https://dataedo.com/download/AdventureWorks.pdf.

Data Source: SalesOrderHeader

Prepare a report (using proper SQL queries) to assess the yearly performance of individual sales representatives working in the AdventureWorks company. The key metrics that we focus on are total sales and the number of orders made by employees. The report should contain the data as shown in Table 1. SQL query + fragment of the result (4 records from ?)

Table 1.1. Result structure for task 1.1

Sales Person	Employee ID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.25	4
Jiang, Stephen	274	2012	453524.52	22
…	…	…	…	…

Rec: 11/58

1.1. Prepare the report without using windowed functions (OVER clause).

Solution:

```
SELECT CONCAT(LastName, ', ',FirstName) AS "Sales Person", BusinessEntityID AS "Employee
ID",  Salesinfo.Year,  Salesinfo.[Sub Total] , Salesinfo.[Number of orders]
FROM Person.Person
INNER JOIN
(SELECT [SalesPersonID], YEAR([OrderDate]) AS [Year],
SUM([SubTotal]) AS [Sub Total],
COUNT([SalesOrderID]) AS [Number of orders]
FROM [AdventureWorks2019].[Sales].[SalesOrderHeader]
GROUP BY [SalesPersonID], YEAR([OrderDate]))Salesinfo
ON Salesinfo.SalesPersonID = Person.BusinessEntityID
ORDER BY [Employee ID], [Year]
```

Sales Person	Employee ID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.2465	4
Jiang, Stephen	274	2012	453524.5233	22
Jiang, Stephen	274	2013	431088.7238	14
Jiang, Stephen	274	2014	178584.3625	8
Blythe, Michael	275	2011	875823.8318	65
Blythe, Michael	275	2012	3375456.8947	148
Blythe, Michael	275	2013	3985374.8995	175
Blythe, Michael	275	2014	1057247.3786	62
Mitchell, Linda	276	2011	1149715.3253	46
Mitchell, Linda	276	2012	3834908.674	151
Mitchell, Linda	276	2013	4111294.9056	162

1.2. Prepare the report using windowed functions (OVER clause).

Solution:

```
SELECT DISTINCT CONCAT(LastName, ', ',FirstName) AS "Sales Person", SalesPersonID AS
"EmployeeID",
YEAR([OrderDate]) AS "Year",
SUM([SubTotal]) OVER (PARTITION BY SalesPersonID, YEAR([OrderDate])) AS "Sub Total",
COUNT(SalesOrderID) OVER (PARTITION BY SalesPersonID, YEAR(OrderDate)) AS [Number of orders]

FROM [AdventureWorks2019].[Sales].[SalesOrderHeader]
INNER JOIN [AdventureWorks2019].[Person].[Person]
ON [Person].[BusinessEntityID] = [SalesOrderHeader].[SalesPersonID]
   ORDER BY [SalesPersonID], [Year]
```

Sales Person	EmployeeID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.2465	4
Jiang, Stephen	274	2012	453524.5233	22
Jiang, Stephen	274	2013	431088.7238	14
Jiang, Stephen	274	2014	178584.3625	8
Blythe, Michael	275	2011	875823.8318	65
Blythe, Michael	275	2012	3375456.8947	148
Blythe, Michael	275	2013	3985374.8995	175
Blythe, Michael	275	2014	1057247.3786	62
Mitchell, Linda	276	2011	1149715.3253	46
Mitchell, Linda	276	2012	3834908.674	151
Mitchell, Linda	276	2013	4111294.9056	162

1.3. Prepare the report using CTE, where first you aggregate the sales data to establish yearly performance metrics, and only then attaching the details of the sales person.

Solution:

```
WITH CTE (SalesPersonID, [Number of orders], [Sub Total], Year) AS
(
SELECT SalesPersonID,
COUNT(SalesOrderID) AS [Number of orders], SUM(SubTotal) ,
YEAR([OrderDate])
 FROM Sales.SalesOrderHeader
 GROUP BY SalesPersonID, YEAR(OrderDate))
SELECT CONCAT(LastName, ', ',FirstName) AS "Sales Person",
 SalesPersonID AS "EmployeeID",
 Year,
 [Sub Total],
 [Number of orders]
FROM CTE INNER JOIN Person.Person ON Person.BusinessEntityID = CTE.SalesPersonID
ORDER BY SalesPersonID, Year
```

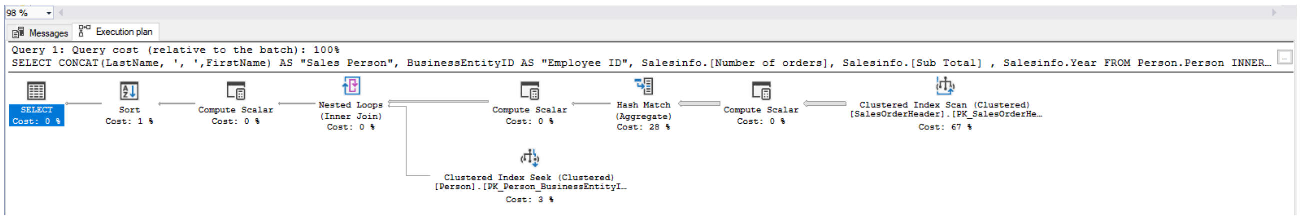
Sales Person	EmployeeID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.2465	4
Jiang, Stephen	274	2012	453524.5233	22
Jiang, Stephen	274	2013	431088.7238	14
Jiang, Stephen	274	2014	178584.3625	8

Blythe, Michael	275	2011	875823.8318	65
Blythe, Michael	275	2012	3375456.8947	148
Blythe, Michael	275	2013	3985374.8995	175
Blythe, Michael	275	2014	1057247.3786	62
Mitchell, Linda	276	2011	1149715.3253	46
Mitchell, Linda	276	2012	3834908.674	151
Mitchell, Linda	276	2013	4111294.9056	162

1.4. Assess the quality of the solutions proposed in the previous two tasks. Which one is more advantageous and why, utilise the execution plans for the three queries.

Solution:

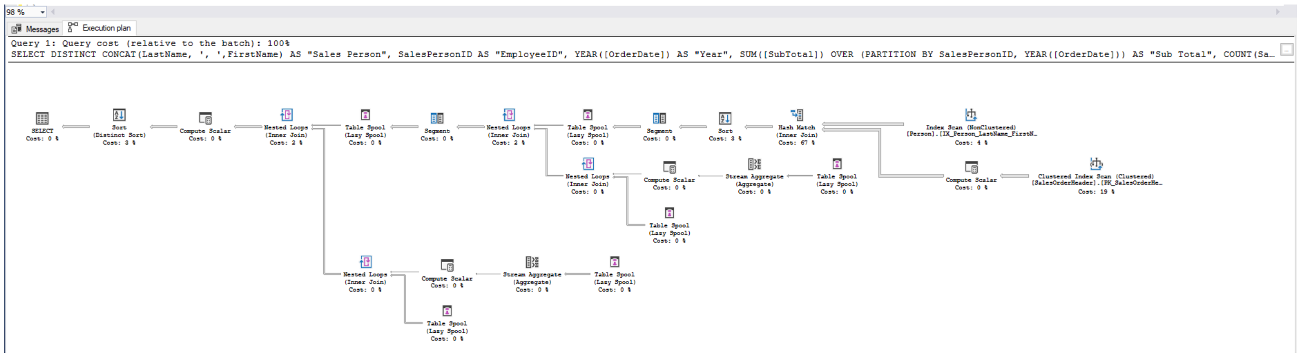
Task 1.1 -> Execution Plan:



```
Query 1: Query cost (relative to the batch): 100%
SELECT CONCAT(LastName, ', ',FirstName) AS "Sales Person", BusinessEntityID AS "Employee ID", Salesinfo.[Number of orders], Salesinfo.[Sub Total] , Salesinfo.Year FROM Person.Person INNER…
```

98 %
Messages
Execution plan
SELECT Cost: 0 %
Sort Cost: 1 %
Compute Scalar Cost: 0 %
Nested Loops (Inner Join) Cost: 0 %
Compute Scalar Cost: 0 %
Hash Match (Aggregate) Cost: 28 %
Compute Scalar Cost: 0 %
Clustered Index Scan (Clustered) [SalesOrderHeader].[PK_SalesOrderHe… Cost: 67 %
Clustered Index Seek (Clustered) [Person].[PK_Person_BusinessEntityI… Cost: 3 %

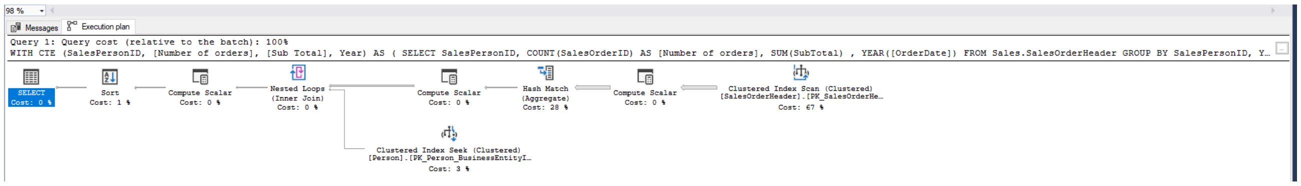
Task 1.2 -> Execution Plan:



```
Query 1: Query cost (relative to the batch): 100%
SELECT DISTINCT CONCAT(LastName, ', ',FirstName) AS "Sales Person", SalesPersonID AS "EmployeeID", YEAR([OrderDate]) AS "Year", SUM([SubTotal]) OVER (PARTITION BY SalesPersonID, YEAR([OrderDate])) AS "Sub Total", COUNT(Sa…
```

98 %
Messages
Execution plan
SELECT Cost: 0 %
Sort (Distinct Sort) Cost: 3 %
Compute Scalar Cost: 0 %
Nested Loops (Inner Join) Cost: 2 %
Table Spool (Lazy Spool) Cost: 0 %
Segment Cost: 0 %
Nested Loops (Inner Join) Cost: 2 %
Table Spool (Lazy Spool) Cost: 0 %
Segment Cost: 0 %
Sort Cost: 3 %
Hash Match (Inner Join) Cost: 67 %
Index Scan (NonClustered) [Person].[IX_Person_LastName_FirstN… Cost: 4 %
Nested Loops (Inner Join) Cost: 0 %
Compute Scalar Cost: 0 %
Stream Aggregate (Aggregate) Cost: 0 %
Table Spool (Lazy Spool) Cost: 0 %
Compute Scalar Cost: 0 %
Clustered Index Scan (Clustered) [SalesOrderHeader].[PK_SalesOrderHe… Cost: 19 %
Table Spool (Lazy Spool) Cost: 0 %
Nested Loops (Inner Join) Cost: 0 %
Compute Scalar Cost: 0 %
Stream Aggregate (Aggregate) Cost: 0 %
Table Spool (Lazy Spool) Cost: 0 %
Table Spool (Lazy Spool) Cost: 0 %

Task 1.3 -> Execution Plan:



```
Query 1: Query cost (relative to the batch): 100%
WITH CTE (SalesPersonID, [Number of orders], [Sub Total], Year) AS ( SELECT SalesPersonID, COUNT(SalesOrderID) AS [Number of orders], SUM(SubTotal) , YEAR([OrderDate]) FROM Sales.SalesOrderHeader GROUP BY SalesPersonID, Y…
```

98 %
Messages
Execution plan
SELECT Cost: 0 %
Sort Cost: 1 %
Compute Scalar Cost: 0 %
Nested Loops (Inner Join) Cost: 0 %
Compute Scalar Cost: 0 %
Hash Match (Aggregate) Cost: 28 %
Compute Scalar Cost: 0 %
Clustered Index Scan (Clustered) [SalesOrderHeader].[PK_SalesOrderHe… Cost: 67 %
Clustered Index Seek (Clustered) [Person].[PK_Person_BusinessEntityI… Cost: 3 %

As far as ı understand of a execution plan is a set of given instruction that indicates the steps of the process when performed during the execution. Moreover, qurey plans are defined to optimize goal to wheather generate or improve the optimum and economical query plan.
In this approach, presented above the execution plan of queries, we can observe that CTE is more faster and effective way to solve task according to execution plan than wheather with OVER clause

or not. The more complex presentation of a execution plan means that its slower and less effective to be used in the query i believe.
More over, CTE is more readable and can be used multiple times in query.
So even though, execution plan of task 1 and task3 are similar, in my opinion the one with using CTE is better option to use it.

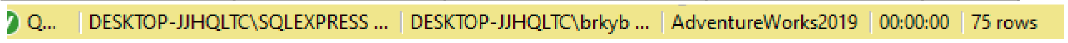
1.5. Pick one of the introduced solutions (1-3) and modify it to include basic data summaries, total subtotal value, and total number of orders for each year and for each employee

Solution:

3 ->

```
WITH CTE (SalesPersonID, [Number of orders], [Sub Total], Year) AS
 (SELECT SalesPersonID, COUNT(SalesOrderID) AS [Number of orders], SUM(SubTotal) AS [Sub
Total], YEAR(OrderDate) AS 'Year'
FROM Sales.SalesOrderHeader GROUP BY ROLLUP(SalesPersonID, YEAR(OrderDate)))
SELECT CONCAT(LastName, ', ',FirstName) AS 'Name', SalesPersonID, [Number of orders], [Sub
Total],
Year FROM CTE
INNER JOIN Person.Person
ON Person.BusinessEntityID = CTE.SalesPersonID
ORDER BY SalesPersonID, Year DESC
```

Name	SalesPersonID	Number of orders	Sub Total	Year
Jiang, Stephen	274	8	178584.3625	2014
Jiang, Stephen	274	14	431088.7238	2013
Jiang, Stephen	274	22	453524.5233	2012
Jiang, Stephen	274	4	28926.2465	2011
Jiang, Stephen	274	48	1092123.8561	NULL
Blythe, Michael	275	62	1057247.3786	2014
Blythe, Michael	275	175	3985374.8995	2013
Blythe, Michael	275	148	3375456.8947	2012
Blythe, Michael	275	65	875823.8318	2011
Blythe, Michael	275	450	9293903.0046	NULL
Mitchell, Linda	276	59	1271088.5216	2014


Q... | DESKTOP-JJHQLTC\SQLEXPRESS ... | DESKTOP-JJHQLTC\brkyb ... | AdventureWorks2019 | 00:00:00 | 75 rows

Rec: 11/75

Conclusions:

Use this section to provide your conclusions:

In conclusion, Even though i didn't have experince to using OVER clause and CTE in the query, at the beginning it was a bit complex for me to get to logic of working principle in the query. However, we

discussed and compare the created queries using windowed functions (OVER clause), without windowed functions and CTE on the execution plan.
Moreover, even though ı am still not sure if the provided query is correct, i had a problem to do and understand the logic of it. Depends on my researches GROUP BY ROLLUP was to only result ı get that using with it.

Task 2

Assume you are a data engineer working in Adventure Works Cycles Company, you are responsible for preparing the data for the analyst who is interested in studying basic sales information from the product and customer perspective. Assume that they will use basic reporting and data visualisation tools, such as pivot tables in Excel and dashboards in Tableau. As such, your task is to prepare a set of new, dedicated, relational structures capable of storing data for basic OLAP processing and further move the data from available transactional sources (AdventureWorks database) using SQL statements. Additional resources on using T-SQL statements to create structures and to move data around different tables and databases are available below:

- *Schema creation - https://msdn.microsoft.com/en-us/library/ms189462.aspx*
- *INSERT INTO...SELECT - https://docs.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15*
- *SELECT INTO clause - http://msdn.microsoft.com/en-us/library/ms188029.aspx*
- *ALTER TABLE clause - https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql*

--- Organize the data into subjects – customer, product, and orders. ---

2.1. Create a database (if it does not exist) with a name that matches your student ID, and then create tables (using CREATE TABLE script) according to the following specification:

a. Dim_Customer (**CustomerID**, FirstName, LastName, TerritoryName, CounrtyRegionName, Group (Geography))
Source data (tables): SalesTerritory, Customer, and Person

b. Dim_Product (**ProductID**, Name, FinishedGoodsFlag, Class, ListPrice, Color, SubCategoryName, CategoryName)
Source data (tables): Product, ProductSubcategory, and ProductCategory

c. Fact_Orders (ProductID, CustomerID, OrderDate, ShipDate, OrderQty, UnitPrice, UnitPriceDiscount, LineTotal)
Source data (tables): SalesOrderDetail, and SalesOrderHeader

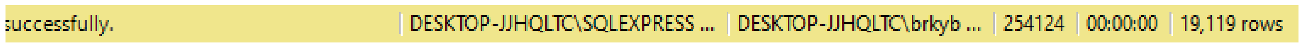
Solution:

Task 2.1a

```
SELECT c.CustomerID, p.FirstName, p.LastName, t.Name AS "TerritoryName", cr.Name AS "Country
Region Name", t.[Group]
INTO [Dim_Customer]
FROM [AdventureWorks2019].[Sales].[Customer] c JOIN [AdventureWorks2019].[Person].[Person] p
ON c.PersonID = p.BusinessEntityID
                                                    JOIN
[AdventureWorks2019].[Sales].[SalesTerritory] t ON c.territoryID = t.territoryID
                                                    JOIN
[AdventureWorks2019].[Person].[CountryRegion] cr ON t.CountryRegionCode =
cr.CountryRegionCode
```

CustomerID	FirstName	LastName	TerritoryName	Country Region Name	Group
29485	Catherine	Abel	Southwest	United States	North America

29486	Kim	Abercrombie	Central	United States	North America
29487	Humberto	Acevedo	Northeast	United States	North America
29484	Gustavo	Achong	Southeast	United States	North America
29488	Pilar	Ackerman	Australia	Australia	Pacific
28866	Aaron	Adams	Southwest	United States	North America
13323	Adam	Adams	Southwest	United States	North America
21139	Alex	Adams	Northwest	United States	North America
29170	Alexandra	Adams	Southwest	United States	North America
19419	Allison	Adams	France	France	Europe
11971	Amanda	Adams	Southwest	United States	North America



```
successfully.    | DESKTOP-JJHQLTC\SQLEXPRESS ... | DESKTOP-JJHQLTC\brkyb ... | 254124 | 00:00:00 | 19,119 rows
```

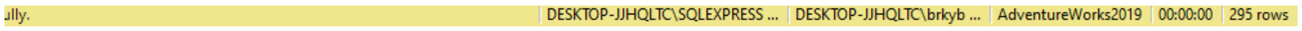
Rec: 11/19,119

Task2.1b

```
SELECT p.ProductId, p.Name, p.FinishedGoodsFlag, p.Class, p.ListPrice, p.Color, s.Name AS
"SubCategoryName", c.Name AS "CategoryName"
INTO  [Dim_Product]
FROM[AdventureWorks2019]. [Production].[Product] p  JOIN
[AdventureWorks2019].[Production].[ProductSubcategory] s ON
p.ProductSubcategoryID=s.ProductSubcategoryID
   JOIN [AdventureWorks2019].[Production].[ProductCategory] c ON
      s.ProductCategoryID=c.ProductCategoryID
```

ProductId	Name	FinishedGoodsFlag	Class	ListPrice	Color	SubCategoryName	CategoryName
680	HL Road Frame - Black, 58	1	H	1431.50	Black	Road Frames	Components
706	HL Road Frame - Red, 58	1	H	1431.50	Red	Road Frames	Components
707	Sport-100 Helmet, Red	1	NULL	34.99	Red	Helmets	Accessories
708	Sport-100 Helmet, Black	1	NULL	34.99	Black	Helmets	Accessories
709	Mountain Bike Socks, M	1	NULL	9.50	White	Socks	Clothing
710	Mountain Bike Socks, L	1	NULL	9.50	White	Socks	Clothing
711	Sport-100 Helmet, Blue	1	NULL	34.99	Blue	Helmets	Accessories
712	AWC Logo Cap	1	NULL	8.99	Multi	Caps	Clothing
713	Long-Sleeve Logo Jersey, S	1	NULL	49.99	Multi	Jerseys	Clothing
714	Long-Sleeve Logo Jersey, M	1	NULL	49.99	Multi	Jerseys	Clothing

715	Long-Sleeve Logo Jersey, L	1	NULL	49.99	Multi	Jerseys	Clothing



```
ully. | DESKTOP-JJHQLTC\SQLEXPRESS ... | DESKTOP-JJHQLTC\brkyb ... | AdventureWorks2019 | 00:00:00 | 295 rows
```

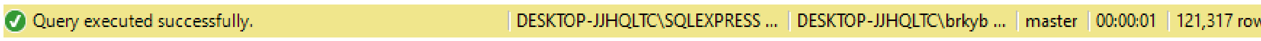
Rec: 11/295

Task 2.1c

```
SELECT p.ProductID, o.CustomerID, o.OrderDate, o.ShipDate, p.OrderQty, p.UnitPrice,
p.UnitPriceDiscount, p.LineTotal
INTO [Fact_Orders]
   FROM [AdventureWorks2019].[Sales].[SalesOrderDetail] p JOIN
      [AdventureWorks2019].[Sales].[SalesOrderHeader] o ON p.SalesOrderID=o.SalesOrderID
```

ProductID	CustomerID	OrderDate	ShipDate	OrderQty	UnitPrice	UnitPriceDiscount	LineTotal
776	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	2024.994	0.00	2024.994000
777	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	3	2024.994	0.00	6074.982000
778	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	2024.994	0.00	2024.994000
771	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	2039.994	0.00	2039.994000
772	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	2039.994	0.00	2039.994000
773	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	2	2039.994	0.00	4079.988000
774	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	2039.994	0.00	2039.994000
714	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	3	28.8404	0.00	86.521200
716	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	28.8404	0.00	28.840400
709	29825	2011-05-31	2011-06-07	6	5.70	0.00	34.200000

		00:00:00.000	00:00:00.000				
712	29825	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	2	5.1865	0.00	10.373000



Query executed successfully. | DESKTOP-JJHQLTC\SQLEXPRESS ... | DESKTOP-JJHQLTC\brkyb ... | master | 00:00:01 | 121,317 row

Rec: 11/121.317