Project 1 Propositions

9 queries from each person:
(3) worst (3) best (3) improved

OCT 2021

ISSUED BY
10:45AM Group 4

REPRESENTATIVE
Min Joo Jeong

Table of Contents

Proposition 1 (Best Simple)

Proposition 1: Return employees assigned to Sales Territory 6 (AdventureWorksDW2017)

Model Diagrams:

Figure 1A: Key View Model for Proposition 1

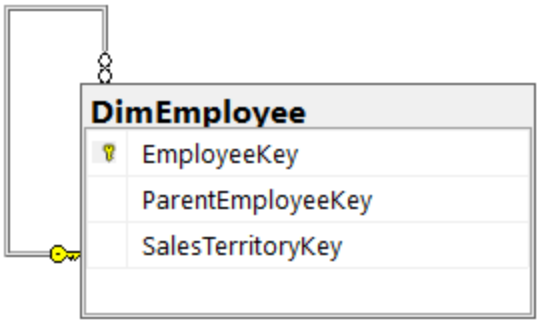
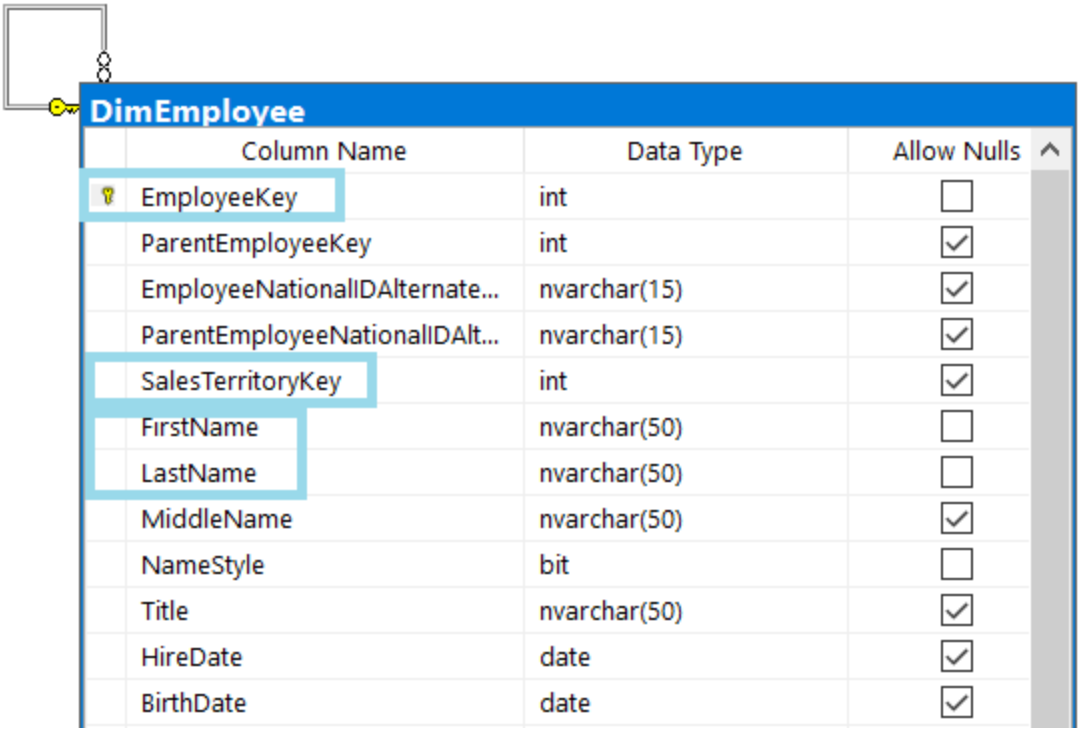


Figure 1B: Standard View Model for Proposition 1



Column Name	Data Type	Allow Nulls
EmployeeKey	int	☐
ParentEmployeeKey	int	☑
EmployeeNationalIDAlternate...	nvarchar(15)	☑
ParentEmployeeNationalIDAlt...	nvarchar(15)	☑
SalesTerritoryKey	int	☑
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
MiddleName	nvarchar(50)	☑
NameStyle	bit	☐
Title	nvarchar(50)	☑
HireDate	date	☑
BirthDate	date	☑

Explanation:

This is the best simple query because of how streamlined the approach is, with selecting only necessary columns from a large table. Declaring a variable for the territory id allows for convenient changes for tentative queries with other territories. Concatenate the first and last names together, and return only the relevant names in the output.

Figure 1C: Tables for SQL query components

Select clause

Table name:	Column name:
dbo.DimEmployee	Salesterritorykey, employeekey, firstname, lastname

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

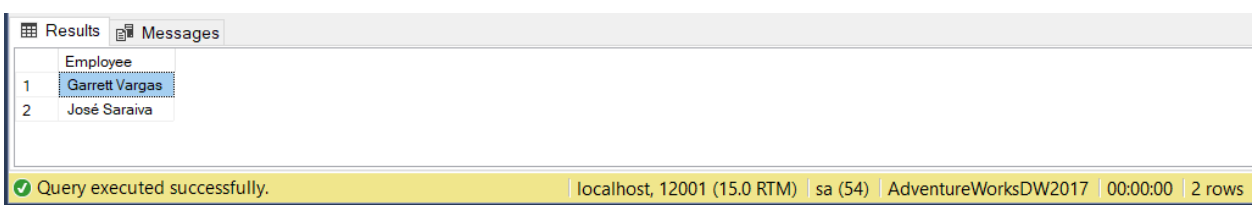
Figure 1D: Formatted SQL Query for Proposition 1

```
USE AdventureWorksDW2017;

DECLARE @territoryid AS INT = 6;

SELECT D.Employee
FROM (
        SELECT salesterritorykey
                ,employeekey
                ,firstname + ' ' + lastname AS Employee
        FROM dbo.DimEmployee
        WHERE SalesTerritoryKey = @territoryid
        ) AS D;
```

Figure 1E: Query Output for Proposition 1



JSON:

Sample JSON Output with total number of rows returned (2)

Figure 1F: Formatted SQL Query with JSON for Proposition 1

```sql
USE AdventureWorksDW2017;

DECLARE @territoryid AS INT = 6;

SELECT D.Employee
FROM (
        SELECT salesterritorykey
                ,employeekey
                ,firstname + ' ' + lastname AS Employee
        FROM dbo.DimEmployee
        WHERE SalesTerritoryKey = @territoryid
        ) AS D;
FOR

JSON PATH
        ,ROOT('EmployeeTerritory6');
```

Figure 1G: Formatted JSON Output for Proposition 1


```
{
   "EmployeeTerritory6":[
      {
         "Employee":"Garrett Vargas"
      },
      {
         "Employee":"José Saraiva"
      }
   ]
}
```

Proposition 2 (Best Medium)

Proposition 2: Return the region for all the managerial staff (AdventureWorksDW2017)

Model Diagrams:

Figure 2A: Key View Model for Proposition 2

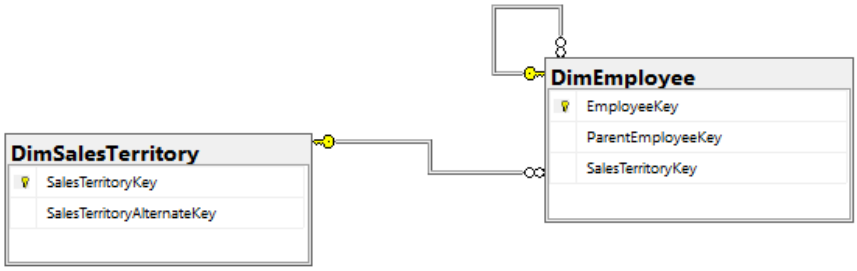
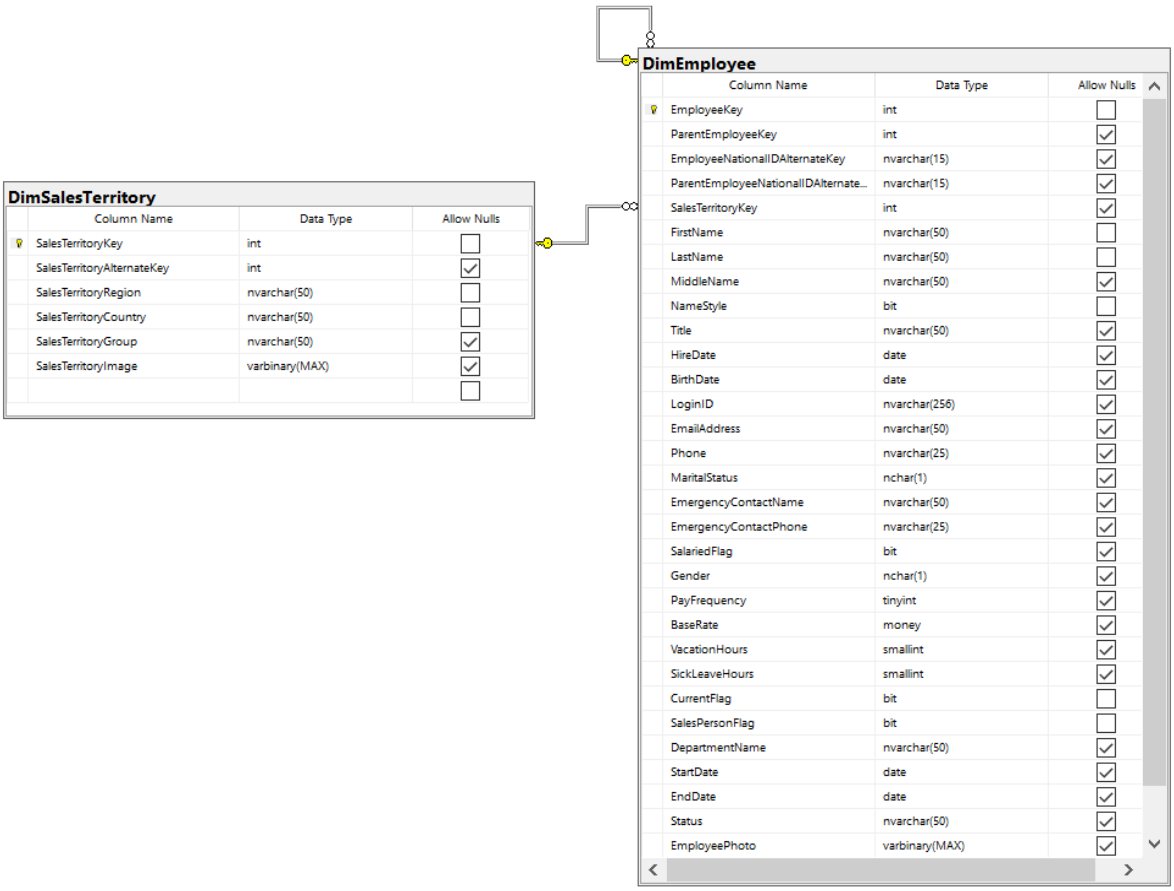


Figure 2B: Standard View Model for Proposition 2



DimSalesTerritory

Column Name	Data Type	Allow Nulls
SalesTerritoryKey	int	☐
SalesTerritoryAlternateKey	int	☑
SalesTerritoryRegion	nvarchar(50)	☐
SalesTerritoryCountry	nvarchar(50)	☐
SalesTerritoryGroup	nvarchar(50)	☑
SalesTerritoryImage	varbinary(MAX)	☑
		☐

DimEmployee

Column Name	Data Type	Allow Nulls
EmployeeKey	int	☐
ParentEmployeeKey	int	☑
EmployeeNationalIDAlternateKey	nvarchar(15)	☑
ParentEmployeeNationalIDAlternate...	nvarchar(15)	☑
SalesTerritoryKey	int	☑
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
MiddleName	nvarchar(50)	☑
NameStyle	bit	☐
Title	nvarchar(50)	☑
HireDate	date	☑
BirthDate	date	☑
LoginID	nvarchar(256)	☑
EmailAddress	nvarchar(50)	☑
Phone	nvarchar(25)	☑
MaritalStatus	nchar(1)	☑
EmergencyContactName	nvarchar(50)	☑
EmergencyContactPhone	nvarchar(25)	☑
SalariedFlag	bit	☑
Gender	nchar(1)	☑
PayFrequency	tinyint	☑
BaseRate	money	☑
VacationHours	smallint	☑
SickLeaveHours	smallint	☑
CurrentFlag	bit	☐
SalesPersonFlag	bit	☐
DepartmentName	nvarchar(50)	☑
StartDate	date	☑
EndDate	date	☑
Status	nvarchar(50)	☑
EmployeePhoto	varbinary(MAX)	☑

Explanation:

Efficient use of CTE to group "Manager" staff using their Titles. Select distinct because DimEmployee table has repeats. Join on DimSalesTerritory to elaborate where the managers are located. Return necessary information like gender or SalariedFlag for potential queries.

Figure 2C: Tables for SQL query components

Select clause

Table name:	Column name:
dbo.DimEmployee	EmployeeKey, FirstName, LastName, Title, SalesTerritoryKey, EmailAddress, Gender, SalariedFlag
dbo.DimSalesTerritory	SalesTerritoryCountry, SalesTerritoryKey

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 2D: Formatted SQL Query for Proposition 2

```
USE AdventureWorksDW2017;

WITH Manager
AS (
        SELECT EmployeeKey
                ,(FirstName + ' ' + LastName) AS Name
                ,Title
                ,SalesTerritoryKey
                ,EmailAddress
                ,Gender
                ,SalariedFlag
        FROM dbo.DimEmployee
        WHERE Title LIKE '%Manager%'
        )
SELECT DISTINCT Manager.Name
        ,Title
        ,B.SalesTerritoryCountry
        ,Gender
        ,SalariedFlag
FROM Manager
INNER JOIN dbo.DimSalesTerritory AS B ON Manager.SalesTerritoryKey = B.SalesTerritoryKey
```

Figure 2E: Query Output for Proposition 2

Results | Messages

	Name	Title	SalesTerritoryCountry	Gender	SalariedFlag
1	Amy Alberts	European Sales Manager	NA	F	1
2	David Bradley	Marketing Manager	NA	M	1
3	David Liu	Accounts Manager	NA	M	1
4	Dylan Miller	Research and Development Manager	NA	M	1
5	Gary Altman	Facilities Manager	NA	M	1
6	Hazem Abol...	Quality Assurance Manager	NA	M	1
7	Jean Trenary	Information Services Manager	NA	F	1
8	Michael Ra...	Research and Development Manager	NA	M	1
9	Paula Barret...	Human Resources Manager	NA	F	1
10	Peter Krebs	Production Control Manager	NA	M	1
11	Roberto Ta...	Engineering Manager	NA	M	1
12	Sheela Word	Purchasing Manager	NA	F	1
13	Stephanie C...	Network Manager	NA	F	1
14	Stephen Jia...	North American Sales Manager	NA	M	1
15	Syed Abbas	Pacific Sales Manager	NA	M	1
16	Wendy Kahn	Finance Manager	NA	F	1
17	Zainal Arifin	Document Control Manager	NA	M	0

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (54) | AdventureWorksDW2017 | 00:00:00 | 17 rows

JSON:

Sample JSON Output with total number of rows returned (17), 4 rows sampled

Figure 2F: Formatted SQL Query with JSON for Proposition 2

```sql
USE AdventureWorksDW2017;

WITH Manager
AS (
        SELECT EmployeeKey
                ,(FirstName + ' ' + LastName) AS Name
                ,Title
                ,SalesTerritoryKey
                ,EmailAddress
                ,Gender
                ,SalariedFlag
        FROM dbo.DimEmployee
        WHERE Title LIKE '%Manager%'
        )
SELECT DISTINCT Manager.Name
        ,Title
        ,B.SalesTerritoryCountry
        ,Gender
        ,SalariedFlag
FROM Manager
INNER JOIN dbo.DimSalesTerritory AS B ON Manager.SalesTerritoryKey = B.SalesTerritoryKey
FOR JSON PATH
        ,ROOT('ManagerRegion');
```

Figure 2G: Formatted JSON Output for Proposition 2


```json
{
   "ManagerRegion":[
      {
         "Name":"Amy Alberts",
         "Title":"European Sales Manager",
         "SalesTerritoryCountry":"NA",
         "Gender":"F",
         "SalariedFlag":true
      },
      {
         "Name":"David Bradley",
         "Title":"Marketing Manager",
         "SalesTerritoryCountry":"NA",
         "Gender":"M",
         "SalariedFlag":true
      },
      {
         "Name":"Hazem Abolrous",
         "Title":"Quality Assurance Manager",
         "SalesTerritoryCountry":"NA",
         "Gender":"M",
         "SalariedFlag":true
      },
      {
         "Name":"Zainal Arifin",
         "Title":"Document Control Manager",
         "SalesTerritoryCountry":"NA",
         "Gender":"M",
         "SalariedFlag":false
      }
   ]
}
```

Proposition 3 (Best Complex)

Proposition 3: Bike shops with a potential list of products stocked based on their store type (AdventureWorksDW2017)

Model Diagrams:

Figure 3A: Key View Model for Proposition 3

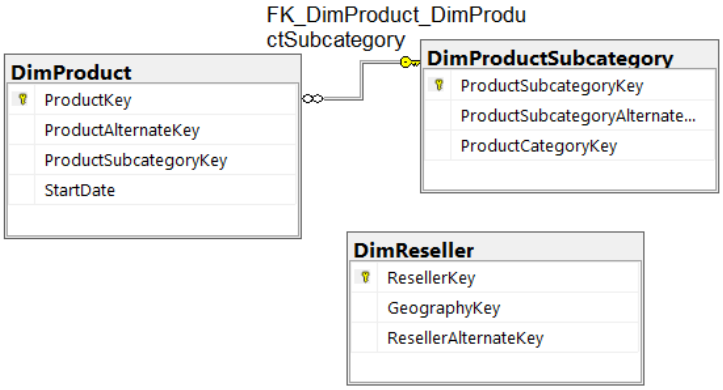
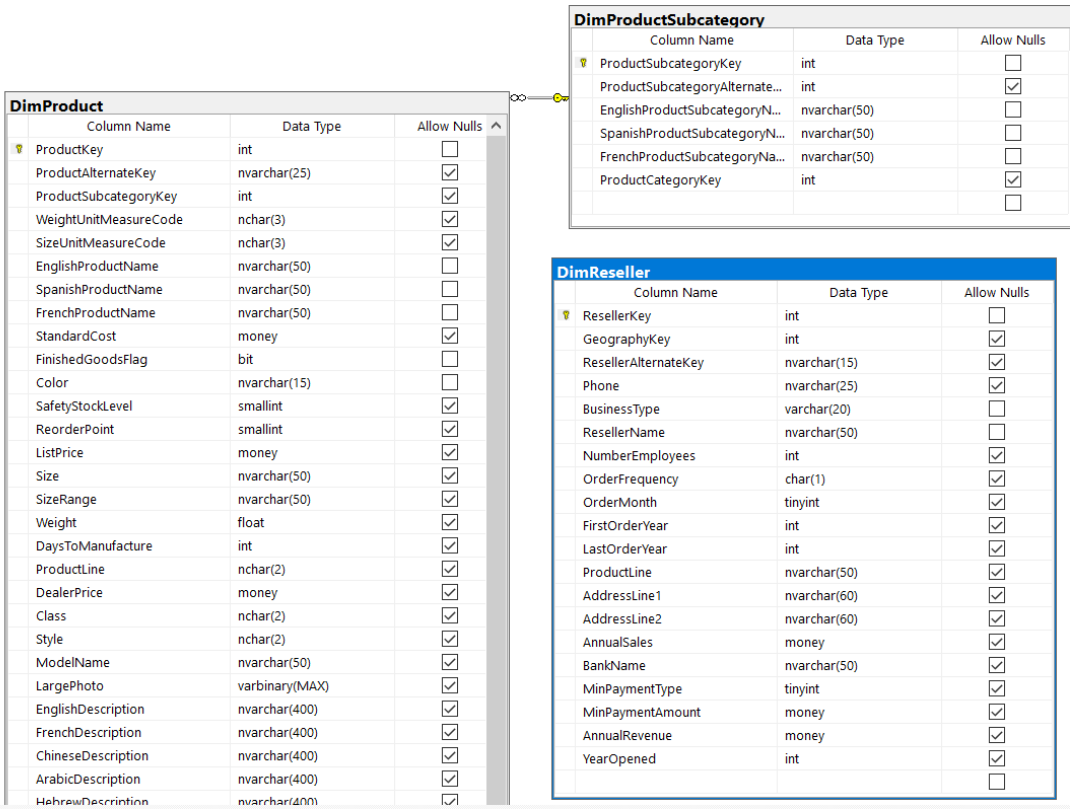


Figure 3B: Standard View Model for Proposition 3



DimProductSubcategory

Column Name	Data Type	Allow Nulls
ProductSubcategoryKey	int	☐
ProductSubcategoryAlternate...	int	☑
EnglishProductSubcategoryN...	nvarchar(50)	☐
SpanishProductSubcategoryN...	nvarchar(50)	☐
FrenchProductSubcategoryNa...	nvarchar(50)	☐
ProductCategoryKey	int	☑
		☐

DimProduct

Column Name	Data Type	Allow Nulls
ProductKey	int	☐
ProductAlternateKey	nvarchar(25)	☑
ProductSubcategoryKey	int	☑
WeightUnitMeasureCode	nchar(3)	☑
SizeUnitMeasureCode	nchar(3)	☑
EnglishProductName	nvarchar(50)	☐
SpanishProductName	nvarchar(50)	☐
FrenchProductName	nvarchar(50)	☐
StandardCost	money	☑
FinishedGoodsFlag	bit	☐
Color	nvarchar(15)	☐
SafetyStockLevel	smallint	☑
ReorderPoint	smallint	☑
ListPrice	money	☑
Size	nvarchar(50)	☑
SizeRange	nvarchar(50)	☑
Weight	float	☑
DaysToManufacture	int	☑
ProductLine	nchar(2)	☑
DealerPrice	money	☑
Class	nchar(2)	☑
Style	nchar(2)	☑
ModelName	nvarchar(50)	☑
LargePhoto	varbinary(MAX)	☑
EnglishDescription	nvarchar(400)	☑
FrenchDescription	nvarchar(400)	☑
ChineseDescription	nvarchar(400)	☑
ArabicDescription	nvarchar(400)	☑
HebrewDescription	nvarchar(400)	☑

DimReseller

Column Name	Data Type	Allow Nulls
ResellerKey	int	☐
GeographyKey	int	☑
ResellerAlternateKey	nvarchar(15)	☑
Phone	nvarchar(25)	☑
BusinessType	varchar(20)	☐
ResellerName	nvarchar(50)	☐
NumberEmployees	int	☑
OrderFrequency	char(1)	☑
OrderMonth	tinyint	☑
FirstOrderYear	int	☑
LastOrderYear	int	☑
ProductLine	nvarchar(50)	☑
AddressLine1	nvarchar(60)	☑
AddressLine2	nvarchar(60)	☑
AnnualSales	money	☑
BankName	nvarchar(50)	☑
MinPaymentType	tinyint	☑
MinPaymentAmount	money	☑
AnnualRevenue	money	☑
YearOpened	int	☑
		☐

Explanation:
A view of Bike Shops was created from DimReseller, and each has a "ProductLine" category. However, the ProductLine input for DimReseller and DimProduct aren't standardized, so the full description of "Road", "Mountain", "Touring" has to be truncated to "R", "M", "T" in order to join the tables. An additional join was created between DimProduct and DimProductSubcategory for descriptive purposes. Use of cross apply makes it apparent that each product is being attributed to each bike store, for a potential list of stock that may be present for each store based on their ProductLine classification.

Figure 3C: Tables for SQL query components

Select clause

Table name:	Column name:
dbo.DimReseller	ResellerKey, ResellerName, ProductLine, NumberEmployees, AnnualRevenue
dbo.DimProduct	ProductLine, EnglishProductName , ProductSubcategoryKey
dbo.DimProductSubcategory	EnglishProductSubcategoryName, ProductSubcategoryKey

Order by (optional, only if exist)

Table name	Column name	Sort order
BikeShops (dbo.DimReseller)	NumberEmployees	Desc

Query:
All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 3D: Formatted SQL Query for Proposition 3

```sql
USE AdventureWorksDW2017;
        --DROP VIEW IF EXISTS BikeShops;
GO

CREATE VIEW BikeShops
AS
SELECT ResellerKey
        ,ResellerName
        ,ProductLine
        ,NumberEmployees
        ,AnnualRevenue
FROM dbo.DimReseller
WHERE BusinessType LIKE '%Bike Shop%';
GO

WITH BigStores
AS (
        SELECT TOP 1
        WITH TIES ResellerName
                ,ProductLine AS StoreType
        FROM BikeShops AS BS
        ORDER BY NumberEmployees DESC
        )
SELECT C1.*
        ,A.*
FROM BigStores AS C1
CROSS APPLY (
        SELECT ProductLine
                ,EnglishProductName
                ,S.EnglishProductSubcategoryName
        FROM dbo.DimProduct AS P
        INNER JOIN dbo.DimProductSubcategory AS S ON P.ProductSubcategoryKey = S.ProductSubcategoryKey
        WHERE P.ProductLine = LEFT(C1.StoreType, 1)
        ) A
```

Figure 3E: Query Output for Proposition 3

Results | Messages

	ResellerName	StoreType	ProductLine	EnglishProductName	EnglishProductSubcategoryName
1	Distant Inn	Road	R	HL Road Frame - Black, 58	Road Frames
2	Highway Bike Shop	Road	R	HL Road Frame - Black, 58	Road Frames
3	Distant Inn	Road	R	HL Road Frame - Red, 58	Road Frames
4	Highway Bike Shop	Road	R	HL Road Frame - Red, 58	Road Frames
5	Fleet Bikes	Mountain	M	Mountain Bike Socks, M	Socks
6	Elemental Sporting Goods	Mountain	M	Mountain Bike Socks, M	Socks
7	One Bike Company	Mountain	M	Mountain Bike Socks, M	Socks
8	Responsible Bike Dealers	Mountain	M	Mountain Bike Socks, M	Socks
9	Fleet Bikes	Mountain	M	Mountain Bike Socks, L	Socks
10	Elemental Sporting Goods	Mountain	M	Mountain Bike Socks, L	Socks
11	One Bike Company	Mountain	M	Mountain Bike Socks, L	Socks
12	Responsible Bike Dealers	Mountain	M	Mountain Bike Socks, L	Socks
13	Distant Inn	Road	R	HL Road Frame - Red, 62	Road Frames
14	Highway Bike Shop	Road	R	HL Road Frame - Red, 62	Road Frames
15	Distant Inn	Road	R	HL Road Frame - Red, 62	Road Frames
16	Highway Bike Shop	Road	R	HL Road Frame - Red, 62	Road Frames
17	Distant Inn	Road	R	HL Road Frame - Red, 62	Road Frames
18	Highway Bike Shop	Road	R	HL Road Frame - Red, 62	Road Frames
19	Distant Inn	Road	R	HL Road Frame - Red, 44	Road Frames

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (80) | AdventureWorksDW2017 | 00:00:00 | 824 rows

JSON:

Sample of 5 rows in JSON Output with total number of rows returned (824)

Figure 3F: Formatted SQL Query with JSON for Proposition 3

```sql
USE AdventureWorksDW2017;
        --DROP VIEW IF EXISTS BikeShops;
GO

CREATE VIEW BikeShops
AS
SELECT ResellerKey
        ,ResellerName
        ,ProductLine
        ,NumberEmployees
        ,AnnualRevenue
FROM dbo.DimReseller
WHERE BusinessType LIKE '%Bike Shop%';
GO

WITH BigStores
AS (
        SELECT TOP 1
        WITH TIES ResellerName
                ,ProductLine AS StoreType
        FROM BikeShops AS BS
        ORDER BY NumberEmployees DESC
        )
SELECT C1.*
        ,A.*
FROM BigStores AS C1
CROSS APPLY (
        SELECT ProductLine
                ,EnglishProductName
                ,S.EnglishProductSubcategoryName
        FROM dbo.DimProduct AS P
        INNER JOIN dbo.DimProductSubcategory AS S ON P.ProductSubcategoryKey = S.ProductSubcategoryKey
        WHERE P.ProductLine = LEFT(C1.StoreType, 1)
        ) A
FOR JSON PATH
        ,ROOT('BikeGoods');
```

Figure 3G: Formatted JSON Output for Proposition 3


```
{
   "BikeGoods":[
      {
         "ResellerName":"Distant Inn",
         "StoreType":"Road",
         "ProductLine":"R ",
         "EnglishProductName":"HL Road Frame - Black, 58",
         "EnglishProductSubcategoryName":"Road Frames"
      },
      {
         "ResellerName":"Distant Inn",
         "StoreType":"Road",
         "ProductLine":"R ",
         "EnglishProductName":"HL Road Frame - Red, 58",
         "EnglishProductSubcategoryName":"Road Frames"
      },
      {
         "ResellerName":"Distant Inn",
         "StoreType":"Road",
         "ProductLine":"R ",
         "EnglishProductName":"HL Road Frame - Red, 62",
         "EnglishProductSubcategoryName":"Road Frames"
      },
      {
         "ResellerName":"Distant Inn",
         "StoreType":"Road",
         "ProductLine":"R ",
         "EnglishProductName":"HL Road Frame - Red, 62",
         "EnglishProductSubcategoryName":"Road Frames"
      },
      {
         "ResellerName":"Distant Inn",
         "StoreType":"Road",
         "ProductLine":"R ",
         "EnglishProductName":"HL Road Frame - Red, 62",
         "EnglishProductSubcategoryName":"Road Frames"
      }
   ]
}
```

Proposition 4 (Worst Simple)

Proposition 4: Amount of suppliers from each country (Northwinds2020TSQLV6)

Model Diagrams:

Figure 4A: Key View Model for Proposition 4

Figure 4B: Standard View Model for Proposition 4

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...	☑
		☐

Explanation:

The worst simple query because of how convoluted table use and select statements became with miscellaneous CTE use and output column renaming, despite how simple it could have been with only two columns from the table (Production.Supplier). D(SupplierCountry, supplierid)'s "D" table expression was never used in the query despite creation. No order to the count of suppliers.

Figure 4C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Supplier	Suppliercountry, supplierid

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 4D: Formatted SQL Query for Proposition 4

```
USE Northwinds2020TSQLV6;

SELECT SupplierCountry
        ,COUNT(DISTINCT supplierid) AS NumberSupplier
FROM (
        SELECT suppliercountry
                ,supplierid
        FROM Production.Supplier
        ) AS D(SupplierCountry, supplierid)
GROUP BY SupplierCountry
```

Figure 4E: Query Output for Proposition 4

Results | Messages

	SupplierCountry	NumberSupplier
1	Australia	2
2	Brazil	1
3	Canada	2
4	Denmark	1
5	Finland	1
6	France	3
7	Germany	3
8	Italy	2
9	Japan	2
10	Netherlands	1
11	Norway	1
12	Singapore	1
13	Spain	1
14	Sweden	2

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (77) | Northwinds2020TSQLV6 | 00:00:00 | 16 rows

JSON:

Sample JSON Output, 7 rows displayed here. Total number of rows returned (16)

Figure 4F: Formatted SQL Query with JSON for Proposition 4

```
USE Northwinds2020TSQLV6;

SELECT SupplierCountry
        ,COUNT(DISTINCT supplierid) AS NumberSupplier
FROM (
        SELECT suppliercountry
                ,supplierid
        FROM Production.Supplier
        ) AS D(SupplierCountry, supplierid)
GROUP BY SupplierCountry
FOR JSON path
        ,ROOT('SupplierCountry');
```

Figure 4G: Formatted JSON Output for Proposition 4


```
{
   "SupplierCountry":[
      {
         "SupplierCountry":"Australia",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"Brazil",
         "NumberSupplier":1
      },
      {
         "SupplierCountry":"Canada",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"Denmark",
         "NumberSupplier":1
      },
      {
         "SupplierCountry":"France",
         "NumberSupplier":3
      },
         "SupplierCountry":"UK",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"USA",
         "NumberSupplier":4
      }
   ]
}
```

Proposition 5 (Worst Medium)

Proposition 5: Pair each UK Employee with a UK Customer Country (Northwinds2020TSQLV6)

Model Diagrams:

Figure 5A: Key View Model for Proposition 5

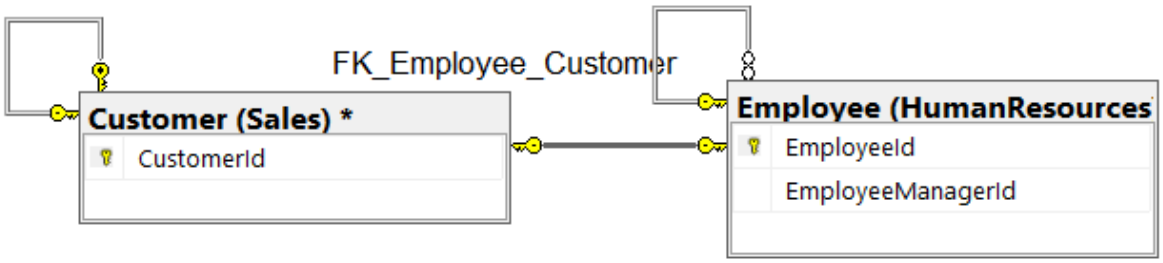
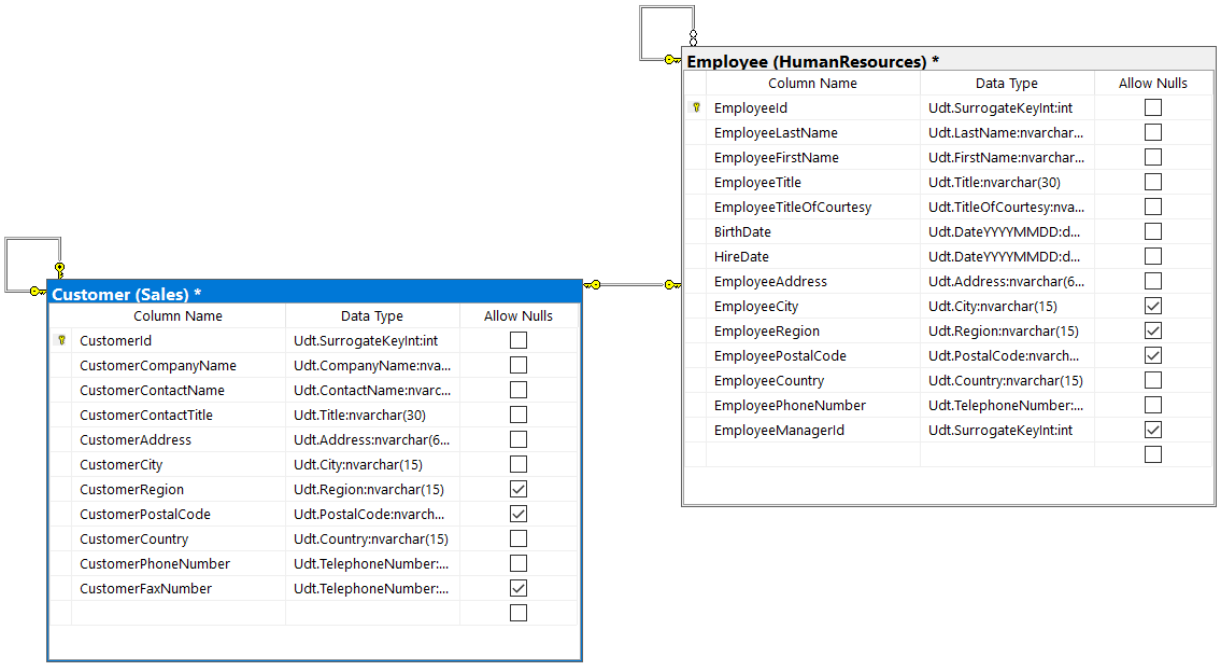


Figure 5B: Standard View Model for Proposition 5



Explanation:
Poorly executed query using a CTE to classify UK customers, and assign every UK Employee to them. Confusing to use a CTE to group UK customers, but not use any for the employees. Paired done with select statement rather than any cross joins, unclear what is being accomplished in code due to lack of descriptors.

Figure 5C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Customer	CustomerCompanyName, CustomerCountry
HumanResources.Employee	EmployeeCountry, EmployeeFirstName, EmployeeLastName, EmployeeID

Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.Customer	CustomerCompanyName	asc

Query:
All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 5D: Formatted SQL Query for Proposition 5

```sql
USE Northwinds2020TSQLV6;

WITH UKCusts
AS (
        SELECT DISTINCT customercompanyname
        FROM Sales.Customer
        WHERE customercountry = N'UK'
        )
SELECT e.EmployeeID
        ,e.EmployeeFirstName + ' ' + e.EmployeeLastName AS NAMES
        ,C.customercompanyname
FROM HumanResources.Employee AS e
        ,UKCusts AS C
WHERE e.EmployeeCountry = N'UK'
ORDER BY C.CustomerCompanyName
```

Figure 5E: Query Output for Proposition 5

Results | Messages

	EmployeeID	names	customercompanyname
1	5	Sven Mortensen	Customer AHPOP
2	6	Paul Suurs	Customer AHPOP
3	7	Russell King	Customer AHPOP
4	9	Patricia Doyle	Customer AHPOP
5	5	Sven Mortensen	Customer GCJSG
6	6	Paul Suurs	Customer GCJSG
7	7	Russell King	Customer GCJSG
8	9	Patricia Doyle	Customer GCJSG
9	5	Sven Mortensen	Customer GYBBY
10	6	Paul Suurs	Customer GYBBY
11	7	Russell King	Customer GYBBY
12	9	Patricia Doyle	Customer GYBBY
13	5	Sven Mortensen	Customer HFBZG
14	6	Paul Suurs	Customer HFBZG
15	7	Russell King	Customer HFBZG
16	9	Patricia Doyle	Customer HFBZG
17	5	Sven Mortensen	Customer LJUCA

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (54) | Northwinds2020TSQLV6 | 00:00:00 | 28 rows

JSON:

Sample 8 rows for JSON Output with total number of rows returned (28)

Figure 5F: Formatted SQL Query with JSON for Proposition 5

```sql
USE Northwinds2020TSQLV6;

WITH UKCusts
AS (
        SELECT DISTINCT customercompanyname
        FROM Sales.Customer
        WHERE customercountry = N'UK'
        )
SELECT e.EmployeeID
        ,e.EmployeeFirstName + ' ' + e.EmployeeLastName AS NAMES
        ,C.customercompanyname
FROM HumanResources.Employee AS e
        ,UKCusts AS C
WHERE e.EmployeeCountry = N'UK'
ORDER BY C.CustomerCompanyName
FOR JSON PATH
        ,ROOT('EmployeeCompany');
```

Figure 5G: Formatted JSON Output for Proposition 5


```json
{
   "EmployeeCompany":[
      {
         "EmployeeID":5,
         "names":"Sven Mortensen",
         "customercompanyname":"Customer AHPOP"
      },
      {
         "EmployeeID":6,
         "names":"Paul Suurs",
         "customercompanyname":"Customer AHPOP"
      },
      {
         "EmployeeID":7,
         "names":"Russell King",
         "customercompanyname":"Customer AHPOP"
      },
      {
         "EmployeeID":9,
         "names":"Patricia Doyle",
         "customercompanyname":"Customer AHPOP"
      },
      {
         "EmployeeID":5,
         "names":"Sven Mortensen",
         "customercompanyname":"Customer GCJSG"
      },
      {
         "EmployeeID":6,
         "names":"Paul Suurs",
         "customercompanyname":"Customer GCJSG"
      },
      {
         "EmployeeID":7,
         "names":"Russell King",
         "customercompanyname":"Customer GCJSG"
      },
      {
         "EmployeeID":9,
         "names":"Patricia Doyle",
         "customercompanyname":"Customer GCJSG"
      }
   ]
}
```

Proposition 6 (Worst Complex)

Proposition 6: Average prices for each supplier with corresponding products (Northwinds2020TSQLV6)

Model Diagrams:

Figure 6A: Key View Model for Proposition 6

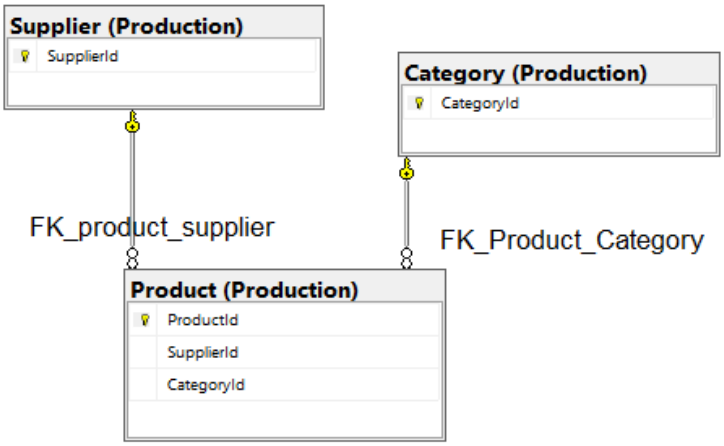
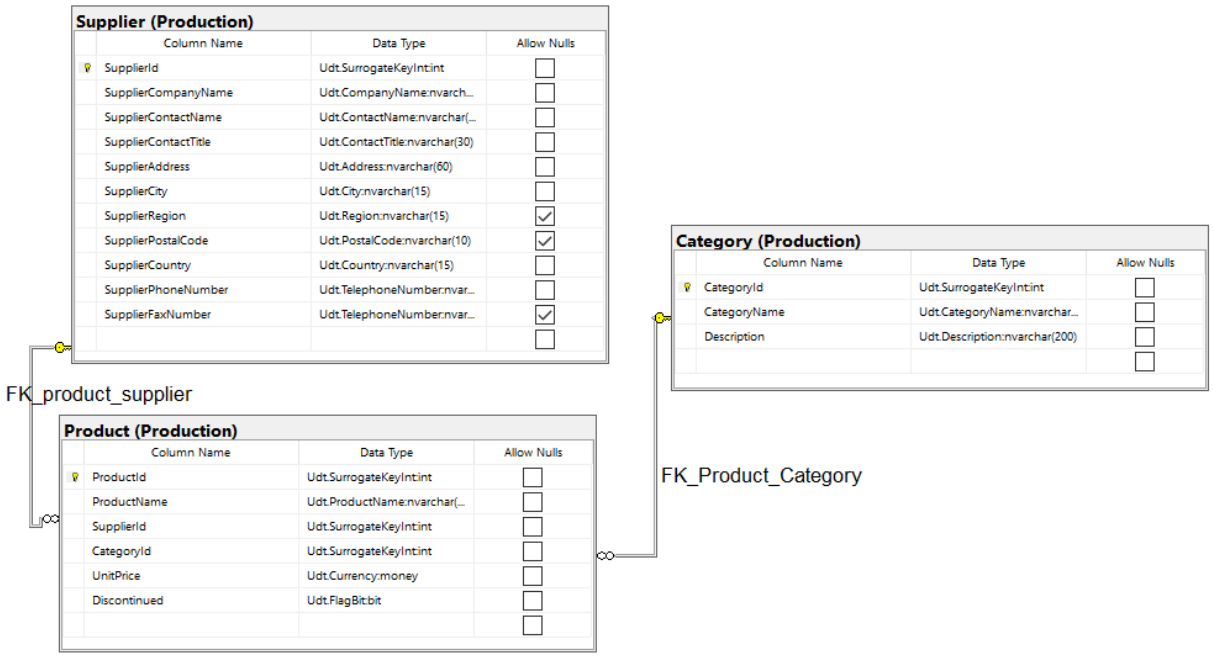


Figure 6B: Standard View Model for Proposition 6



Explanation:
Average prices for each supplier, using corresponding products (described using Category table). Uses Production.Supplier, Production.Product (join on supplierid). This is a poorly executed query because there is an Order by clause which sorts and selects the top 50 most expensive products, but the output isn't organized by such. Additionally, the execution is convoluted with a CTE (poorly named CT1) that summarizes those top 50 products, from a view that is created prior. The final select statement results in the output averaging the prices for each supplier.

Figure 6C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Supplier	supplierid, SupplierCompanyName,
Production.Category	CategoryName
Production.Product	ProductName, UnitPrice, supplierid

Order by (optional, only if exist)

Table name	Column name	Sort order
Production.Product	UnitPrice	DESC

Query:
All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 6D: Formatted SQL Query for Proposition 6

```sql
USE Northwinds2020TSQLV6;
        --DROP VIEW IF EXISTS SupplyProd;
GO

CREATE VIEW SupplyProd
AS
SELECT S.supplierid
        ,S.SupplierCompanyName
        ,C.CategoryName
        ,PD.ProductName
        ,PD.UnitPrice
FROM Production.Supplier AS S
LEFT OUTER JOIN (
        Production.Product AS PD INNER JOIN Production.Category AS C ON PD.categoryid = C.categoryid
        ) ON S.supplierid = PD.supplierid;
GO

WITH CT1
AS (
        SELECT TOP (50)
        WITH TIES *
        FROM SupplyProd
        ORDER BY UnitPrice DESC
        )
SELECT SupplierCompanyName
        ,AVG(UnitPrice) AS avgprice
FROM CT1
GROUP BY SupplierCompanyName
```

Figure 6E: Query Output for Proposition 6

Results | Messages

	SupplierCompanyName	avgprice
1	Supplier BWGYE	81.00
2	Supplier CIYNM	32.725
3	Supplier ELCRN	18.0833
4	Supplier EQPNC	29.50
5	Supplier ERVYZ	24.00
6	Supplier GQRCV	40.7125
7	Supplier JDNUG	18.40
8	Supplier JNNES	42.90
9	Supplier KEREV	33.40
10	Supplier LVJUA	140.75
11	Supplier NZLIF	28.75
12	Supplier OAVQT	44.50
13	Supplier OGLRK	38.90
14	Supplier QOVFD	64.00
15	Supplier QQYEU	21.00
16	Supplier QWUSF	23.25
17	Supplier QZGUF	22.50

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (89) | Northwinds2020TSQLV6 | 00:00:00 | 25 rows

JSON:

7 row Sample JSON Output with total number of rows returned (25)

Figure 6F: Formatted SQL Query with JSON for Proposition 6

```
USE Northwinds2020TSQLV6;
        --DROP VIEW IF EXISTS SupplyProd;
GO

CREATE VIEW SupplyProd
AS
SELECT S.supplierid
        ,S.SupplierCompanyName
        ,C.CategoryName
        ,PD.ProductName
        ,PD.UnitPrice
FROM Production.Supplier AS S
LEFT OUTER JOIN (
        Production.Product AS PD INNER JOIN Production.Category AS C ON PD.categoryid = C.categoryid
        ) ON S.supplierid = PD.supplierid;
GO

WITH CT1
AS (
        SELECT TOP (50)
        WITH TIES *
        FROM SupplyProd
        ORDER BY UnitPrice DESC
        )
SELECT SupplierCompanyName
        ,AVG(UnitPrice) AS avgprice
FROM CT1
GROUP BY SupplierCompanyName
FOR JSON PATH
        ,ROOT('AvgPrices');
```

Figure 6G: Formatted JSON Output for Proposition 6


```
{
   "AvgPrices":[
      {
         "SupplierCompanyName":"Supplier BWGYE",
         "avgprice":81.0000
      },
      {
         "SupplierCompanyName":"Supplier CIYNM",
         "avgprice":32.7250
      },
      {
         "SupplierCompanyName":"Supplier KEREV",
         "avgprice":33.4000
      },
      {
         "SupplierCompanyName":"Supplier QOVFD",
         "avgprice":64.0000
      },
      {
         "SupplierCompanyName":"Supplier VHQZD",
         "avgprice":20.3500
      },
      {
         "SupplierCompanyName":"Supplier ZPYVS",
         "avgprice":37.5650
      },
      {
         "SupplierCompanyName":"Supplier ZWZDM",
         "avgprice":28.7500
      }
   ]
}
```

Proposition 7 (Improved Simple)

Proposition 7: Count of unique customers for days in 2016 (WideWorldImporters)

Model Diagrams:

Figure 7A: Key View Model for Proposition 7

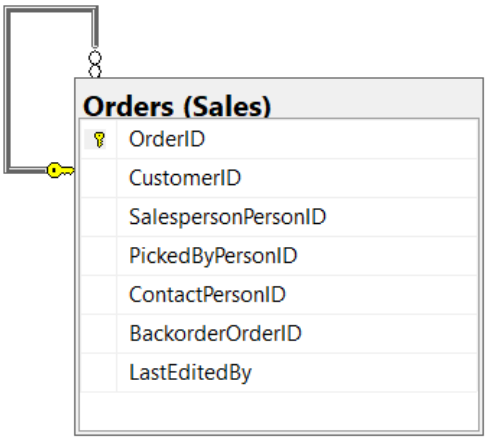
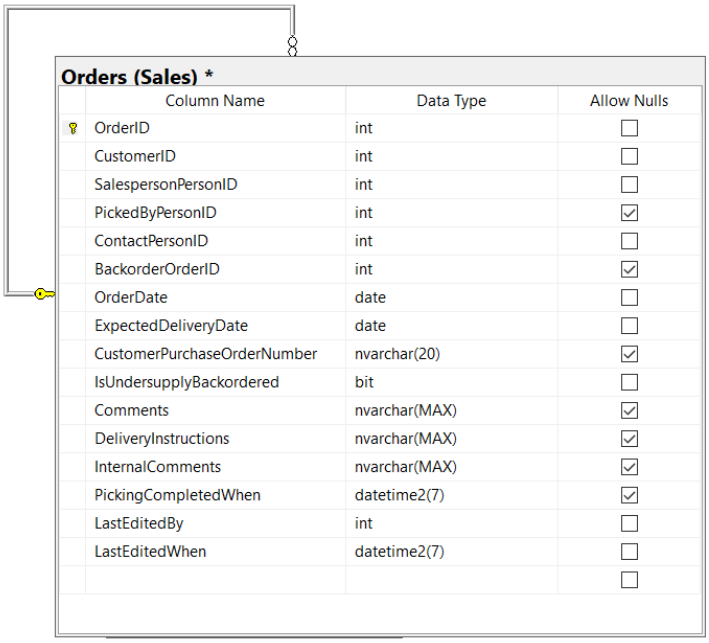


Figure 7B: Standard View Model for Proposition 7



Orders (Sales) *

	Column Name	Data Type	Allow Nulls
🔑	OrderID	int	☐
	CustomerID	int	☐
	SalespersonPersonID	int	☐
	PickedByPersonID	int	☑
	ContactPersonID	int	☐
	BackorderOrderID	int	☑
	OrderDate	date	☐
	ExpectedDeliveryDate	date	☐
	CustomerPurchaseOrderNumber	nvarchar(20)	☑
	IsUndersupplyBackordered	bit	☐
	Comments	nvarchar(MAX)	☑
	DeliveryInstructions	nvarchar(MAX)	☑
	InternalComments	nvarchar(MAX)	☑
	PickingCompletedWhen	datetime2(7)	☑
	LastEditedBy	int	☐
	LastEditedWhen	datetime2(7)	☐
			☐

Explanation:

Use a CTE to simplify the processing of relevant data to find out the count of unique customers per day in 2016. It is an improved query because only relevant columns and information from Sales.Orders are kept with the CTE. COUNT of distinct customers is done in the query. Output is ordered by date.

Figure 7C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Orders	orderdate, customerid

Order by (optional, only if exist)

Table name	Column name	Sort order
C (Sales.Orders CTE)	days2016	asc

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 7D: Formatted SQL Query for Proposition 7

```
USE WideWorldImporters;

WITH C (
        days2016
        ,customerid
        )
AS (
        SELECT orderdate
                ,customerid
        FROM Sales.Orders
        WHERE YEAR(orderdate) = 2016
        )
SELECT days2016
        ,COUNT(DISTINCT customerid) AS numcusts
FROM C
GROUP BY days2016
ORDER BY days2016 ASC;
```

Figure 7E: Query Output for Proposition 7

Results | Messages

	days2016	numcusts
1	2016-01-01	41
2	2016-01-02	43
3	2016-01-04	84
4	2016-01-05	80
5	2016-01-06	99
6	2016-01-07	100
7	2016-01-08	76
8	2016-01-09	54
9	2016-01-11	39
10	2016-01-12	80
11	2016-01-13	62
12	2016-01-14	46
13	2016-01-15	85
14	2016-01-16	35
15	2016-01-18	82
16	2016-01-19	52

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (82) | WideWorldImporters | 00:00:00 | 130 rows

JSON:

Sample JSON Output with total number of rows returned (130), 6 displayed

Figure 7F: Formatted SQL Query with JSON for Proposition 7

```
USE WideWorldImporters;

WITH C (
        days2016
        ,customerid
        )
AS (
        SELECT orderdate
                ,customerid
        FROM Sales.Orders
        WHERE YEAR(orderdate) = 2016
        )
SELECT days2016
        ,COUNT(DISTINCT customerid) AS numcusts
FROM C
GROUP BY days2016
ORDER BY days2016 ASC
FOR JSON PATH
        ,ROOT('Customers2016');
```

Figure 7G: Formatted JSON Output for Proposition 7


```
{
   "Customers2016":[
      {
         "days2016":"2016-01-01",
         "numcusts":41
      },
      {
         "days2016":"2016-01-02",
         "numcusts":43
      },
      {
         "days2016":"2016-01-04",
         "numcusts":84
      },
      {
         "days2016":"2016-01-11",
         "numcusts":39
      },
      {
         "days2016":"2016-01-12",
         "numcusts":80
      },
      {
         "days2016":"2016-01-13",
         "numcusts":62
      }   ]
}
```

Proposition 8 (Improved Medium)

Proposition 8: Descriptive result of 10 most recent internet orders from customers. (AdventureWorksDW2017)

Model Diagrams:

Figure 8A: Key View Model for Proposition 8

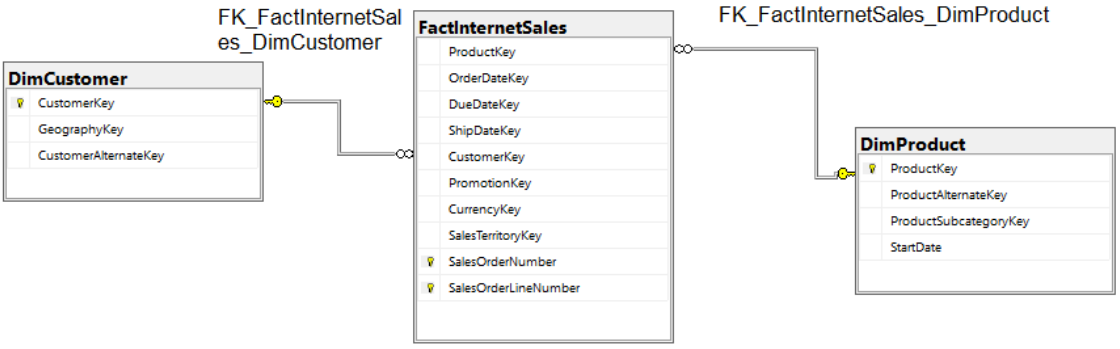
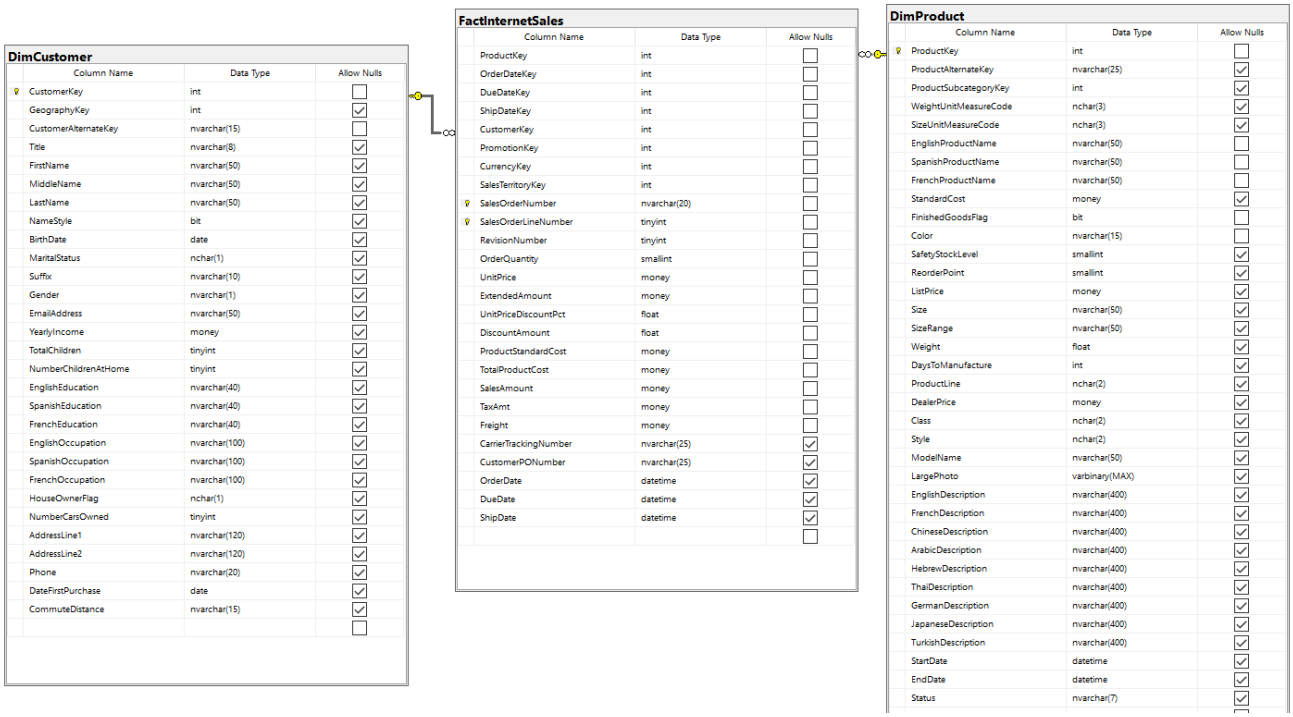


Figure 8B: Standard View Model for Proposition 8



DimCustomer

Column Name	Data Type	Allow Nulls
CustomerKey	int	☐
GeographyKey	int	☑
CustomerAlternateKey	nvarchar(15)	☐
Title	nvarchar(8)	☑
FirstName	nvarchar(50)	☑
MiddleName	nvarchar(50)	☑
LastName	nvarchar(50)	☑
NameStyle	bit	☑
BirthDate	date	☑
MaritalStatus	nchar(1)	☑
Suffix	nvarchar(10)	☑
Gender	nvarchar(1)	☑
EmailAddress	nvarchar(50)	☑
YearlyIncome	money	☑
TotalChildren	tinyint	☑
NumberChildrenAtHome	tinyint	☑
EnglishEducation	nvarchar(40)	☑
SpanishEducation	nvarchar(40)	☑
FrenchEducation	nvarchar(40)	☑
EnglishOccupation	nvarchar(100)	☑
SpanishOccupation	nvarchar(100)	☑
FrenchOccupation	nvarchar(100)	☑
HouseOwnerFlag	nchar(1)	☑
NumberCarsOwned	tinyint	☑
AddressLine1	nvarchar(120)	☑
AddressLine2	nvarchar(120)	☑
Phone	nvarchar(20)	☑
DateFirstPurchase	date	☑
CommuteDistance	nvarchar(15)	☑
		☐

FactInternetSales

Column Name	Data Type	Allow Nulls
ProductKey	int	☐
OrderDateKey	int	☐
DueDateKey	int	☐
ShipDateKey	int	☐
CustomerKey	int	☐
PromotionKey	int	☐
CurrencyKey	int	☐
SalesTerritoryKey	int	☐
SalesOrderNumber	nvarchar(20)	☐
SalesOrderLineNumber	tinyint	☐
RevisionNumber	tinyint	☐
OrderQuantity	smallint	☐
UnitPrice	money	☐
ExtendedAmount	money	☐
UnitPriceDiscountPct	float	☐
DiscountAmount	float	☐
ProductStandardCost	money	☐
TotalProductCost	money	☐
SalesAmount	money	☐
TaxAmt	money	☐
Freight	money	☐
CarrierTrackingNumber	nvarchar(25)	☑
CustomerPONumber	nvarchar(25)	☑
OrderDate	datetime	☑
DueDate	datetime	☑
ShipDate	datetime	☑
		☐

DimProduct

Column Name	Data Type	Allow Nulls
ProductKey	int	☐
ProductAlternateKey	nvarchar(25)	☑
ProductSubcategoryKey	int	☑
WeightUnitMeasureCode	nchar(3)	☑
SizeUnitMeasureCode	nchar(3)	☑
EnglishProductName	nvarchar(50)	☐
SpanishProductName	nvarchar(50)	☐
FrenchProductName	nvarchar(50)	☐
StandardCost	money	☑
FinishedGoodsFlag	bit	☐
Color	nvarchar(15)	☐
SafetyStockLevel	smallint	☑
ReorderPoint	smallint	☑
ListPrice	money	☑
Size	nvarchar(50)	☑
SizeRange	nvarchar(50)	☑
Weight	float	☑
DaysToManufacture	int	☑
ProductLine	nchar(2)	☑
DealerPrice	money	☑
Class	nchar(2)	☑
Style	nchar(2)	☑
ModelName	nvarchar(50)	☑
LargePhoto	varbinary(MAX)	☑
EnglishDescription	nvarchar(400)	☑
FrenchDescription	nvarchar(400)	☑
ChineseDescription	nvarchar(400)	☑
ArabicDescription	nvarchar(400)	☑
HebrewDescription	nvarchar(400)	☑
ThaiDescription	nvarchar(400)	☑
GermanDescription	nvarchar(400)	☑
JapaneseDescription	nvarchar(400)	☑
TurkishDescription	nvarchar(400)	☑
StartDate	datetime	☑
EndDate	datetime	☑
Status	nvarchar(7)	☑

Explanation:

Improved from a previous query that returned too many rows of data which processed slowly and heavily. Returns only top 10 (most recent) descriptive internet orders from customers, since all orders would be 4000+ rows.

Figure 8C: Tables for SQL query components

Select clause

Table name:	Column name:
dbo.DimCustomer	EmailAddress, CustomerKey
dbo.FactInternetSales	SalesAmount, SalesOrderNumber, ProductKey, OrderDate
dbo.DimProduct	ProductKey, EnglishProductName

Order by (optional, only if exist)

Table name	Column name	Sort order
dbo.FactInternetSales	OrderDate	DESC

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 8D: Formatted SQL Query for Proposition 8

```
USE AdventureWorksDW2017;

SELECT TOP 10 sales.CustomerKey
        ,cust.EmailAddress
        ,sales.SalesOrderNumber
        ,prod.EnglishProductName
        ,sales.SalesAmount
        ,sales.OrderDate
FROM dbo.DimCustomer AS cust
LEFT OUTER JOIN (
        dbo.FactInternetSales AS sales INNER JOIN dbo.DimProduct AS prod ON sales.ProductKey = prod.ProductKey
        ) ON sales.CustomerKey = cust.CustomerKey
ORDER BY sales.OrderDate DESC
```

Figure 8E: Query Output for Proposition 8

Results | Messages

	CustomerKey	EmailAddress	SalesOrderNumber	EnglishProductName	SalesAmount	OrderDate
1	19585	kristi33@adventure-works.com	SO75089	LL Road Tire	21.49	2014-01-28 00:00:00.000
2	14680	marvin8@adventure-works.com	SO75088	Road Tire Tube	3.99	2014-01-28 00:00:00.000
3	16170	juan8@adventure-works.com	SO75093	Road Tire Tube	3.99	2014-01-28 00:00:00.000
4	20601	carrie7@adventure-works.com	SO75091	LL Road Tire	21.49	2014-01-28 00:00:00.000
5	19585	kristi33@adventure-works.com	SO75089	Road Tire Tube	3.99	2014-01-28 00:00:00.000
6	26564	franklin2@adventure-works.com	SO75092	LL Road Tire	21.49	2014-01-28 00:00:00.000
7	23381	bianca17@adventure-works.com	SO75098	ML Mountain Tire	29.99	2014-01-28 00:00:00.000
8	20601	carrie7@adventure-works.com	SO75091	Road Tire Tube	3.99	2014-01-28 00:00:00.000
9	13350	isaac3@adventure-works.com	SO75100	Water Bottle - 30 oz.	4.99	2014-01-28 00:00:00.000
10	26564	franklin2@adventure-works.com	SO75092	Road Tire Tube	3.99	2014-01-28 00:00:00.000

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (54) | AdventureWorksDW2017 | 00:00:00 | 10 rows

JSON:

Sampled 4 rows for JSON Output with total number of rows returned (10)

Figure 8F: Formatted SQL Query with JSON for Proposition 8

```
USE AdventureWorksDW2017;

SELECT TOP 10 sales.CustomerKey
        ,cust.EmailAddress
        ,sales.SalesOrderNumber
        ,prod.EnglishProductName
        ,sales.SalesAmount
        ,sales.OrderDate
FROM dbo.DimCustomer AS cust
LEFT OUTER JOIN (
        dbo.FactInternetSales AS sales INNER JOIN dbo.DimProduct AS prod ON sales.ProductKey = prod.ProductKey
        ) ON sales.CustomerKey = cust.CustomerKey
ORDER BY sales.OrderDate DESC
FOR JSON PATH
        ,ROOT('CustomerOrder');
```

Figure 8G: Formatted JSON Output for Proposition 8


```
{
   "CustomerOrder":[
      {
         "CustomerKey":19585,
         "EmailAddress":"kristi33@adventure-works.com",
         "SalesOrderNumber":"SO75089",
         "EnglishProductName":"Sport-100 Helmet, Blue",
         "SalesAmount":34.9900,
         "OrderDate":"2014-01-28T00:00:00"
      },
      {
         "CustomerKey":14680,
         "EmailAddress":"marvin8@adventure-works.com",
         "SalesOrderNumber":"SO75088",
         "EnglishProductName":"ML Road Tire",
         "SalesAmount":24.9900,
         "OrderDate":"2014-01-28T00:00:00"
      },
      {
         "CustomerKey":16170,
         "EmailAddress":"juan8@adventure-works.com",
         "SalesOrderNumber":"SO75093",
         "EnglishProductName":"HL Road Tire",
         "SalesAmount":32.6000,
         "OrderDate":"2014-01-28T00:00:00"
      },
      {
         "CustomerKey":11927,
         "EmailAddress":"nicole32@adventure-works.com",
         "SalesOrderNumber":"SO75085",
         "EnglishProductName":"AWC Logo Cap",
         "SalesAmount":8.9900,
         "OrderDate":"2014-01-28T00:00:00"
      }
   ]
}
```

Proposition 9 (Improved Complex)

Proposition 9: Daily average temperature of warehouse vehicle compared to overall average and average temperature of the cold room for each corresponding day (WideWorldImporters)

Model Diagrams:

Figure 9A: Key View Model for Proposition 9

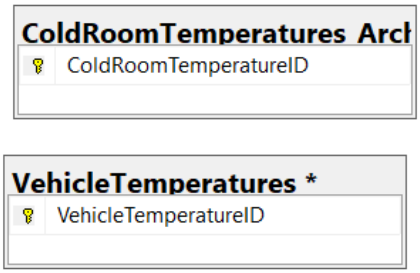


Figure 9B: Standard View Model for Proposition 9

ColdRoomTemperatures_Archive (Warehouse) *

Column Name	Data Type	Allow Nulls
ColdRoomTemperatureID	bigint	☐
ColdRoomSensorNumber	int	☐
RecordedWhen	datetime2(7)	☐
Temperature	decimal(10, 2)	☐
ValidFrom	datetime2(7)	☐
ValidTo	datetime2(7)	☐
		☐

VehicleTemperatures *

Column Name	Data Type	Allow Nulls
VehicleTemperatureID	bigint	☐
VehicleRegistration	nvarchar(20)	☐
ChillerSensorNumber	int	☐
RecordedWhen	datetime2(7)	☐
Temperature	decimal(10, 2)	☐
FullSensorData	nvarchar(1000)	☑
IsCompressed	bit	☐
CompressedSensorData	varbinary(MAX)	☑
		☐

Explanation:
A function improved this query, as the implementation of a variable allows for flexibility in comparison of temperatures. In this query, the variable is set to the overall average of the Vehicle temperature. The function compares and results in only the averaged daily temperature and days less than the overall average. CAST is used to group multiple readings per day (then averaged), and used for adjusting decimal places. The averaged temperatures are then compared to the average daily ColdRoom temperatures.

Figure 9C: Tables for SQL query components

Select clause

Table name:	Column name:
Warehouse.VehicleTemperatures	Temperature, RecordedWhen
ColdRoomTemperatures_Archive	Temperature, RecordedWhen

Order by (optional, only if exist)

Table name	Column name	Sort order
Warehouse.VehicleTemperatures	RecordedDay (RecordedWhen)	asc/desc

Query:
All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 9D: Formatted SQL Query for Proposition 9

```sql
USE WideWorldImporters;

DROP FUNCTION

IF EXISTS dbo.VehTemperatures;GO
        CREATE FUNCTION dbo.VehTemperatures (@temp AS FLOAT)
        RETURNS TABLE
        AS
        RETURN

        SELECT CAST(RecordedWhen AS DATE) AS RecordDay
                ,CAST(AVG(Temperature) AS DECIMAL(5, 2)) AS VehicleAvgTemp
        FROM Warehouse.VehicleTemperatures
        WHERE Temperature < @temp
        GROUP BY CAST(RecordedWhen AS DATE)
GO

DECLARE @temp AS FLOAT;

SELECT @temp = AVG(Temperature)
FROM Warehouse.ColdRoomTemperatures_Archive;

SELECT VH.RecordDay
        ,VH.VehicleAvgTemp
        ,ColdRmAvgTemp
FROM dbo.VehTemperatures(@temp) AS VH
INNER JOIN (
        SELECT CAST(RecordedWhen AS DATE) AS RecordDay
                ,CAST(AVG(Temperature) AS DECIMAL(5, 2)) AS ColdRmAvgTemp
        FROM Warehouse.ColdRoomTemperatures_Archive
        GROUP BY CAST(RecordedWhen AS DATE)
        ) AS CR ON VH.RecordDay = CR.RecordDay
ORDER BY RecordDay
```

Figure 9E: Query Output for Proposition 9

Results | Messages

	RecordDay	VehicleAvgTemp	ColdRmAvgTemp
1	2016-01-01	3.51	4.00
2	2016-01-02	3.48	3.99
3	2016-01-03	3.51	4.00
4	2016-01-04	3.47	4.00
5	2016-01-05	3.48	4.00
6	2016-01-06	3.52	4.00
7	2016-01-07	3.47	4.00
8	2016-01-08	3.48	4.00
9	2016-01-09	3.47	4.00
10	2016-01-10	3.49	4.00
11	2016-01-11	3.49	4.00
12	2016-01-12	3.50	4.00
13	2016-01-13	3.50	4.00
14	2016-01-14	3.53	4.01
15	2016-01-15	3.50	4.01
16	2016-01-16	3.49	4.00
17	2016-01-17	3.50	4.00

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (56) | WideWorldImporters | 00:00:04 | 152 rows

JSON:

Sample JSON Output of 5 rows out of total number of rows returned (152)

Figure 9F: Formatted SQL Query with JSON for Proposition 9

```sql
USE WideWorldImporters;

DROP FUNCTION

IF EXISTS dbo.VehTemperatures;GO
        CREATE FUNCTION dbo.VehTemperatures (@temp AS FLOAT)
        RETURNS TABLE
        AS
        RETURN

        SELECT CAST(RecordedWhen AS DATE) AS RecordDay
                ,CAST(AVG(Temperature) AS DECIMAL(5, 2)) AS VehicleAvgTemp
        FROM Warehouse.VehicleTemperatures
        WHERE Temperature < @temp
        GROUP BY CAST(RecordedWhen AS DATE)
GO

DECLARE @temp AS FLOAT;

SELECT @temp = AVG(Temperature)
FROM Warehouse.ColdRoomTemperatures_Archive;

SELECT VH.RecordDay
        ,VH.VehicleAvgTemp
        ,ColdRmAvgTemp
FROM dbo.VehTemperatures(@temp) AS VH
INNER JOIN (
        SELECT CAST(RecordedWhen AS DATE) AS RecordDay
                ,CAST(AVG(Temperature) AS DECIMAL(5, 2)) AS ColdRmAvgTemp
        FROM Warehouse.ColdRoomTemperatures_Archive
        GROUP BY CAST(RecordedWhen AS DATE)
        ) AS CR ON VH.RecordDay = CR.RecordDay
ORDER BY RecordDay
FOR JSON PATH
        ,ROOT('Temperatures');
```

Figure 9G: Formatted JSON Output for Proposition 9


```
{
   "Temperatures":[
      {
         "RecordDay":"2016-04-24",
         "VehicleAvgTemp":3.52,
         "ColdRmAvgTemp":4.00
      },
      {
         "RecordDay":"2016-05-11",
         "VehicleAvgTemp":3.45,
         "ColdRmAvgTemp":4.00
      },
      {
         "RecordDay":"2016-01-29",
         "VehicleAvgTemp":3.52,
         "ColdRmAvgTemp":4.00
      },
      {
         "RecordDay":"2016-02-15",
         "VehicleAvgTemp":3.50,
         "ColdRmAvgTemp":4.00
      },
      {
         "RecordDay":"2016-02-01",
         "VehicleAvgTemp":3.51,
         "ColdRmAvgTemp":4.00
      }
   ]
}
```