CSCI 331 PROJECT 1 – INDIVIDUAL PDF

PROFESSOR: PETER HELLER

SECTION: 9:15-10:30 AM

GROUP: G9-5

PDF OF: KEVIN TANG

THIS PDF CONTAINS THE 3 TOP, 3 WORST, AND WORST CORRECTED QUERIES

CONTENTS

TOP #1: A SIMPLE QUERY

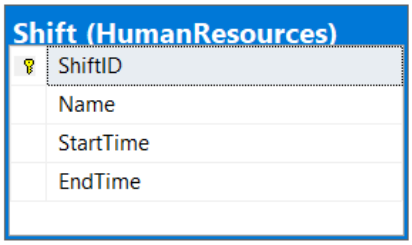
PROBLEM STATEMENT

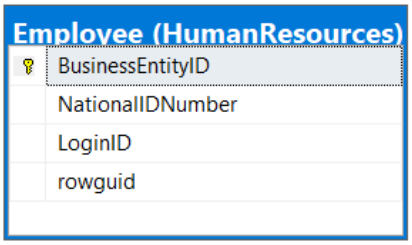
Finds the start and end times for the three shifts that an employee can choose to work between using AdventureWorks2017

REASON IT IS A TOP

The query does not do anything other than look through the shifts of the employees and returns when they begin and end, along with putting the label of the time of day beside the times

KEY AND STANDARD VIEW OF TABLES USED





Shift (HumanResources)

Column Name	Data Type	Allow Nulls
ShiftID	tinyint	☐
Name	Name:nvarchar(50)	☐
StartTime	time(7)	☐
EndTime	time(7)	☐
ModifiedDate	datetime	☐
		☐

Employee (HumanResources)

Column Name	Data Type	Allow Nulls
BusinessEntityID	int	☐
NationalIDNumber	nvarchar(15)	☐
LoginID	nvarchar(256)	☐
OrganizationNode	hierarchyid	☑
OrganizationLevel		☑
JobTitle	nvarchar(50)	☐
BirthDate	date	☐
MaritalStatus	nchar(1)	☐
Gender	nchar(1)	☐
HireDate	date	☐
SalariedFlag	Flag:bit	☐
VacationHours	smallint	☐
SickLeaveHours	smallint	☐
CurrentFlag	Flag:bit	☐
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐
		☐

TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
HumanResources.Shift	S.[Name] S.StartTime S.EndTime

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY

```
USE AdventureWorks2017;
SELECT S.[Name] AS [Time of Day],
       S.StartTime,
       S.EndTime
FROM HumanResources.Shift AS S
    LEFT JOIN HumanResources.Employee AS E
        ON S.ModifiedDate = E.ModifiedDate;
```

RELATIONAL AND JSON OUTPUT (3 ROWS AFFECTED)

	Time of Day	StartTime	EndTime
1	Day	07:00:00.0000000	15:00:00.0000000
2	Evening	15:00:00.0000000	23:00:00.0000000
3	Night	23:00:00.0000000	07:00:00.0000000


```
{
    "Working Shifts": [{
            "Time of Day": "Day",
            "StartTime": "07:00:00",
            "EndTime": "15:00:00"
        }, {
            "Time of Day": "Evening",
            "StartTime": "15:00:00",
            "EndTime": "23:00:00"
        }, {
            "Time of Day": "Night",
            "StartTime": "23:00:00",
            "EndTime": "07:00:00"
        }
    ]
}
```

TOP #2: A MEDIUM QUERY

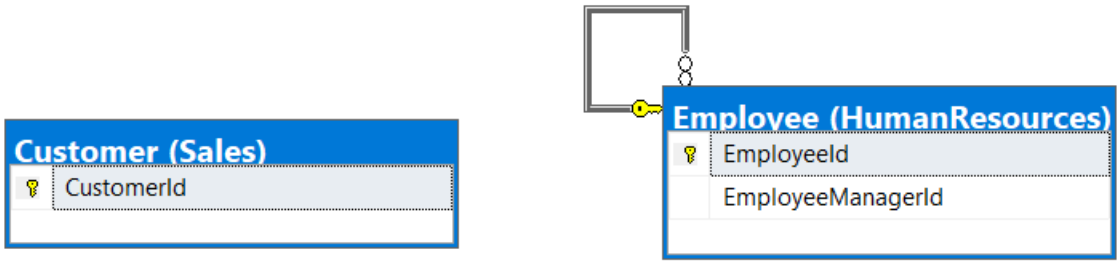
PROBLEM STATEMENT

Returns employee ID and country if there are customers in the same country

REASON IT IS A TOP

Uses a cross join to determine how many customers are from a certain country

KEY AND STANDARD VIEW OF TABLES USED



Customer (Sales)

Column Name	Data Type	Allow Nulls
CustomerId	Udt.SurrogateKeyInt:int	☐
CustomerCompanyName	Udt.CompanyName:nvarch...	☐
CustomerContactName	Udt.ContactName:nvarchar...	☐
CustomerContactTitle	Udt.Title:nvarchar(30)	☐
CustomerAddress	Udt.Address:nvarchar(60)	☐
CustomerCity	Udt.City:nvarchar(15)	☐
CustomerRegion	Udt.Region:nvarchar(15)	☑
CustomerPostalCode	Udt.PostalCode:nvarchar(10)	☑
CustomerCountry	Udt.Country:nvarchar(15)	☐
CustomerPhoneNumber	Udt.TelephoneNumber:nva...	☐
CustomerFaxNumber	Udt.TelephoneNumber:nva...	☑
		☐

Employee (HumanResources)

Column Name	Data Type	Allow Nulls
EmployeeId	Udt.SurrogateKeyInt:int	☐
EmployeeLastName	Udt.LastName:nvarchar(25)	☐
EmployeeFirstName	Udt.FirstName:nvarchar(25)	☐
EmployeeTitle	Udt.Title:nvarchar(30)	☐
EmployeeTitleOfCourtesy	Udt.TitleOfCourtesy:nvarch...	☐
BirthDate	Udt.DateYYYYMMDD:date	☐
HireDate	Udt.DateYYYYMMDD:date	☐
EmployeeAddress	Udt.Address:nvarchar(60)	☐
EmployeeCity	Udt.City:nvarchar(15)	☑
EmployeeRegion	Udt.Region:nvarchar(15)	☑
EmployeePostalCode	Udt.PostalCode:nvarchar(10)	☑
EmployeeCountry	Udt.Country:nvarchar(15)	☐
EmployeePhoneNumber	Udt.TelephoneNumber:nva...	☐
EmployeeManagerId	Udt.SurrogateKeyInt:int	☑
		☐

TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
HumanResources.Employee	EmployeeId EmployeeCountry

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY

```
USE Northwinds2022TSQLV7;
SELECT HRE.EmployeeId,
       HRE.EmployeeCountry
FROM HumanResources.Employee AS HRE
    CROSS JOIN Sales.Customer AS C
WHERE HRE.EmployeeCountry =
(
    SELECT C.CustomerCountry
)
GROUP BY HRE.EmployeeId,
         HRE.EmployeeCountry;
```

RELATIONAL AND JSON OUTPUT (9 ROWS AFFECTED)

	EmployeeId	EmployeeCountry
1	1	USA
2	2	USA
3	3	USA
4	4	USA
5	5	UK
6	6	UK
7	7	UK
8	8	USA
9	9	UK


```
{
    "Same Countries of Employees and Customers": [{
            "EmployeeId": 1,
            "EmployeeCountry": "USA"
        }, {
            "EmployeeId": 2,
            "EmployeeCountry": "USA"
        }, {
            "EmployeeId": 3,
            "EmployeeCountry": "USA"
        }, {
            "EmployeeId": 4,
            "EmployeeCountry": "USA"
        }, {
            "EmployeeId": 5,
            "EmployeeCountry": "UK"
        }, {
            "EmployeeId": 6,
            "EmployeeCountry": "UK"
        }, {
            "EmployeeId": 7,
            "EmployeeCountry": "UK"
        }, {
            "EmployeeId": 8,
            "EmployeeCountry": "USA"
        }, {
            "EmployeeId": 9,
            "EmployeeCountry": "UK"
        }
    ]
}
```

TOP #3: A MEDIUM QUERY

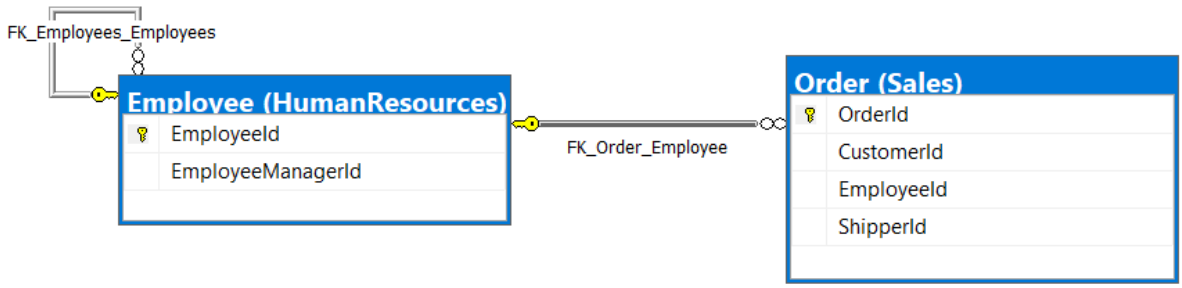
PROBLEM STATEMENT

Returns details of employees, such as their name, ID, title, and order ID's

REASON IT IS A TOP

Makes use of CONCAT and INNER JOIN to combine name and title of employees including details in a table

KEY AND STANDARD VIEW OF TABLES USED



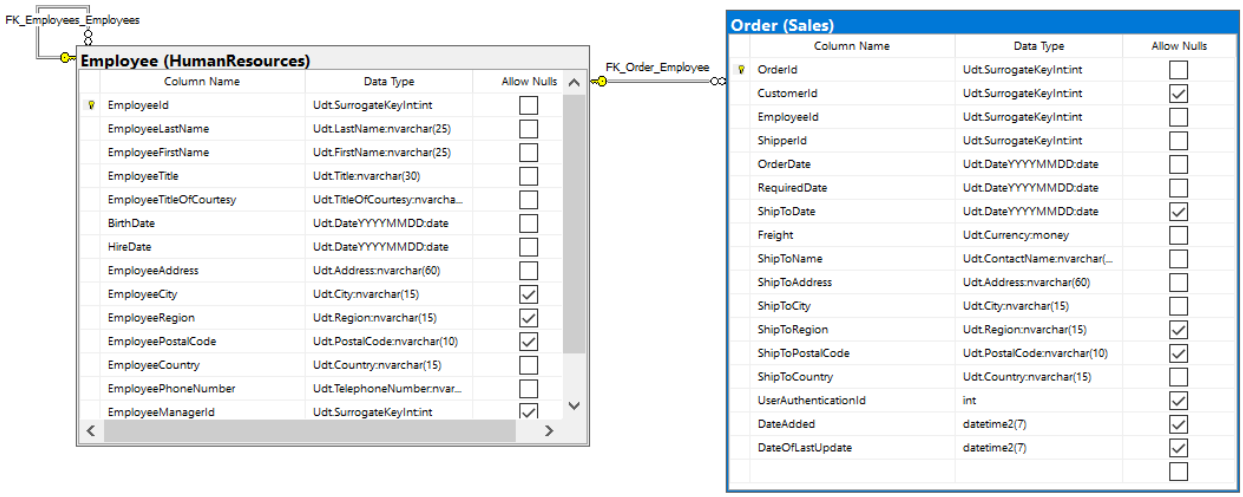


TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
Derived	Name And Title
Sales.Order	EmployeeId OrderId

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY

```
USE Northwinds2022TSQLV7;
SELECT DISTINCT
       CONCAT(E.EmployeeFirstName, ' ', E.EmployeeLastName, ', ', E.EmployeeTitle) AS
[Name And Title],
       O.EmployeeId,
       O.OrderId
FROM HumanResources.Employee AS E
    INNER JOIN Sales.[Order] AS O
        ON E.EmployeeId =
        (
            SELECT O.EmployeeId
        )
GROUP BY E.EmployeeLastName,
         E.EmployeeFirstName,
         O.EmployeeId,
         O.OrderId,
         E.EmployeeTitle;
```

RELATIONAL AND JSON OUTPUT (830 ROWS AFFECTED)

	Name And Title	EmployeeId	OrderId
1	Sven Mortensen, Sales Manager	5	10248
2	Paul Suurs, Sales Representative	6	10249
3	Yael Peled, Sales Representative	4	10250
4	Judy Lew, Sales Manager	3	10251
5	Yael Peled, Sales Representative	4	10252
6	Judy Lew, Sales Manager	3	10253
7	Sven Mortensen, Sales Manager	5	10254
8	Patricia Doyle, Sales Representative	9	10255
9	Judy Lew, Sales Manager	3	10256
10	Yael Peled, Sales Representative	4	10257


```
{
    "Employee Details": [{
            "Name And Title": "Sven Mortensen, Sales Manager",
            "EmployeeId": 5,
            "OrderId": 10248
        }, {
            "Name And Title": "Paul Suurs, Sales Representative",
            "EmployeeId": 6,
            "OrderId": 10249
        }, {
            "Name And Title": "Yael Peled, Sales Representative",
            "EmployeeId": 4,
            "OrderId": 10250
        }, {
            "Name And Title": "Judy Lew, Sales Manager",
            "EmployeeId": 3,
            "OrderId": 10251
        }, {
            "Name And Title": "Yael Peled, Sales Representative",
            "EmployeeId": 4,
            "OrderId": 10252
        }, {
            "Name And Title": "Judy Lew, Sales Manager",
            "EmployeeId": 3,
            "OrderId": 10253
        }, {
            "Name And Title": "Sven Mortensen, Sales Manager",
            "EmployeeId": 5,
            "OrderId": 10254
        }, {
            "Name And Title": "Patricia Doyle, Sales Representative
            "EmployeeId": 9,
            "OrderId": 10255
        }, {
            "Name And Title": "Judy Lew, Sales Manager",
            "EmployeeId": 3,
            "OrderId": 10256
        }, {
            "Name And Title": "Yael Peled, Sales Representative",
            "EmployeeId": 4,
            "OrderId": 10257
        }, {
```

WORST #1: A SIMPLE QUERY

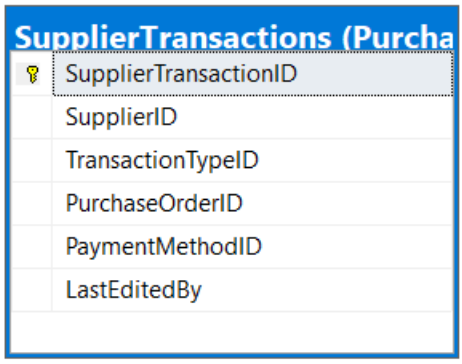
PROBLEM STATEMENT

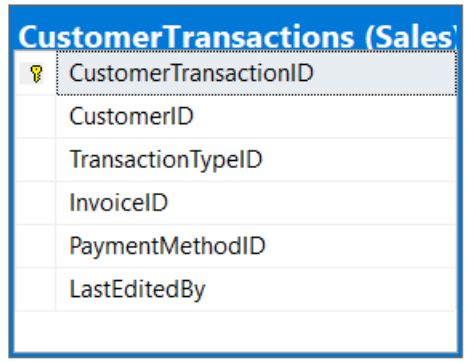
Customer purchase on same day of restocking using WideWorldImporters

REASON IT IS A WORST

Use of LEFT JOIN was unnecessary

KEY AND STANDARD VIEW OF TABLES USED





SupplierTransactions (Purchasing)

Column Name	Data Type	Allow Nulls
SupplierTransactionID	int	☐
SupplierID	int	☐
TransactionTypeID	int	☐
PurchaseOrderID	int	☑
PaymentMethodID	int	☑
SupplierInvoiceNumber	nvarchar(20)	☑
TransactionDate	date	☐
AmountExcludingTax	decimal(18, 2)	☐
TaxAmount	decimal(18, 2)	☐
TransactionAmount	decimal(18, 2)	☐
OutstandingBalance	decimal(18, 2)	☐
FinalizationDate	date	☑
IsFinalized		☑
LastEditedBy	int	☐
LastEditedWhen	datetime2(7)	☐
		☐

CustomerTransactions (Sales)

Column Name	Data Type	Allow Nulls
CustomerTransactionID	int	☐
CustomerID	int	☐
TransactionTypeID	int	☐
InvoiceID	int	☑
PaymentMethodID	int	☑
TransactionDate	date	☐
AmountExcludingTax	decimal(18, 2)	☐
TaxAmount	decimal(18, 2)	☐
TransactionAmount	decimal(18, 2)	☐
OutstandingBalance	decimal(18, 2)	☐
FinalizationDate	date	☑
IsFinalized		☑
LastEditedBy	int	☐
LastEditedWhen	datetime2(7)	☐
		☐

TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
Purchasing.SupplierTransactions	TransactionDate TransactionAmount TaxAmount AmountExcludingTax

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY OF WORST

```
USE WideWorldImporters;
SELECT DISTINCT TOP (1000)
       ST.TransactionDate,
       ST.TransactionAmount,
       ST.TaxAmount,
       ST.AmountExcludingTax
FROM Purchasing.SupplierTransactions AS ST
    LEFT JOIN Sales.CustomerTransactions AS CT
        ON ST.TransactionDate = CT.TransactionDate

ORDER BY ST.TransactionAmount DESC;
```

QUERY OF WORST CORRECTED

```
USE WideWorldImporters;
SELECT DISTINCT TOP (1000)
       ST.TransactionDate,
       ST.TransactionAmount,
       ST.TaxAmount,
       ST.AmountExcludingTax
FROM Purchasing.SupplierTransactions AS ST,
     Sales.CustomerTransactions AS CT
WHERE ST.TransactionDate = CT.TransactionDate
ORDER BY ST.TransactionAmount DESC;
```

HOW IT WAS CORRECTED:

Removed LEFT JOIN

RELATIONAL AND JSON OUTPUT (1000 ROWS AFFECTED)

	TransactionDate	TransactionAmount	TaxAmount	AmountExcludingTax
1	2016-05-31	1871894.10	244160.10	1627734.00
2	2016-05-30	1868333.70	243695.70	1624638.00
3	2016-05-27	1862517.00	242937.00	1619580.00
4	2016-05-25	1861081.80	242749.80	1618332.00
5	2016-05-26	1860916.20	242728.20	1618188.00
6	2016-05-23	1859094.60	242490.60	1616604.00
7	2016-05-24	1858825.50	242455.50	1616370.00
8	2016-05-20	1855071.90	241965.90	1613106.00
9	2016-05-19	1851842.70	241544.70	1610298.00
10	2016-05-17	1848758.40	241142.40	1607616.00
11	2016-05-18	1848192.60	241068.60	1607124.00
12	2016-05-16	1843969.80	240517.80	1603452.00
13	2016-05-13	1838049.60	239745.60	1598304.00
14	2016-05-12	1836545.40	239549.40	1596996.00
15	2016-05-10	1832764.20	239056.20	1593708.00
16	2016-05-11	1832419.20	239011.20	1593408.00
17	2016-05-09	1830480.30	238758.30	1591722.00


```
{
    "Same Day Purchase": [{
            "TransactionDate": "2016-05-31",
            "TransactionAmount": 1871894.10,
            "TaxAmount": 244160.10,
            "AmountExcludingTax": 1627734.00
        }, {
            "TransactionDate": "2016-05-30",
            "TransactionAmount": 1868333.70,
            "TaxAmount": 243695.70,
            "AmountExcludingTax": 1624638.00
        }, {
            "TransactionDate": "2016-05-27",
            "TransactionAmount": 1862517.00,
            "TaxAmount": 242937.00,
            "AmountExcludingTax": 1619580.00
        }, {
            "TransactionDate": "2016-05-25",
            "TransactionAmount": 1861081.80,
            "TaxAmount": 242749.80,
            "AmountExcludingTax": 1618332.00
        }, {
            "TransactionDate": "2016-05-26",
            "TransactionAmount": 1860916.20,
            "TaxAmount": 242728.20,
            "AmountExcludingTax": 1618188.00
        }, {
            "TransactionDate": "2016-05-23",
            "TransactionAmount": 1859094.60,
            "TaxAmount": 242490.60,
            "AmountExcludingTax": 1616604.00
        }, {
            "TransactionDate": "2016-05-24",
            "TransactionAmount": 1858825.50,
            "TaxAmount": 242455.50,
            "AmountExcludingTax": 1616370.00
        }, {
            "TransactionDate": "2016-05-20",
            "TransactionAmount": 1855071.90,
            "TaxAmount": 241965.90,
            "AmountExcludingTax": 1613106.00
```

WORST #2: A MEDIUM QUERY

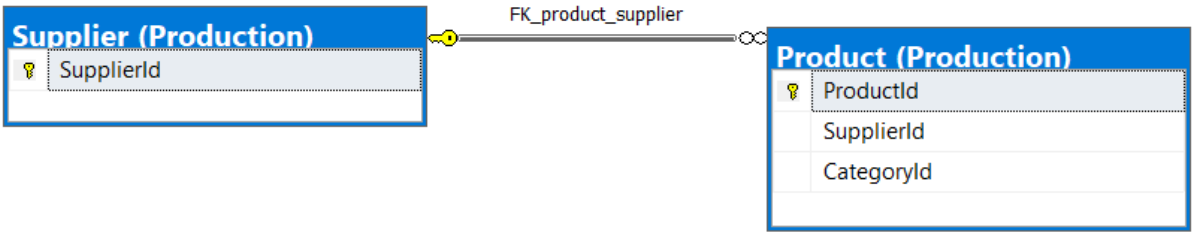
PROBLEM STATEMENT

Shows the details of all products that are priced below $100 using Northwinds2022TSQLV7

REASON IT IS A WORST

Subquery was unnecessary

KEY AND STANDARD VIEW OF TABLES USED



Supplier (Production)

	Column Name	Data Type	Allow Nulls
🔑	SupplierId	Udt.SurrogateKeyInt:int	☐
	SupplierCompanyName	Udt.CompanyName:nvarch...	☐
	SupplierContactName	Udt.ContactName:nvarchar...	☐
	SupplierContactTitle	Udt.ContactTitle:nvarchar(...	☐
	SupplierAddress	Udt.Address:nvarchar(60)	☐
	SupplierCity	Udt.City:nvarchar(15)	☐
	SupplierRegion	Udt.Region:nvarchar(15)	☑
	SupplierPostalCode	Udt.PostalCode:nvarchar(10)	☑
	SupplierCountry	Udt.Country:nvarchar(15)	☐
	SupplierPhoneNumber	Udt.TelephoneNumber:nva...	☐
	SupplierFaxNumber	Udt.TelephoneNumber:nva...	☑
			☐

Product (Production)

	Column Name	Data Type	Allow Nulls
🔑	ProductId	Udt.SurrogateKeyInt:int	☐
	ProductName	Udt.ProductName:nvarchar...	☐
	SupplierId	Udt.SurrogateKeyInt:int	☐
	CategoryId	Udt.SurrogateKeyInt:int	☐
	UnitPrice	Udt.Currency:money	☐
	Discontinued	Udt.FlagBit:bit	☐
			☐

FK_product_supplier

TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
Production.Supplier	ProductName ProductId UnitPrice

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY OF WORST

```
USE Northwinds2022TSQLV7;
SELECT DISTINCT
       ProductName,
       ProductId,
       UnitPrice
FROM Production.Product
    CROSS JOIN Production.Supplier AS S
WHERE UnitPrice IN
      (
          SELECT UnitPrice FROM Production.Product WHERE UnitPrice < 100
      );
```

QUERY OF WORST CORRECTED

```
USE Northwinds2022TSQLV7;
SELECT DISTINCT
       ProductName,
       ProductId,
       UnitPrice
FROM Production.Product
    CROSS JOIN Production.Supplier AS S
WHERE UnitPrice < 100;
```

HOW IT WAS CORRECTED:

Removed subquery

RELATIONAL AND JSON OUTPUT (75 ROWS AFFECTED)

	ProductName	ProductId	UnitPrice
1	Product HHYDP	1	18.00
2	Product RECZE	2	19.00
3	Product IMEHJ	3	10.00
4	Product KSBRM	4	22.00
5	Product EPEIM	5	21.35
6	Product VAIIV	6	25.00
7	Product HMLNI	7	30.00
8	Product WVJFP	8	40.00
9	Product AOZBW	9	97.00


```
{
    "Inexpensive Products": [{
            "ProductName": "Product HHYDP",
            "ProductId": 1,
            "UnitPrice": 18.0000
        }, {
            "ProductName": "Product RECZE",
            "ProductId": 2,
            "UnitPrice": 19.0000
        }, {
            "ProductName": "Product IMEHJ",
            "ProductId": 3,
            "UnitPrice": 10.0000
        }, {
            "ProductName": "Product KSBRM",
            "ProductId": 4,
            "UnitPrice": 22.0000
        }, {
            "ProductName": "Product EPEIM",
            "ProductId": 5,
            "UnitPrice": 21.3500
        }, {
            "ProductName": "Product VAIIV",
            "ProductId": 6,
            "UnitPrice": 25.0000
        }, {
            "ProductName": "Product HMLNI",
            "ProductId": 7,
            "UnitPrice": 30.0000
        }, {
            "ProductName": "Product WVJFP",
            "ProductId": 8,
            "UnitPrice": 40.0000
        }, {
            "ProductName": "Product AOZBW",
            "ProductId": 9,
            "UnitPrice": 97.0000
        }, {
            "ProductName": "Product YHXGE",
            "ProductId": 10,
            "UnitPrice": 31.0000
        }, {
```

WORST #3: A COMPLEX QUERY

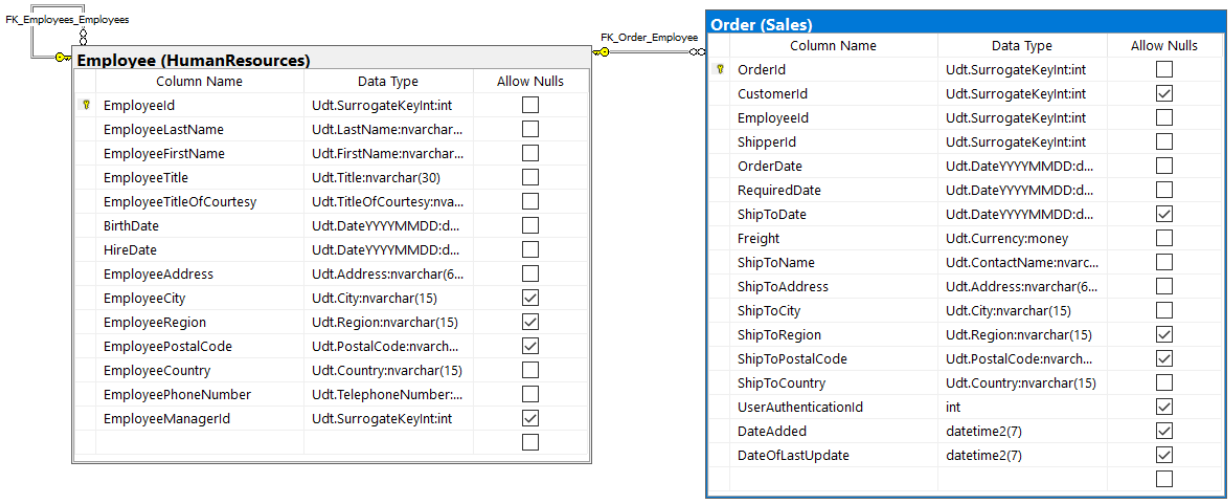
PROBLEM STATEMENT

Shows the top 3 oldest employees at the company using Northwinds2022TSQLV7

REASON IT IS A WORST

Having two CROSS JOINs was not necessary

KEY AND STANDARD VIEW OF TABLES USED



Column Name	Data Type	Allow Nulls
EmployeeId	Udt.SurrogateKeyInt:int	☐
EmployeeLastName	Udt.LastName:nvarchar...	☐
EmployeeFirstName	Udt.FirstName:nvarchar...	☐
EmployeeTitle	Udt.Title:nvarchar(30)	☐
EmployeeTitleOfCourtesy	Udt.TitleOfCourtesy:nva...	☐
BirthDate	Udt.DateYYYYMMDD:d...	☐
HireDate	Udt.DateYYYYMMDD:d...	☐
EmployeeAddress	Udt.Address:nvarchar(6...	☐
EmployeeCity	Udt.City:nvarchar(15)	☑
EmployeeRegion	Udt.Region:nvarchar(15)	☑
EmployeePostalCode	Udt.PostalCode:nvarch...	☑
EmployeeCountry	Udt.Country:nvarchar(15)	☐
EmployeePhoneNumber	Udt.TelephoneNumber:...	☐
EmployeeManagerId	Udt.SurrogateKeyInt:int	☑
		☐

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyInt:int	☐
CustomerId	Udt.SurrogateKeyInt:int	☑
EmployeeId	Udt.SurrogateKeyInt:int	☐
ShipperId	Udt.SurrogateKeyInt:int	☐
OrderDate	Udt.DateYYYYMMDD:d...	☐
RequiredDate	Udt.DateYYYYMMDD:d...	☐
ShipToDate	Udt.DateYYYYMMDD:d...	☑
Freight	Udt.Currency:money	☐
ShipToName	Udt.ContactName:nvarc...	☐
ShipToAddress	Udt.Address:nvarchar(6...	☐
ShipToCity	Udt.City:nvarchar(15)	☐
ShipToRegion	Udt.Region:nvarchar(15)	☑
ShipToPostalCode	Udt.PostalCode:nvarch...	☑
ShipToCountry	Udt.Country:nvarchar(15)	☐
UserAuthenticationId	int	☑
DateAdded	datetime2(7)	☑
DateOfLastUpdate	datetime2(7)	☑
		☐

TABLE SHOWING COLUMNS PROJECTED IN THE END

Table Name	Column Name
Derived	Years At Company
Sales.[Order]	EmployeeId

TABLE SHOWING HOW PROJECTION SORTED (IF APPLICABLE)

Not applicable

QUERY OF WORST

```
USE Northwinds2022TSQLV7;
DROP FUNCTION IF EXISTS dbo.MostRecentHires;
GO
CREATE FUNCTION dbo.MostRecentHires
(
    @hiredate DATE
)
RETURNS NVARCHAR(10)
AS
BEGIN
    DECLARE @diff NVARCHAR(10);
    SELECT @diff = DATEDIFF(YEAR, @hiredate, GETDATE());
    RETURN @diff;
END;
GO
SELECT TOP (3)
       O.EmployeeId AS [Employee ID],
       dbo.MostRecentHires(E.HireDate) AS [Years At Company]
FROM Sales.Customer AS C
    CROSS JOIN HumanResources.Employee AS E
    CROSS JOIN Sales.[Order] AS O
WHERE C.CustomerId =
(
    SELECT O.CustomerId WHERE E.EmployeeId = O.EmployeeId
)
GROUP BY O.EmployeeId,
         dbo.MostRecentHires(E.HireDate)
ORDER BY dbo.MostRecentHires(E.HireDate) DESC;
```

QUERY OF WORST CORRECTED

```
USE Northwinds2022TSQLV7;
DROP FUNCTION IF EXISTS dbo.MostRecentHires;
GO
CREATE FUNCTION dbo.MostRecentHires
(
    @hiredate DATE
)
RETURNS NVARCHAR(10)
AS
BEGIN
    DECLARE @diff NVARCHAR(10);
    SELECT @diff = DATEDIFF(YEAR, @hiredate, GETDATE());
    RETURN @diff;
END;
GO
SELECT TOP (3)
       O.EmployeeId AS [Employee ID],
       dbo.MostRecentHires(E.HireDate) AS [Years At Company]
FROM HumanResources.Employee AS E
    CROSS JOIN Sales.[Order] AS O
WHERE E.EmployeeId = O.EmployeeId
GROUP BY O.EmployeeId,
         dbo.MostRecentHires(E.HireDate)

ORDER BY dbo.MostRecentHires(E.HireDate) DESC;
```

HOW IT WAS CORRECTED:

Removed one CROSS JOIN

RELATIONAL AND JSON OUTPUT (3 ROWS AFFECTED)

	Employee ID	Years At Company
1	1	9
2	2	9
3	3	9


```
{
    "Oldest Workers": [{
            "Employee ID": 1,
            "Years At Company": "9"
        }, {
            "Employee ID": 2,
            "Years At Company": "9"
        }, {
            "Employee ID": 3,
            "Years At Company": "9"
        }
    ]
}
```