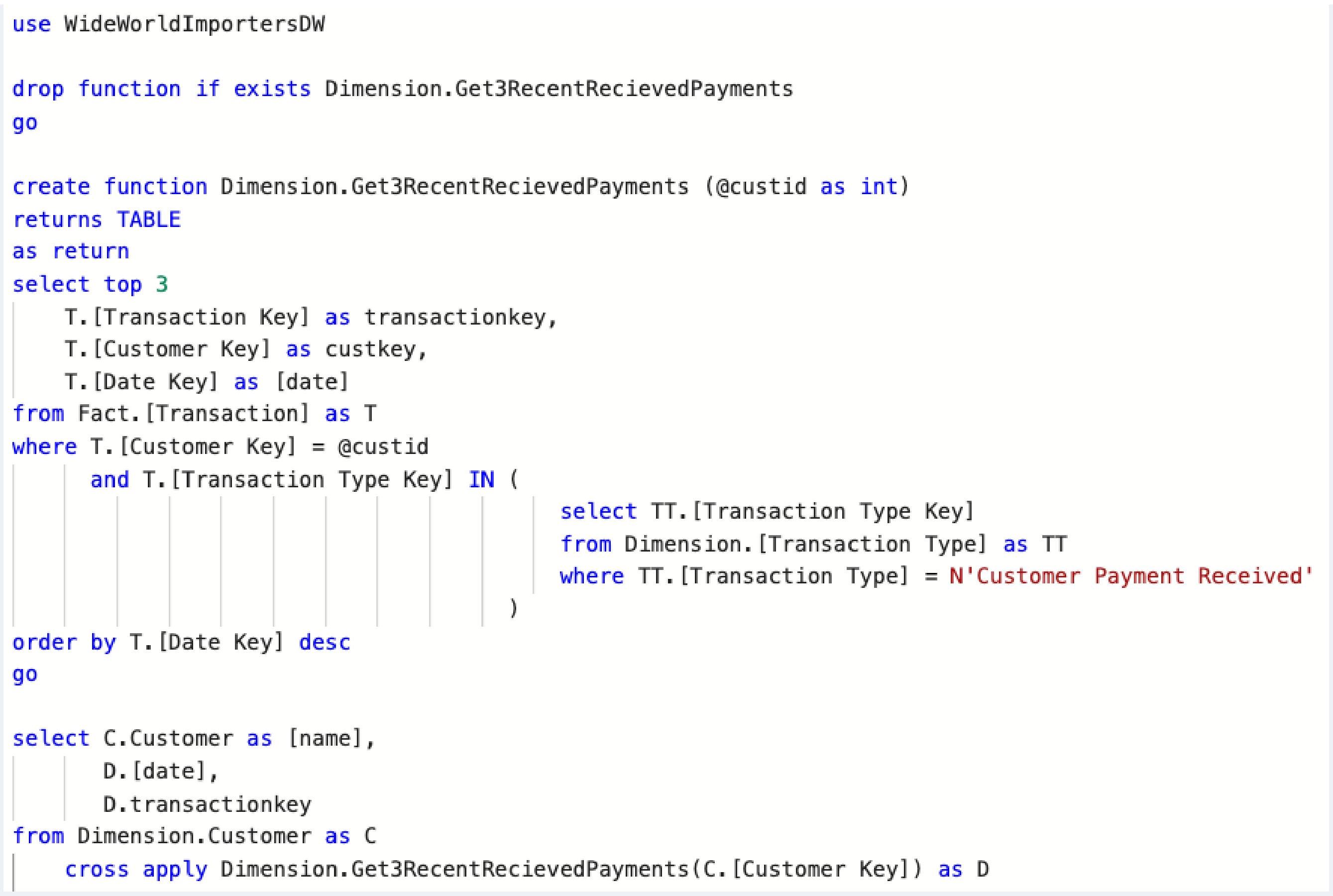
Project following columns from their respective tables in the select clause

Table Name	Column Name
Fact.Transaction	TransactionKey CustomerKey DateKey
Dimension.TransactionType	TransactionTypeKey
Dimension.Get3RecentRecievedPayments (function returns table)	transactionkey custkey date
Dimension.Customer	Customer

Order By

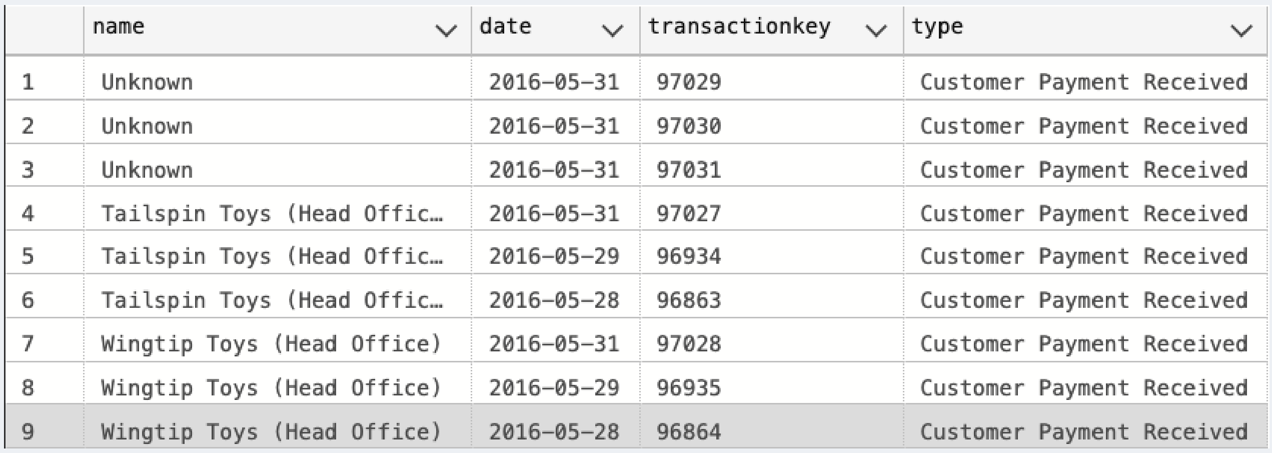
Table Name	Column Name	Sort Order
Fact.Transaction	DateKey	DESC

Problem Solving Query



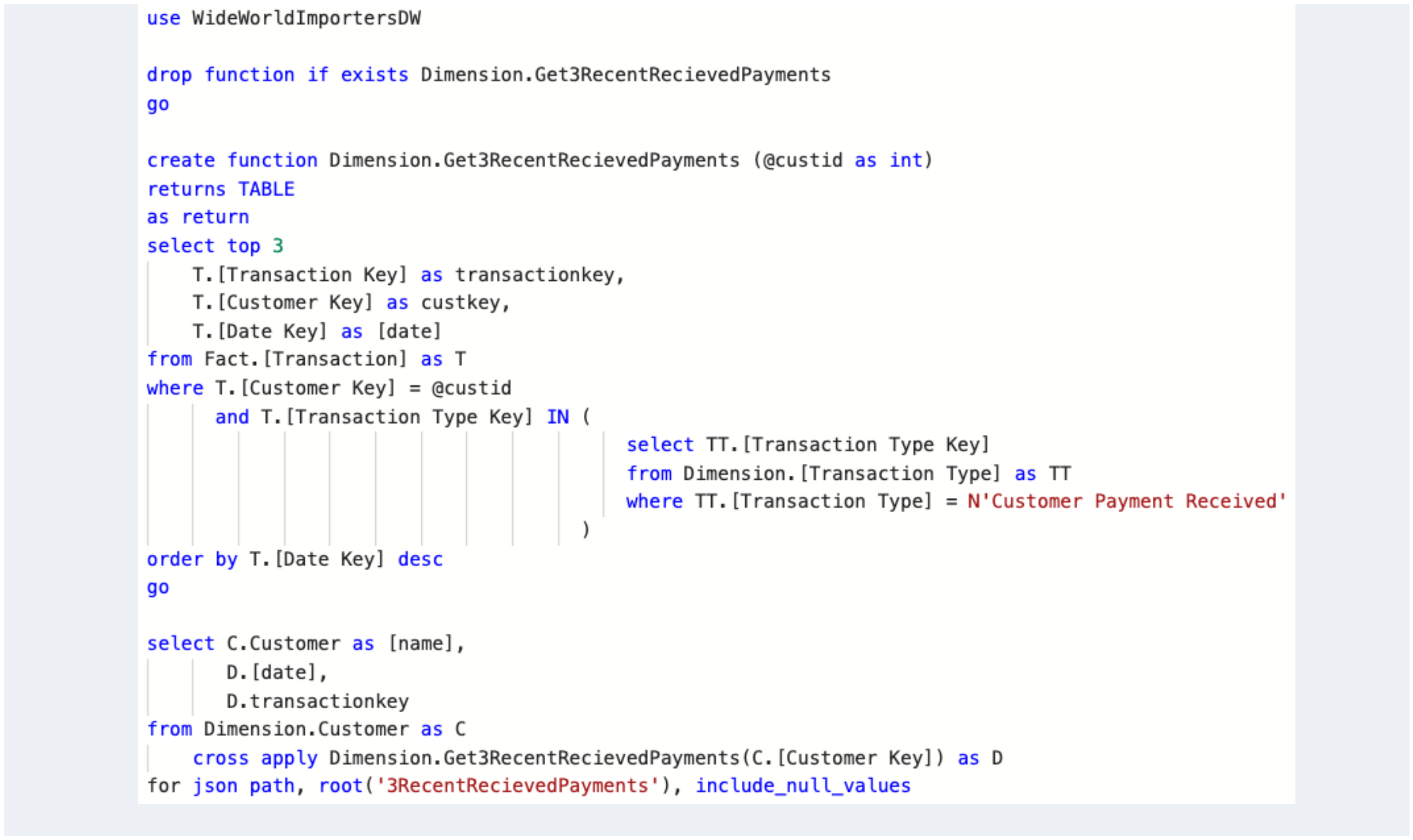
```sql
use WideWorldImportersDW

drop function if exists Dimension.Get3RecentRecievedPayments
go

create function Dimension.Get3RecentRecievedPayments (@custid as int)
returns TABLE
as return
select top 3
    T.[Transaction Key] as transactionkey,
    T.[Customer Key] as custkey,
    T.[Date Key] as [date]
from Fact.[Transaction] as T
where T.[Customer Key] = @custid
      and T.[Transaction Type Key] IN (
                                      select TT.[Transaction Type Key]
                                      from Dimension.[Transaction Type] as TT
                                      where TT.[Transaction Type] = N'Customer Payment Received'
                                     )
order by T.[Date Key] desc
go

select C.Customer as [name],
       D.[date],
       D.transactionkey
from Dimension.Customer as C
    cross apply Dimension.Get3RecentRecievedPayments(C.[Customer Key]) as D
```

Sample Relational Output with total number of rows returned (9)

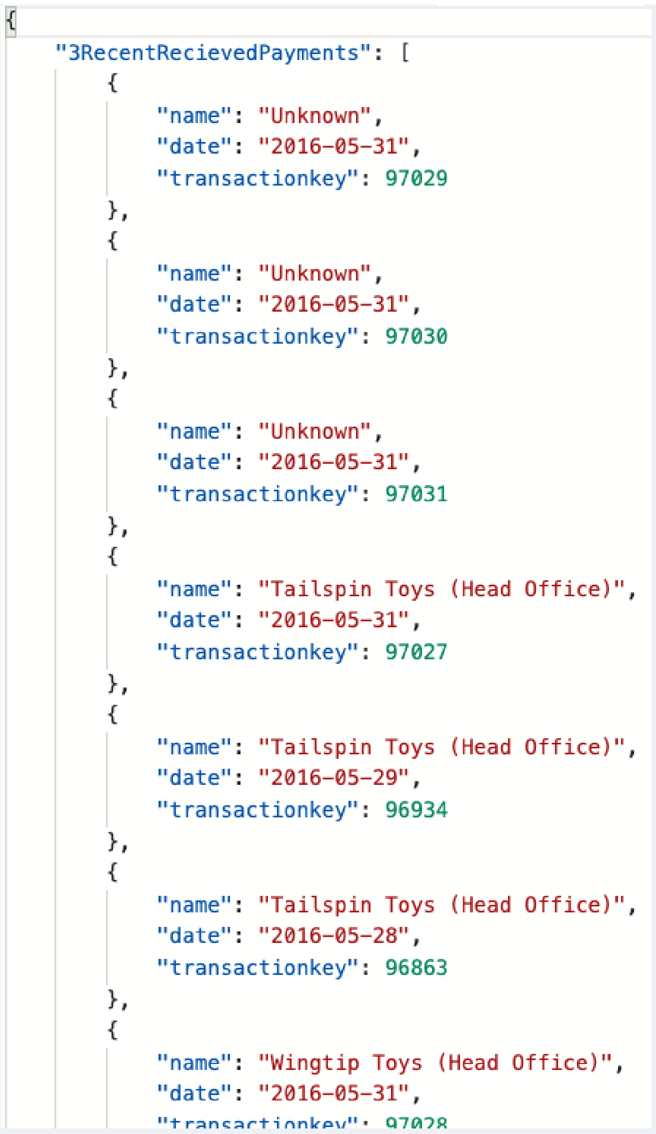


	name	date	transactionkey	type
1	Unknown	2016-05-31	97029	Customer Payment Received
2	Unknown	2016-05-31	97030	Customer Payment Received
3	Unknown	2016-05-31	97031	Customer Payment Received
4	Tailspin Toys (Head Offic…	2016-05-31	97027	Customer Payment Received
5	Tailspin Toys (Head Offic…	2016-05-29	96934	Customer Payment Received
6	Tailspin Toys (Head Offic…	2016-05-28	96863	Customer Payment Received
7	Wingtip Toys (Head Office)	2016-05-31	97028	Customer Payment Received
8	Wingtip Toys (Head Office)	2016-05-29	96935	Customer Payment Received
9	Wingtip Toys (Head Office)	2016-05-28	96864	Customer Payment Received

Sample JSON Output with total number of rows returned (9)



```sql
use WideWorldImportersDW

drop function if exists Dimension.Get3RecentRecievedPayments
go

create function Dimension.Get3RecentRecievedPayments (@custid as int)
returns TABLE
as return
select top 3
    T.[Transaction Key] as transactionkey,
    T.[Customer Key] as custkey,
    T.[Date Key] as [date]
from Fact.[Transaction] as T
where T.[Customer Key] = @custid
      and T.[Transaction Type Key] IN (
                                        select TT.[Transaction Type Key]
                                        from Dimension.[Transaction Type] as TT
                                        where TT.[Transaction Type] = N'Customer Payment Received'
                                      )
order by T.[Date Key] desc
go

select C.Customer as [name],
       D.[date],
       D.transactionkey
from Dimension.Customer as C
    cross apply Dimension.Get3RecentRecievedPayments(C.[Customer Key]) as D
for json path, root('3RecentRecievedPayments'), include_null_values
```

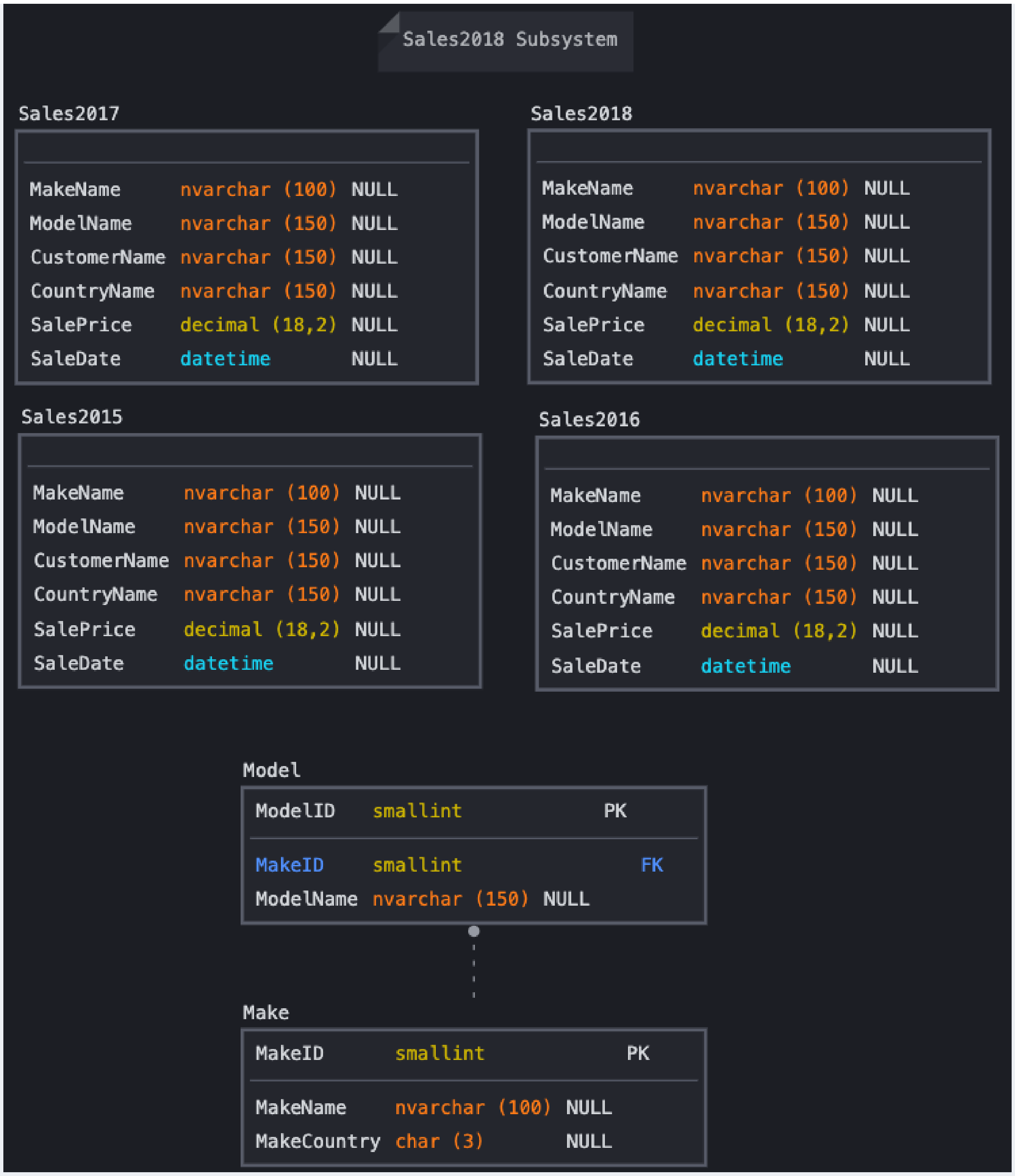


```
{
    "3RecentRecievedPayments": [
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97029
        },
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97030
        },
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97031
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-31",
            "transactionkey": 97027
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-29",
            "transactionkey": 96934
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-28",
            "transactionkey": 96863
        },
        {
            "name": "Wingtip Toys (Head Office)",
            "date": "2016-05-31",
            "transactionkey": 97028
```

CORRECTION

SIMPLE

Example of Sales2018 sub-system in PrestigeCarsOriginal



Sales2017

MakeName	nvarchar (100)	NULL
ModelName	nvarchar (150)	NULL
CustomerName	nvarchar (150)	NULL
CountryName	nvarchar (150)	NULL
SalePrice	decimal (18,2)	NULL
SaleDate	datetime	NULL

Sales2018

MakeName	nvarchar (100)	NULL
ModelName	nvarchar (150)	NULL
CustomerName	nvarchar (150)	NULL
CountryName	nvarchar (150)	NULL
SalePrice	decimal (18,2)	NULL
SaleDate	datetime	NULL

Sales2015

MakeName	nvarchar (100)	NULL
ModelName	nvarchar (150)	NULL
CustomerName	nvarchar (150)	NULL
CountryName	nvarchar (150)	NULL
SalePrice	decimal (18,2)	NULL
SaleDate	datetime	NULL

Sales2016

MakeName	nvarchar (100)	NULL
ModelName	nvarchar (150)	NULL
CustomerName	nvarchar (150)	NULL
CountryName	nvarchar (150)	NULL
SalePrice	decimal (18,2)	NULL
SaleDate	datetime	NULL

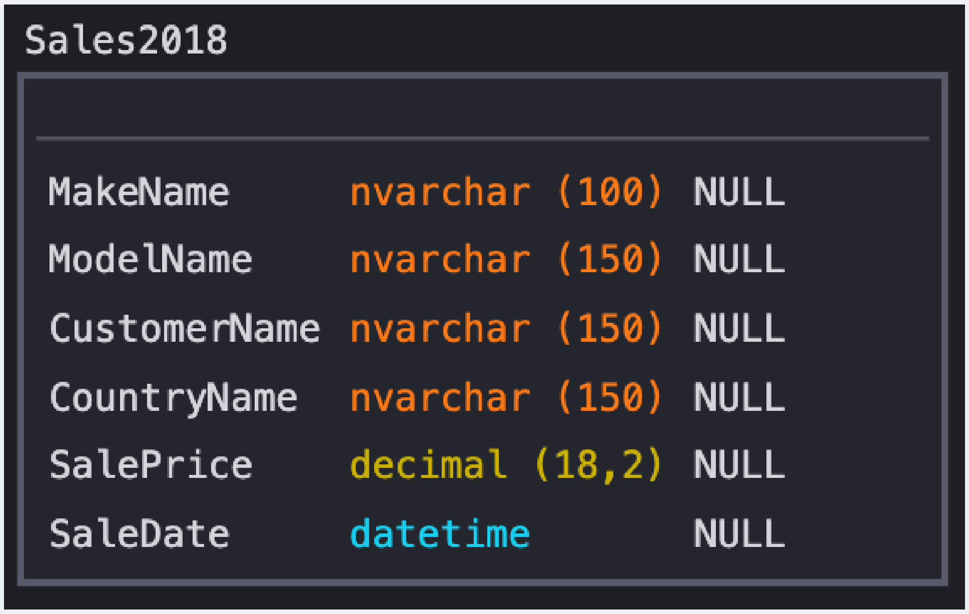
Model

ModelID	smallint	PK
MakeID	smallint	FK
ModelName	nvarchar (150)	NULL

Make

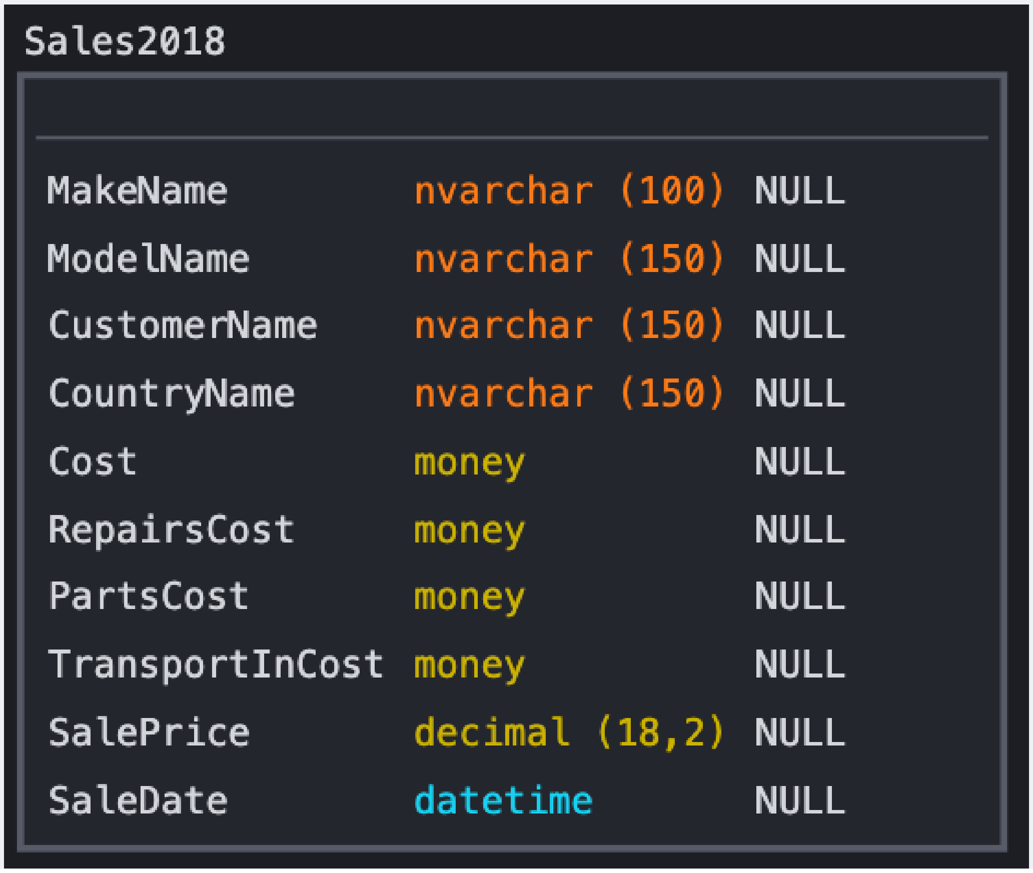
MakeID	smallint	PK
MakeName	nvarchar (100)	NULL
MakeCountry	char (3)	NULL

Diagram of Tables



```
Sales2018

MakeName     nvarchar (100) NULL
ModelName    nvarchar (150) NULL
CustomerName nvarchar (150) NULL
CountryName  nvarchar (150) NULL
SalePrice    decimal (18,2) NULL
SaleDate     datetime       NULL
```

Columns from Standard View



```
Sales2018

MakeName         nvarchar (100) NULL
ModelName        nvarchar (150) NULL
CustomerName     nvarchar (150) NULL
CountryName      nvarchar (150) NULL
Cost             money          NULL
RepairsCost      money          NULL
PartsCost        money          NULL
TransportInCost  money          NULL
SalePrice        decimal (18,2) NULL
SaleDate         datetime       NULL
```

Proposition C04: Return the total sales price for each make of cars sold in 2018.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

An example of a make of car would be “BMW” or “Bugatti”. The total sales price would be the sum of all sales prices in 2018 for the specific make. Necessary to use a group by.

The correction is made by noticing we receive the total sales for each specific model rather than make type. In the corrected version, we should only group by makename to get the total sales for that entire make of cars, rather than subsections of that make via model.

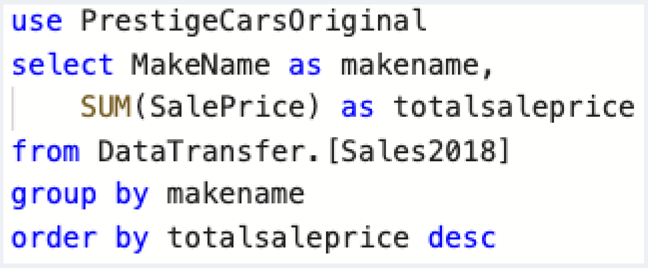
Project following columns from their respective tables in the select clause

Table Name	Column Name
DataTransfer.Sales2018	MakeName SalePrice

Order By

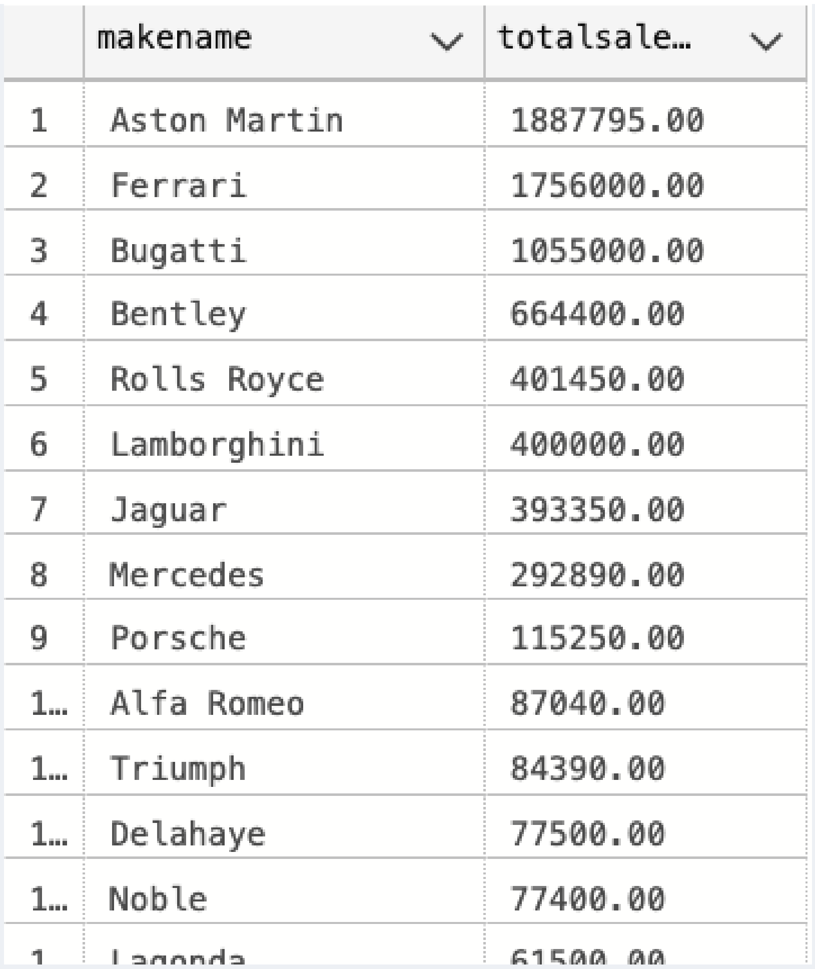
Table Name	Column Name	Sort Order
DataTransfer.Sales2018	totalsalesprice	DESC

Problem Solving Query



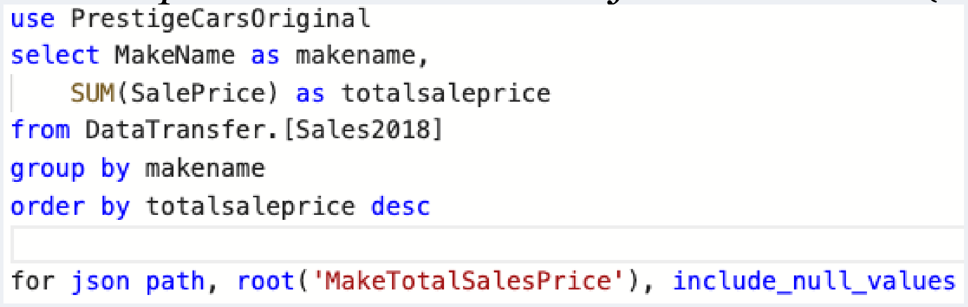
```sql
use PrestigeCarsOriginal
select MakeName as makename,
    SUM(SalePrice) as totalsaleprice
from DataTransfer.[Sales2018]
group by makename
order by totalsaleprice desc
```

Sample Relational Output with total number of rows returned (21)

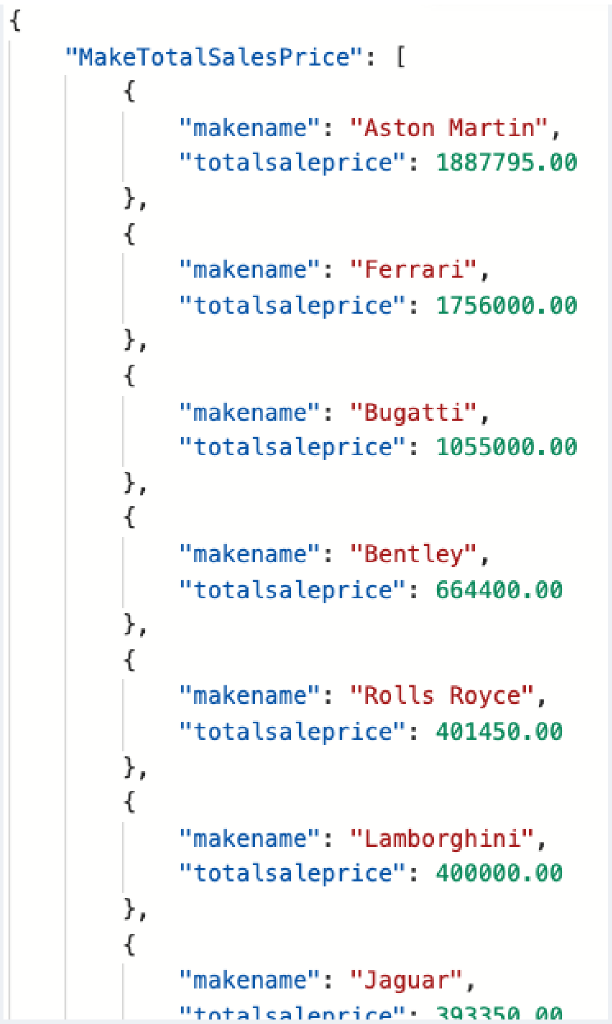


	makename	totalsale…
1	Aston Martin	1887795.00
2	Ferrari	1756000.00
3	Bugatti	1055000.00
4	Bentley	664400.00
5	Rolls Royce	401450.00
6	Lamborghini	400000.00
7	Jaguar	393350.00
8	Mercedes	292890.00
9	Porsche	115250.00
1…	Alfa Romeo	87040.00
1…	Triumph	84390.00
1…	Delahaye	77500.00
1…	Noble	77400.00
1	Lagonda	61500.00

Sample JSON Output with total number of rows returned (21)



```
use PrestigeCarsOriginal
select MakeName as makename,
    SUM(SalePrice) as totalsaleprice
from DataTransfer.[Sales2018]
group by makename
order by totalsaleprice desc

for json path, root('MakeTotalSalesPrice'), include_null_values
```



```
{
    "MakeTotalSalesPrice": [
        {
            "makename": "Aston Martin",
            "totalsaleprice": 1887795.00
        },
        {
            "makename": "Ferrari",
            "totalsaleprice": 1756000.00
        },
        {
            "makename": "Bugatti",
            "totalsaleprice": 1055000.00
        },
        {
            "makename": "Bentley",
            "totalsaleprice": 664400.00
        },
        {
            "makename": "Rolls Royce",
            "totalsaleprice": 401450.00
        },
        {
            "makename": "Lamborghini",
            "totalsaleprice": 400000.00
        },
        {
            "makename": "Jaguar",
            "totalsaleprice": 393350.00
```

MEDIUM

Example of WorkOrder sub-system in AdventureWorks2017

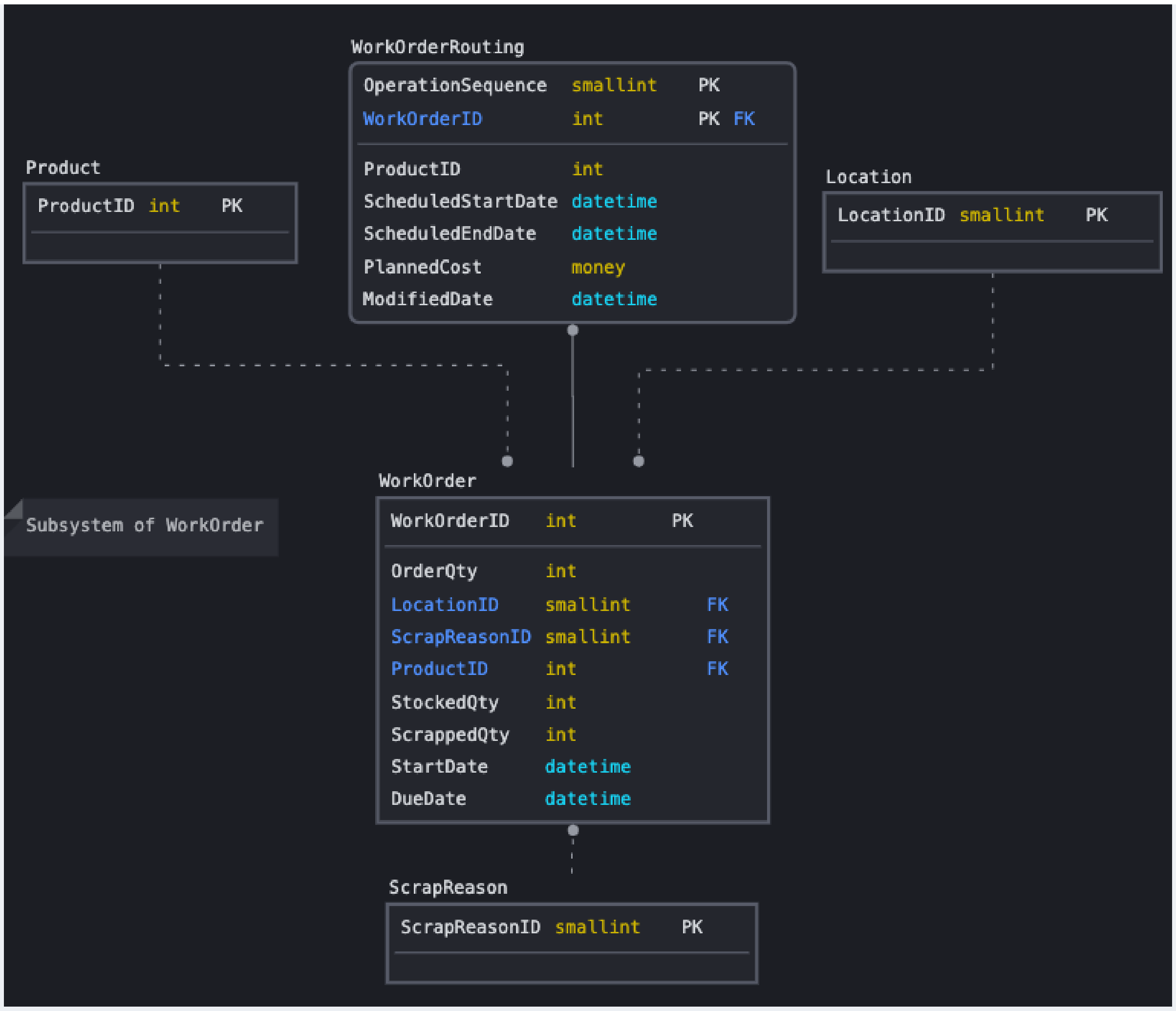
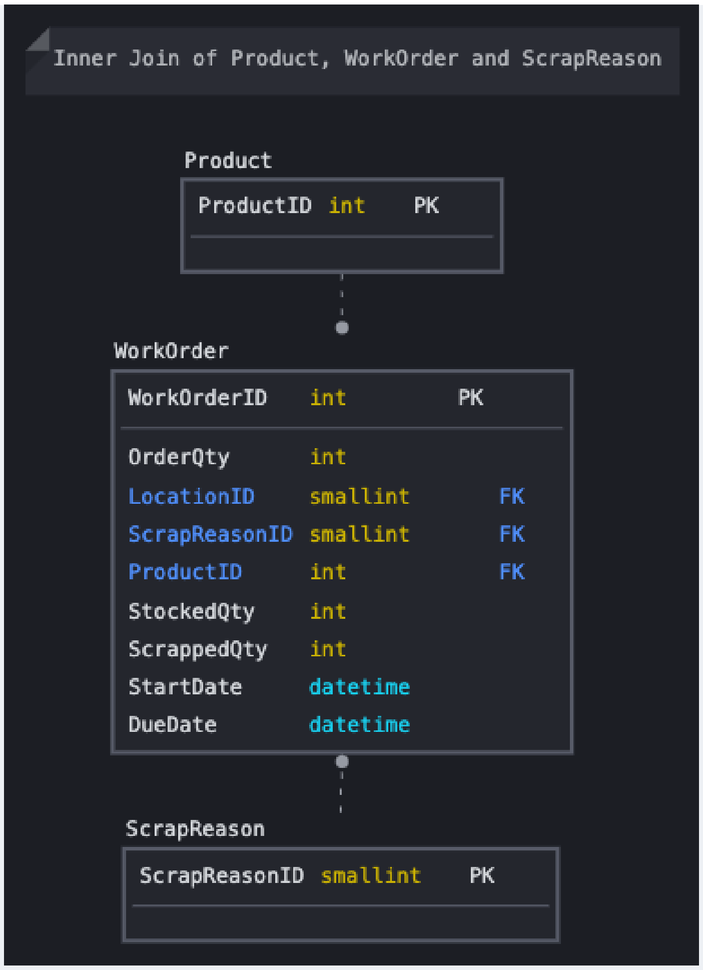
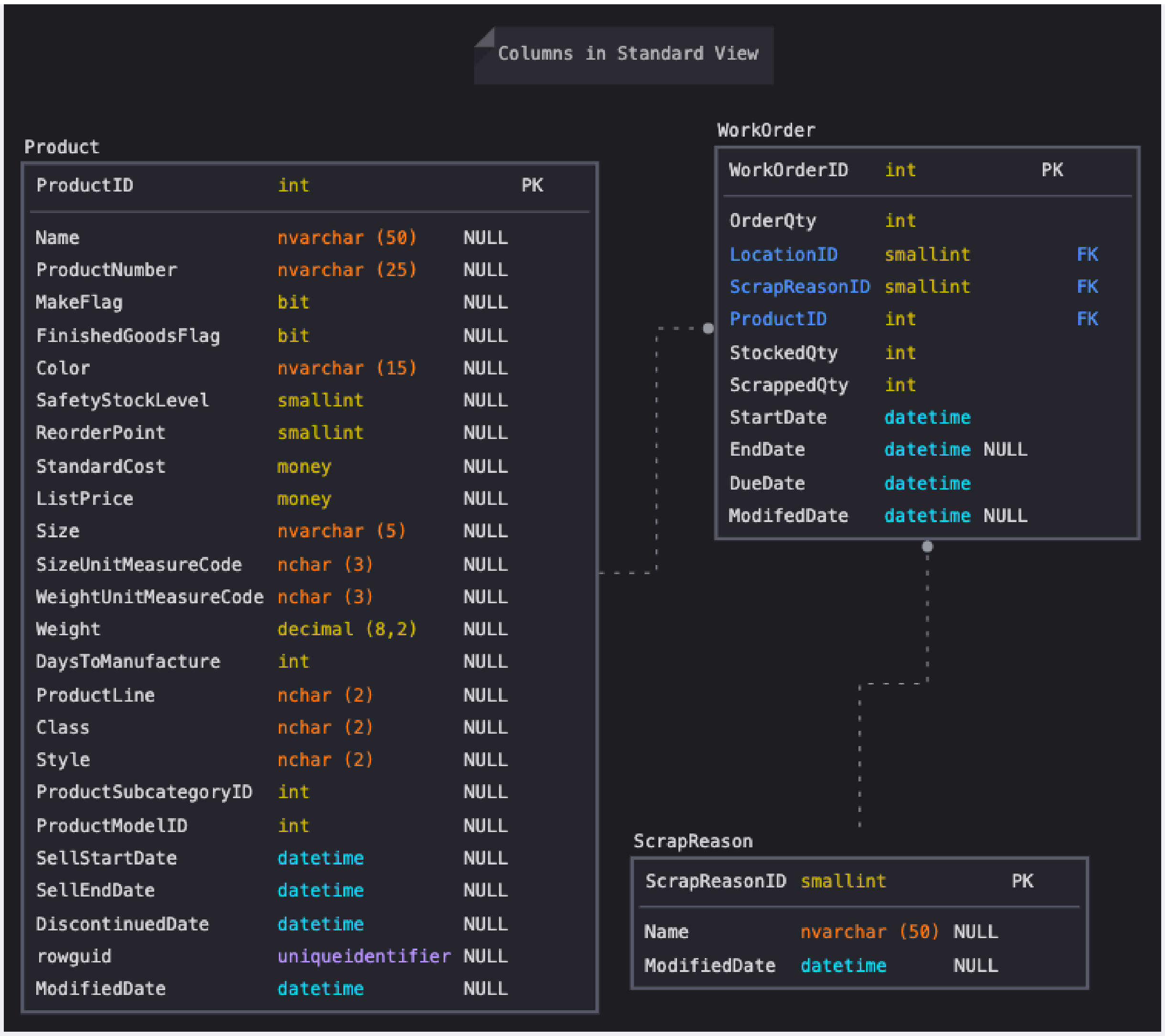


Diagram of Tables



Columns from Standard View



Product

ProductID	int		PK
Name	nvarchar (50)	NULL	
ProductNumber	nvarchar (25)	NULL	
MakeFlag	bit	NULL	
FinishedGoodsFlag	bit	NULL	
Color	nvarchar (15)	NULL	
SafetyStockLevel	smallint	NULL	
ReorderPoint	smallint	NULL	
StandardCost	money	NULL	
ListPrice	money	NULL	
Size	nvarchar (5)	NULL	
SizeUnitMeasureCode	nchar (3)	NULL	
WeightUnitMeasureCode	nchar (3)	NULL	
Weight	decimal (8,2)	NULL	
DaysToManufacture	int	NULL	
ProductLine	nchar (2)	NULL	
Class	nchar (2)	NULL	
Style	nchar (2)	NULL	
ProductSubcategoryID	int	NULL	
ProductModelID	int	NULL	
SellStartDate	datetime	NULL	
SellEndDate	datetime	NULL	
DiscontinuedDate	datetime	NULL	
rowguid	uniqueidentifier	NULL	
ModifiedDate	datetime	NULL	

WorkOrder

WorkOrderID	int		PK
OrderQty	int		
LocationID	smallint		FK
ScrapReasonID	smallint		FK
ProductID	int		FK
StockedQty	int		
ScrappedQty	int		
StartDate	datetime		
EndDate	datetime	NULL	
DueDate	datetime		
ModifedDate	datetime	NULL	

ScrapReason

ScrapReasonID	smallint		PK
Name	nvarchar (50)	NULL	
ModifiedDate	datetime	NULL	

Proposition C05: Return all scrapped product names with the reason they were scrapped.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

The correction comes from the readability of the query and logic. A where clause is unnecessary and clogs the query, when we can use the same condition in the on clause of the join. Rather than deriving a scrapped table and joining it with the Product table, the WorkOrder and Product can be joined directly with a strong filter to find the desired property.

Project following columns from their respective tables in the select clause

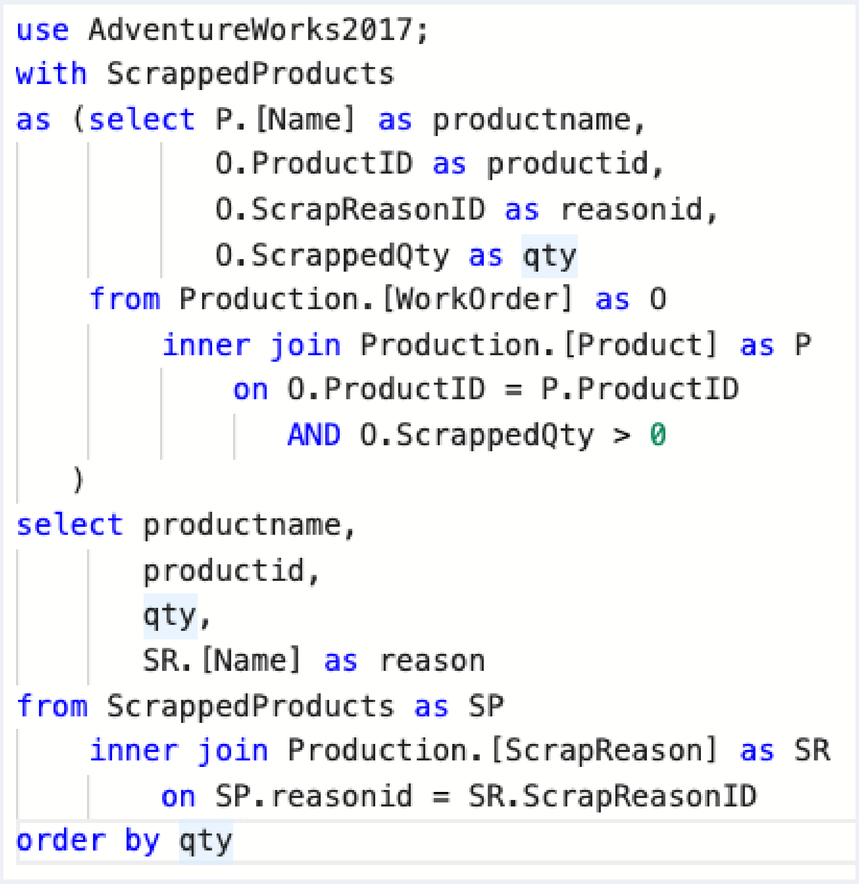
Table Name	Column Name

Production.WorkOrder	ScrappedQty ScrapReasonID ProductID
Production.Product	Name
ScrappedProducts	productname productid reasonid qty
Production.[ScrapReason]	Name

Order By

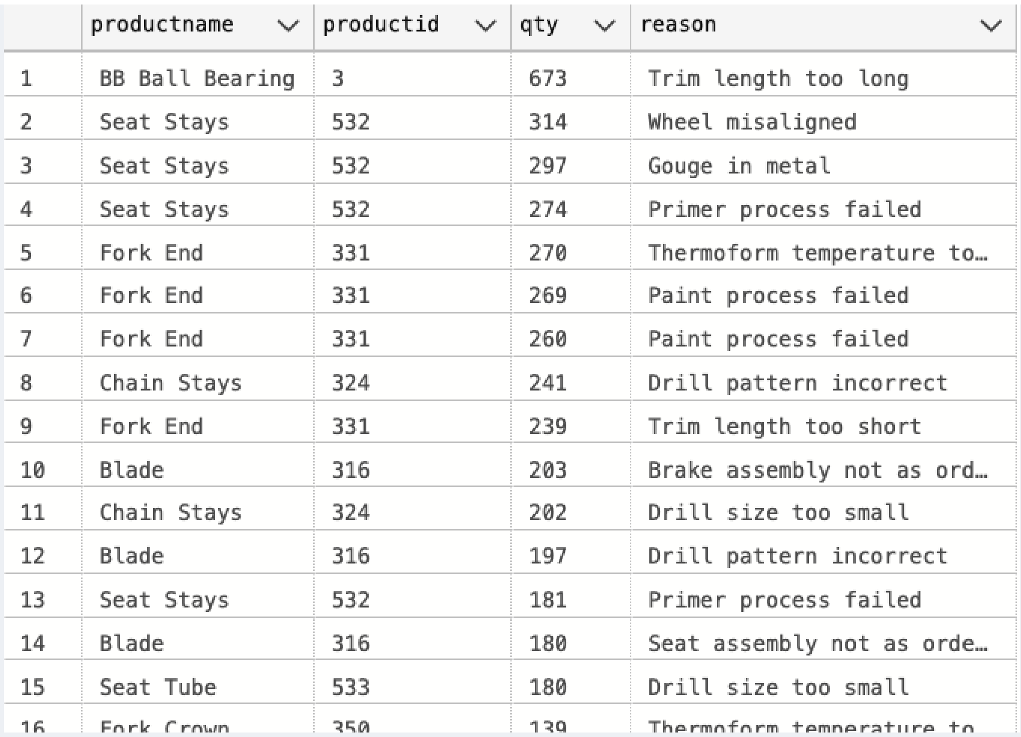
Table Name	Column Name	Sort Order
Production.WorkOrder	ScrappedQty	DESC

Problem Solving Query



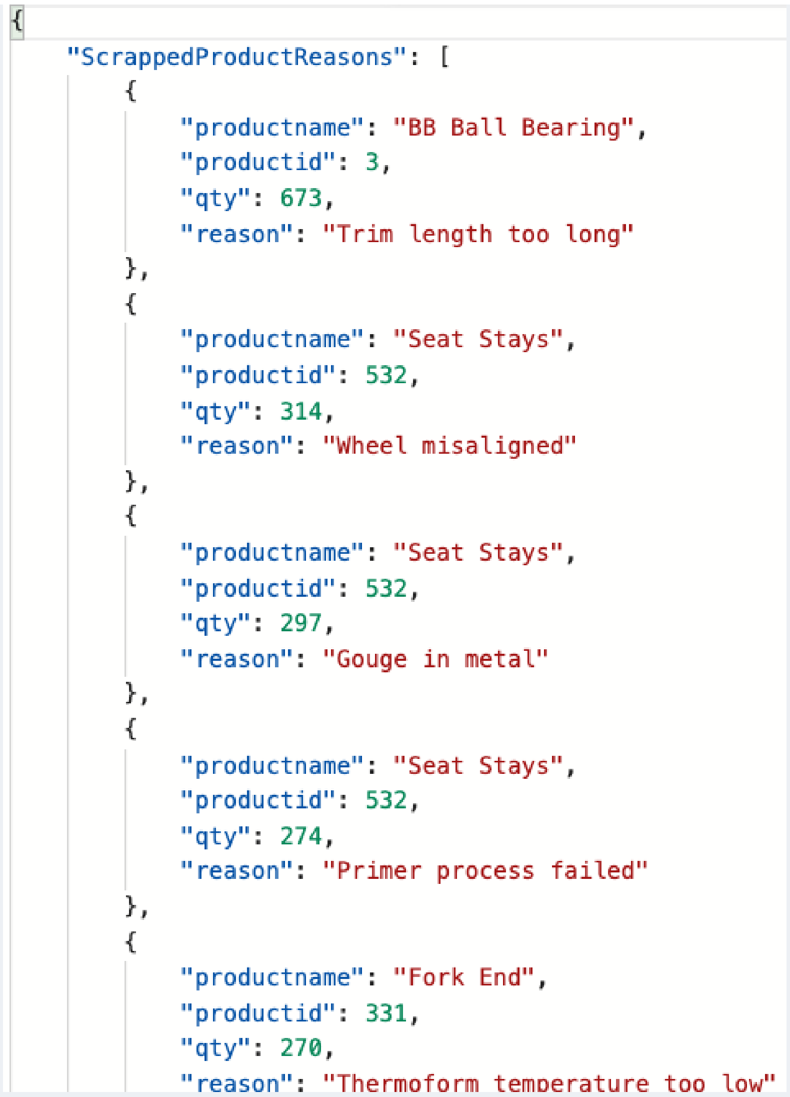
```
use AdventureWorks2017;
with ScrappedProducts
as (select P.[Name] as productname,
           O.ProductID as productid,
           O.ScrapReasonID as reasonid,
           O.ScrappedQty as qty
    from Production.[WorkOrder] as O
        inner join Production.[Product] as P
            on O.ProductID = P.ProductID
               AND O.ScrappedQty > 0
   )
select productname,
       productid,
       qty,
       SR.[Name] as reason
from ScrappedProducts as SP
    inner join Production.[ScrapReason] as SR
        on SP.reasonid = SR.ScrapReasonID
order by qty
```

Sample Relational Output with total number of rows returned (729)



	productname	productid	qty	reason
1	BB Ball Bearing	3	673	Trim length too long
2	Seat Stays	532	314	Wheel misaligned
3	Seat Stays	532	297	Gouge in metal
4	Seat Stays	532	274	Primer process failed
5	Fork End	331	270	Thermoform temperature to…
6	Fork End	331	269	Paint process failed
7	Fork End	331	260	Paint process failed
8	Chain Stays	324	241	Drill pattern incorrect
9	Fork End	331	239	Trim length too short
10	Blade	316	203	Brake assembly not as ord…
11	Chain Stays	324	202	Drill size too small
12	Blade	316	197	Drill pattern incorrect
13	Seat Stays	532	181	Primer process failed
14	Blade	316	180	Seat assembly not as orde…
15	Seat Tube	533	180	Drill size too small
16	Fork Crown	350	139	Thermoform temperature to

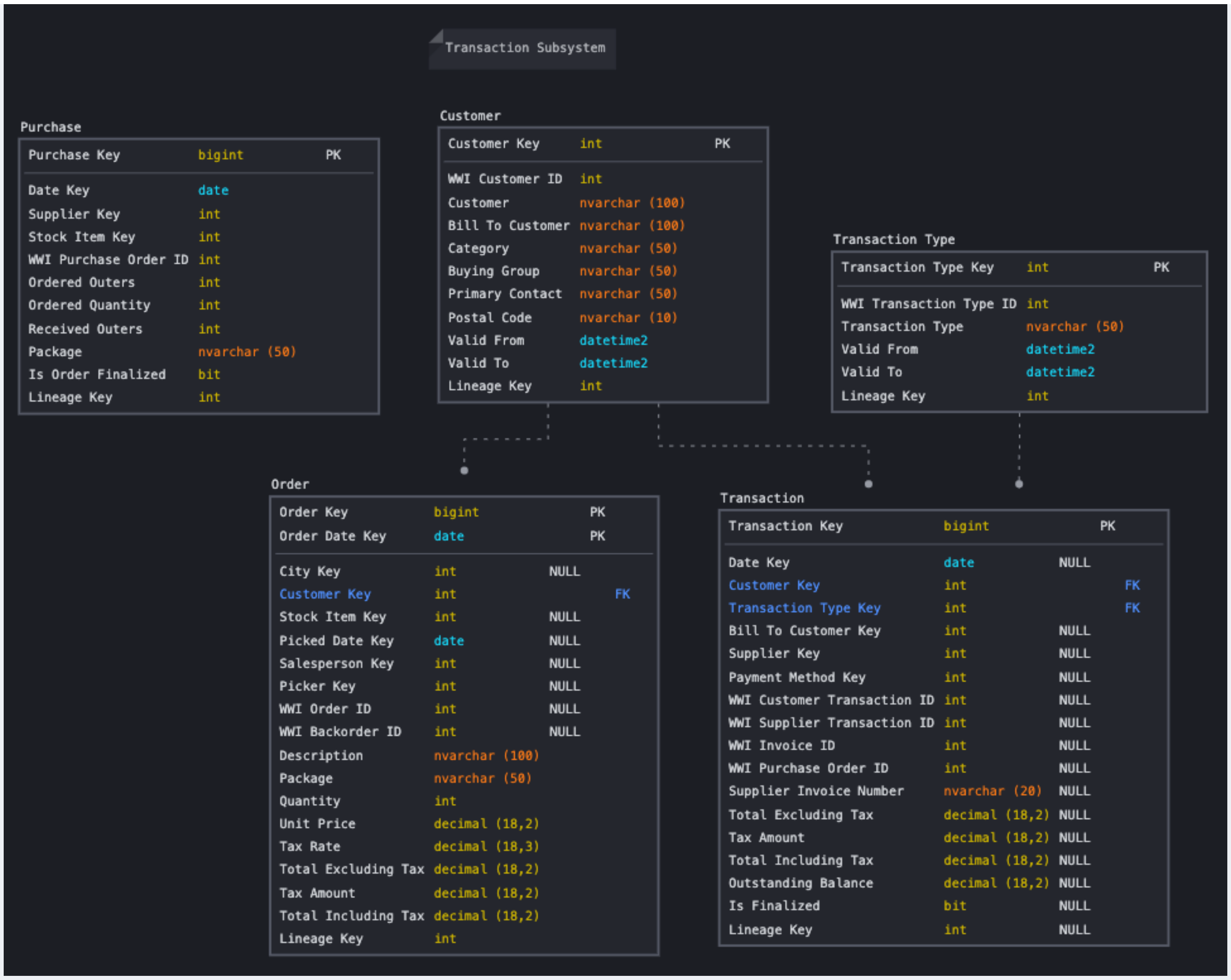
Sample JSON Output with total number of rows returned (729)



```
{
    "ScrappedProductReasons": [
        {
            "productname": "BB Ball Bearing",
            "productid": 3,
            "qty": 673,
            "reason": "Trim length too long"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 314,
            "reason": "Wheel misaligned"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 297,
            "reason": "Gouge in metal"
        },
        {
            "productname": "Seat Stays",
            "productid": 532,
            "qty": 274,
            "reason": "Primer process failed"
        },
        {
            "productname": "Fork End",
            "productid": 331,
            "qty": 270,
            "reason": "Thermoform temperature too low"
```

COMPLEX

Example of Transaction sub-system in WideWorldImportersDW



Purchase

Purchase Key	bigint	PK
Date Key	date	
Supplier Key	int	
Stock Item Key	int	
WWI Purchase Order ID	int	
Ordered Outers	int	
Ordered Quantity	int	
Received Outers	int	
Package	nvarchar (50)	
Is Order Finalized	bit	
Lineage Key	int	

Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

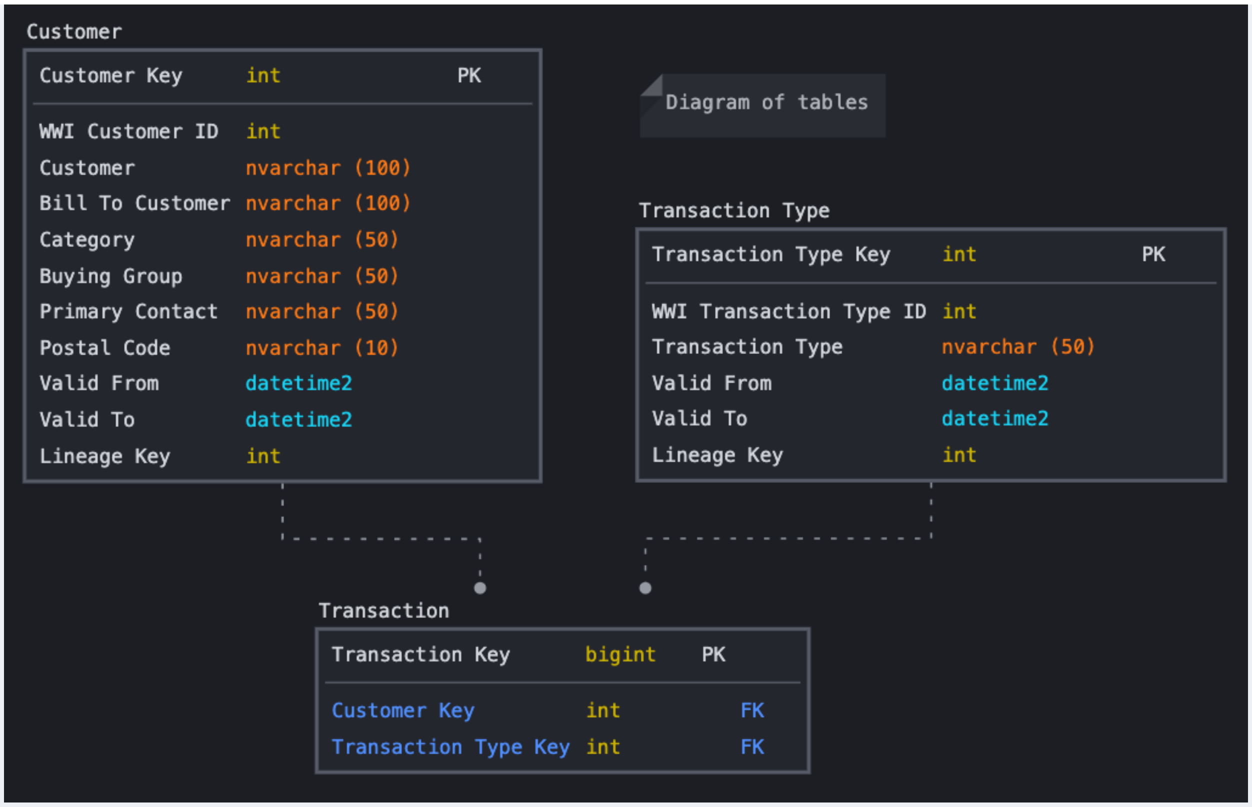
Order

Order Key	bigint	PK	
Order Date Key	date	PK	
City Key	int	NULL	
Customer Key	int		FK
Stock Item Key	int	NULL	
Picked Date Key	date	NULL	
Salesperson Key	int	NULL	
Picker Key	int	NULL	
WWI Order ID	int	NULL	
WWI Backorder ID	int	NULL	
Description	nvarchar (100)		
Package	nvarchar (50)		
Quantity	int		
Unit Price	decimal (18,2)		
Tax Rate	decimal (18,3)		
Total Excluding Tax	decimal (18,2)		
Tax Amount	decimal (18,2)		
Total Including Tax	decimal (18,2)		
Lineage Key	int		

Transaction

Transaction Key	bigint	PK	
Date Key	date	NULL	
Customer Key	int		FK
Transaction Type Key	int		FK
Bill To Customer Key	int	NULL	
Supplier Key	int	NULL	
Payment Method Key	int	NULL	
WWI Customer Transaction ID	int	NULL	
WWI Supplier Transaction ID	int	NULL	
WWI Invoice ID	int	NULL	
WWI Purchase Order ID	int	NULL	
Supplier Invoice Number	nvarchar (20)	NULL	
Total Excluding Tax	decimal (18,2)	NULL	
Tax Amount	decimal (18,2)	NULL	
Total Including Tax	decimal (18,2)	NULL	
Outstanding Balance	decimal (18,2)	NULL	
Is Finalized	bit	NULL	
Lineage Key	int	NULL	

Diagram of Tables



Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

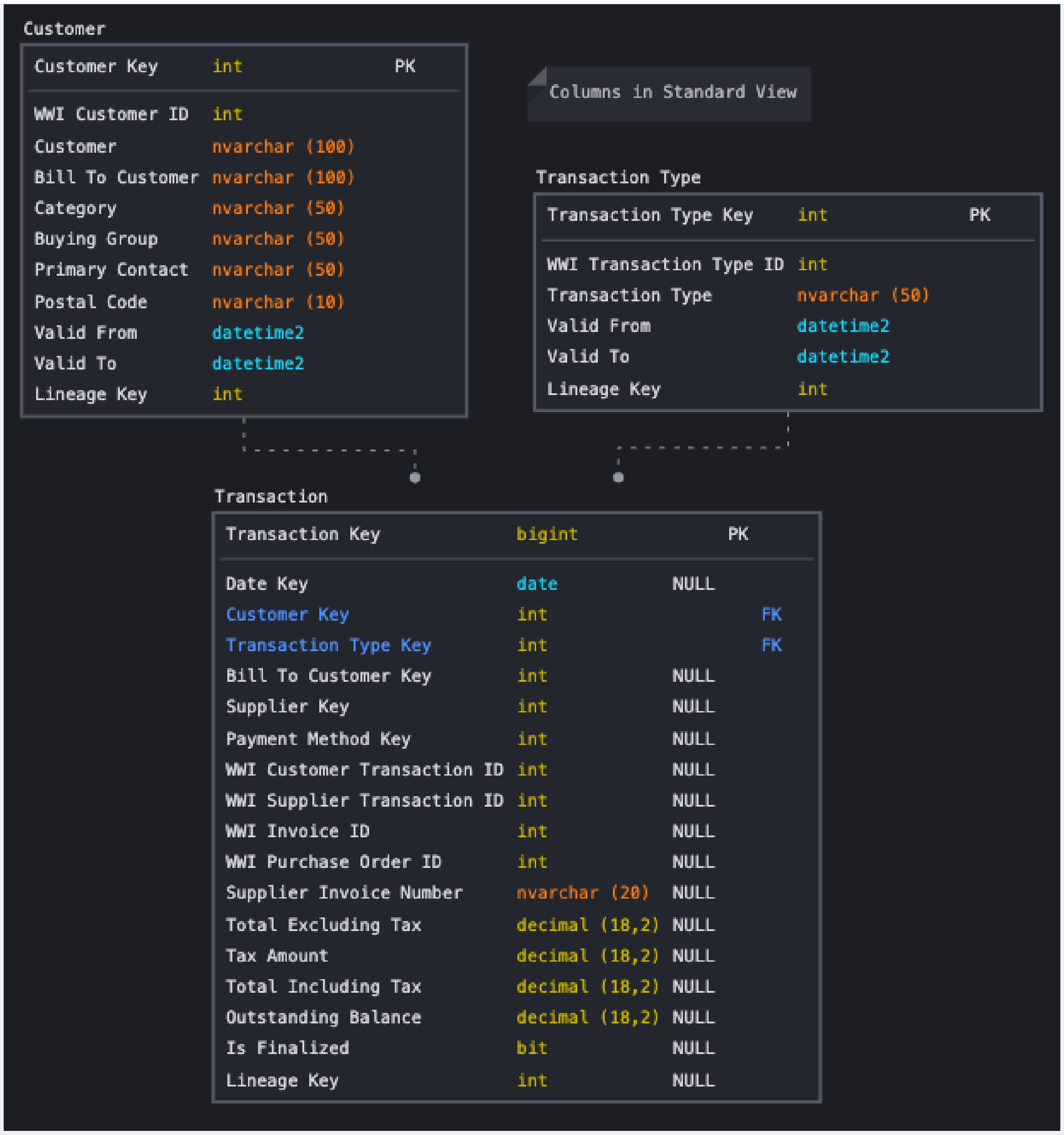
Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction

Transaction Key	bigint	PK
Customer Key	int	FK
Transaction Type Key	int	FK

Columns from Standard View



Customer

Customer Key	int	PK
WWI Customer ID	int	
Customer	nvarchar (100)	
Bill To Customer	nvarchar (100)	
Category	nvarchar (50)	
Buying Group	nvarchar (50)	
Primary Contact	nvarchar (50)	
Postal Code	nvarchar (10)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction Type

Transaction Type Key	int	PK
WWI Transaction Type ID	int	
Transaction Type	nvarchar (50)	
Valid From	datetime2	
Valid To	datetime2	
Lineage Key	int	

Transaction

Transaction Key	bigint		PK
Date Key	date	NULL	
Customer Key	int		FK
Transaction Type Key	int		FK
Bill To Customer Key	int	NULL	
Supplier Key	int	NULL	
Payment Method Key	int	NULL	
WWI Customer Transaction ID	int	NULL	
WWI Supplier Transaction ID	int	NULL	
WWI Invoice ID	int	NULL	
WWI Purchase Order ID	int	NULL	
Supplier Invoice Number	nvarchar (20)	NULL	
Total Excluding Tax	decimal (18,2)	NULL	
Tax Amount	decimal (18,2)	NULL	
Total Including Tax	decimal (18,2)	NULL	
Outstanding Balance	decimal (18,2)	NULL	
Is Finalized	bit	NULL	
Lineage Key	int	NULL	

Proposition C06: Return each customers 3 most recent received payments.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue.

The correction comes in the form of readability and logic. Rather than use a where clause with a multi-valued subquery, we can utilize the fact that one entry for “Customer Payment Received" meaning we can use it as a scalar value in either an on clause or where clause. To decrease clutter and readability, join the Transaction and Transaction Type with an on filter than filters for the matching customer, a matching transaction type key between the two tables, and matches the transaction type to the scalar string value. This allows for a clear where clause.

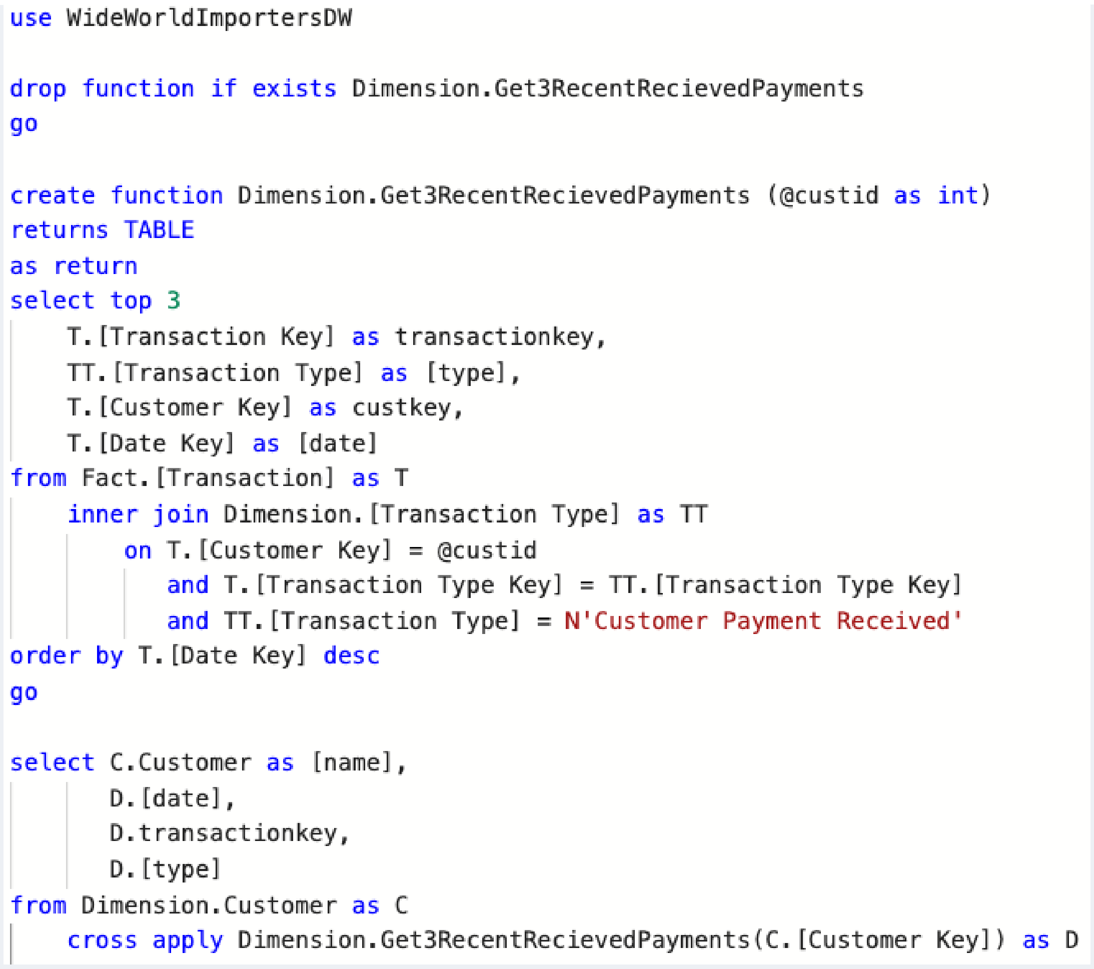
Project following columns from their respective tables in the select clause

Table Name	Column Name
Fact.Transaction	TransactionKey CustomerKey DateKey
Dimension.TransactionType	TransactionType
Dimension.Get3RecentRecievedPayments (function returns table)	transactionkey type custkey date
Dimension.Customer	name

Order By

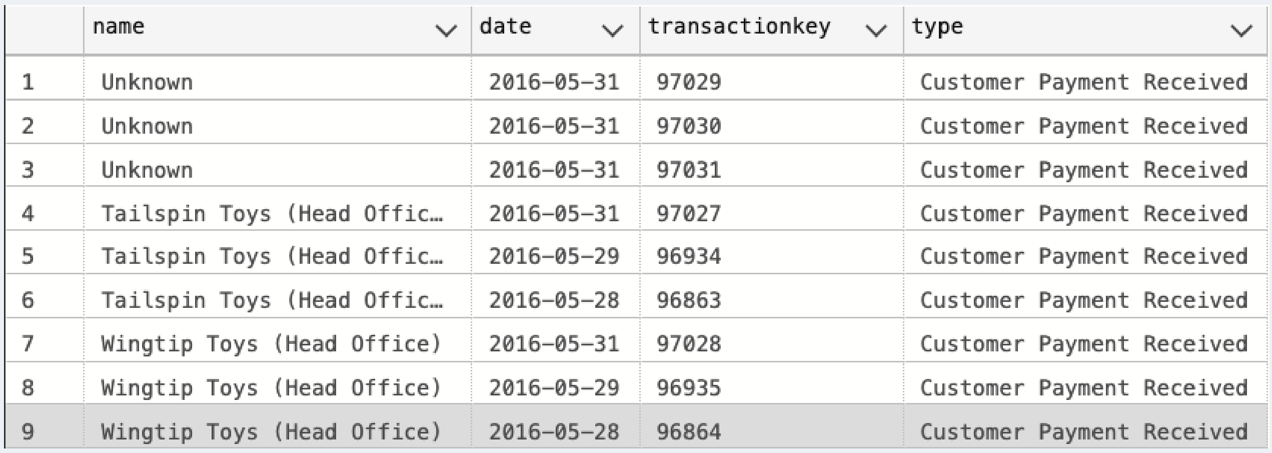
Table Name	Column Name	Sort Order
Fact.Transaction	DateKey	DESC

Problem Solving Query



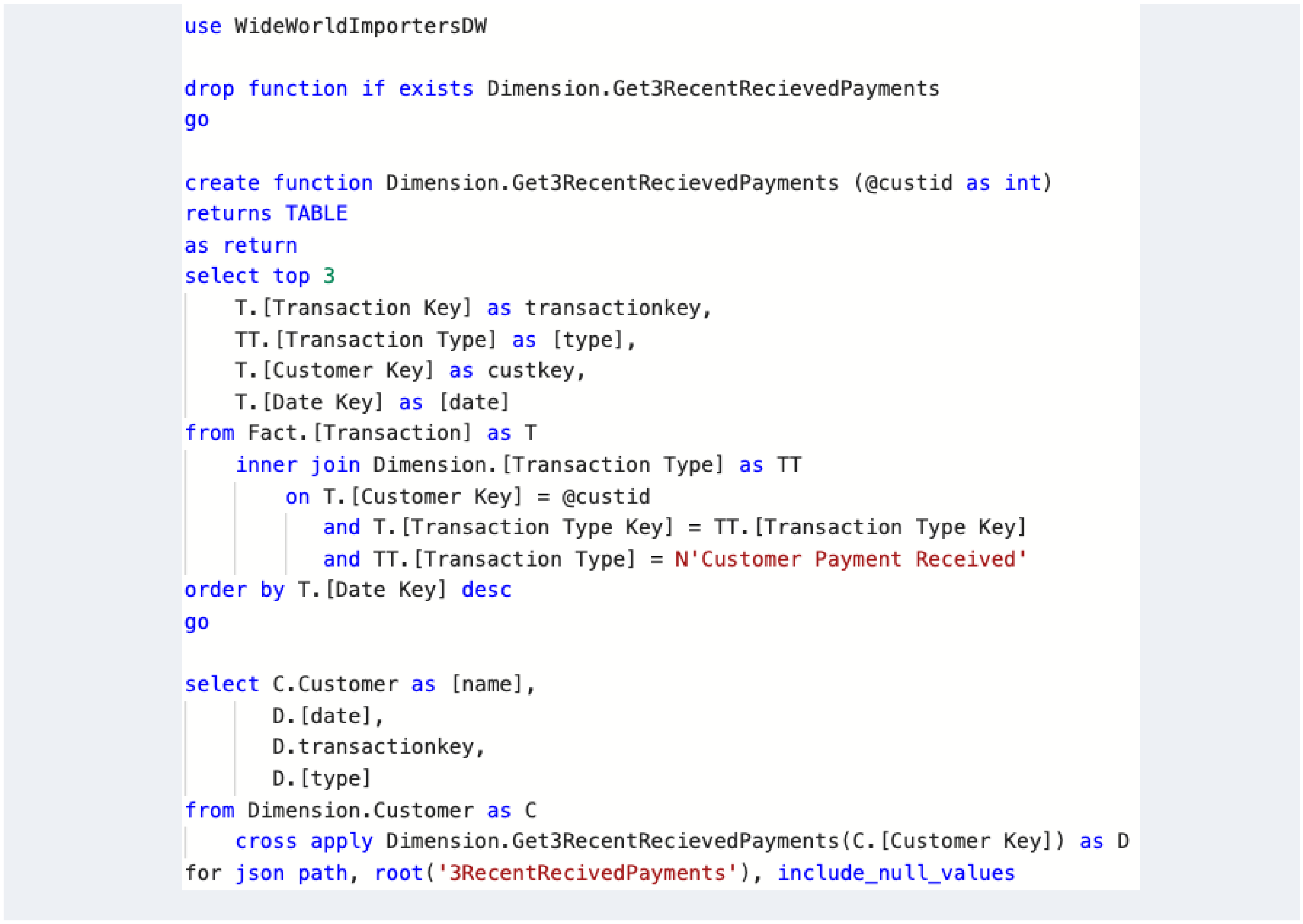
```sql
use WideWorldImportersDW

drop function if exists Dimension.Get3RecentRecievedPayments
go

create function Dimension.Get3RecentRecievedPayments (@custid as int)
returns TABLE
as return
select top 3
    T.[Transaction Key] as transactionkey,
    TT.[Transaction Type] as [type],
    T.[Customer Key] as custkey,
    T.[Date Key] as [date]
from Fact.[Transaction] as T
    inner join Dimension.[Transaction Type] as TT
        on T.[Customer Key] = @custid
           and T.[Transaction Type Key] = TT.[Transaction Type Key]
           and TT.[Transaction Type] = N'Customer Payment Received'
order by T.[Date Key] desc
go

select C.Customer as [name],
       D.[date],
       D.transactionkey,
       D.[type]
from Dimension.Customer as C
    cross apply Dimension.Get3RecentRecievedPayments(C.[Customer Key]) as D
```

Sample Relational Output with total number of rows returned (9)

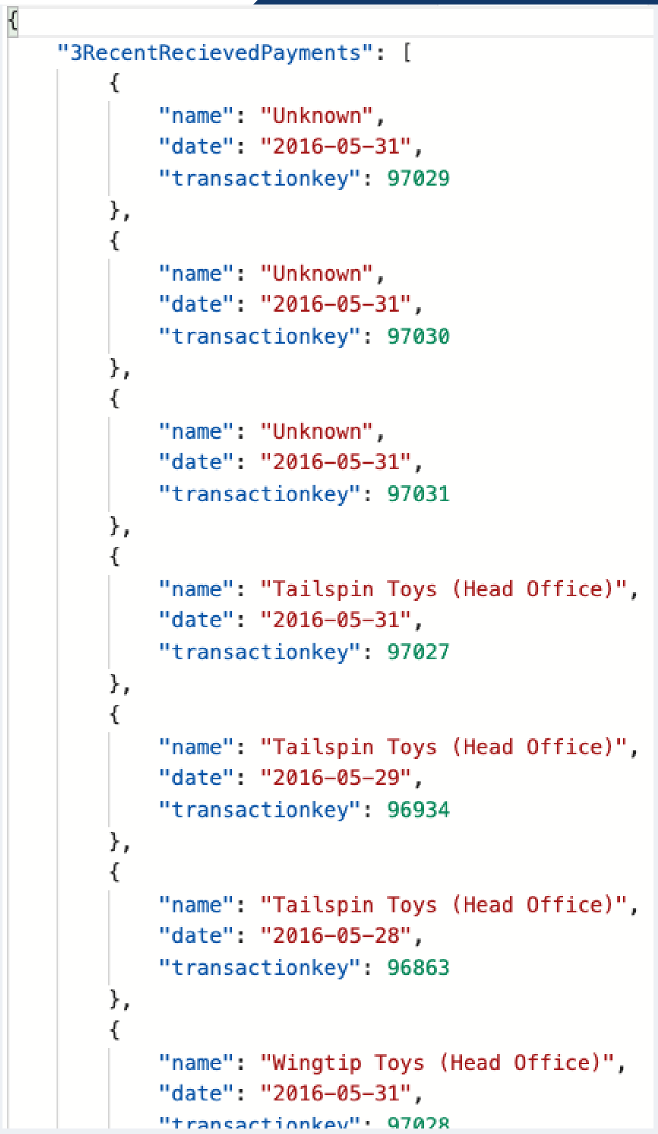


	name	date	transactionkey	type
1	Unknown	2016-05-31	97029	Customer Payment Received
2	Unknown	2016-05-31	97030	Customer Payment Received
3	Unknown	2016-05-31	97031	Customer Payment Received
4	Tailspin Toys (Head Offic…	2016-05-31	97027	Customer Payment Received
5	Tailspin Toys (Head Offic…	2016-05-29	96934	Customer Payment Received
6	Tailspin Toys (Head Offic…	2016-05-28	96863	Customer Payment Received
7	Wingtip Toys (Head Office)	2016-05-31	97028	Customer Payment Received
8	Wingtip Toys (Head Office)	2016-05-29	96935	Customer Payment Received
9	Wingtip Toys (Head Office)	2016-05-28	96864	Customer Payment Received

Sample JSON Output with total number of rows returned (9)



```
use WideWorldImportersDW

drop function if exists Dimension.Get3RecentRecievedPayments
go

create function Dimension.Get3RecentRecievedPayments (@custid as int)
returns TABLE
as return
select top 3
    T.[Transaction Key] as transactionkey,
    TT.[Transaction Type] as [type],
    T.[Customer Key] as custkey,
    T.[Date Key] as [date]
from Fact.[Transaction] as T
    inner join Dimension.[Transaction Type] as TT
        on T.[Customer Key] = @custid
           and T.[Transaction Type Key] = TT.[Transaction Type Key]
           and TT.[Transaction Type] = N'Customer Payment Received'
order by T.[Date Key] desc
go

select C.Customer as [name],
       D.[date],
       D.transactionkey,
       D.[type]
from Dimension.Customer as C
    cross apply Dimension.Get3RecentRecievedPayments(C.[Customer Key]) as D
for json path, root('3RecentRecivedPayments'), include_null_values
```



```
{
    "3RecentRecievedPayments": [
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97029
        },
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97030
        },
        {
            "name": "Unknown",
            "date": "2016-05-31",
            "transactionkey": 97031
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-31",
            "transactionkey": 97027
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-29",
            "transactionkey": 96934
        },
        {
            "name": "Tailspin Toys (Head Office)",
            "date": "2016-05-28",
            "transactionkey": 96863
        },
        {
            "name": "Wingtip Toys (Head Office)",
            "date": "2016-05-31",
            "transactionkey": 97028
```