Task 1

```
CREATE PARTITION FUNCTION monthlyConsumptionYears(CHAR(4))
AS RANGE LEFT FOR VALUES (2020, 2021, 2022, 2023);

CREATE PARTITION SCHEME monthlyConsumptionYearsPartition
AS PARTITION monthlyConsumptionYears
TO ([PRIMARY], [PRIMARY], [PRIMARY], [PRIMARY], [PRIMARY]);

ALTER TABLE Energy.MonthlyConsumption DROP CONSTRAINT
PK_MonthlyConsumption;

CREATE CLUSTERED INDEX PK_MonthlyConsumption
ON Energy.MonthlyConsumption (Year, Month,
DistrictMunicipalityParishCode, VoltageLevel)
ON monthlyConsumptionYearsPartition(Year);

SELECT * FROM sys.partitions as p
WHERE
    p.rows IN (
        SELECT COUNT(*) FROM Energy.MonthlyConsumption
    ) OR p.rows IN (
        SELECT COUNT(*) FROM Energy.MonthlyConsumption
        GROUP BY Year
    )
ORDER BY p.rows DESC
```

To separate a table into multiple partitions it is necessary to create N partitions and to distinguish from them, a partition function, so that we can know in which partition to look for, given a certain record, in this case, we separate them by Year.

Before splitting the table into partitions, it is mandatory to remove the primary key's clustered index, since it organises the records by their primary key inside a single partition. Only after, a new clustered index can be created following the newly created partition function, separating each record into its respective partition.

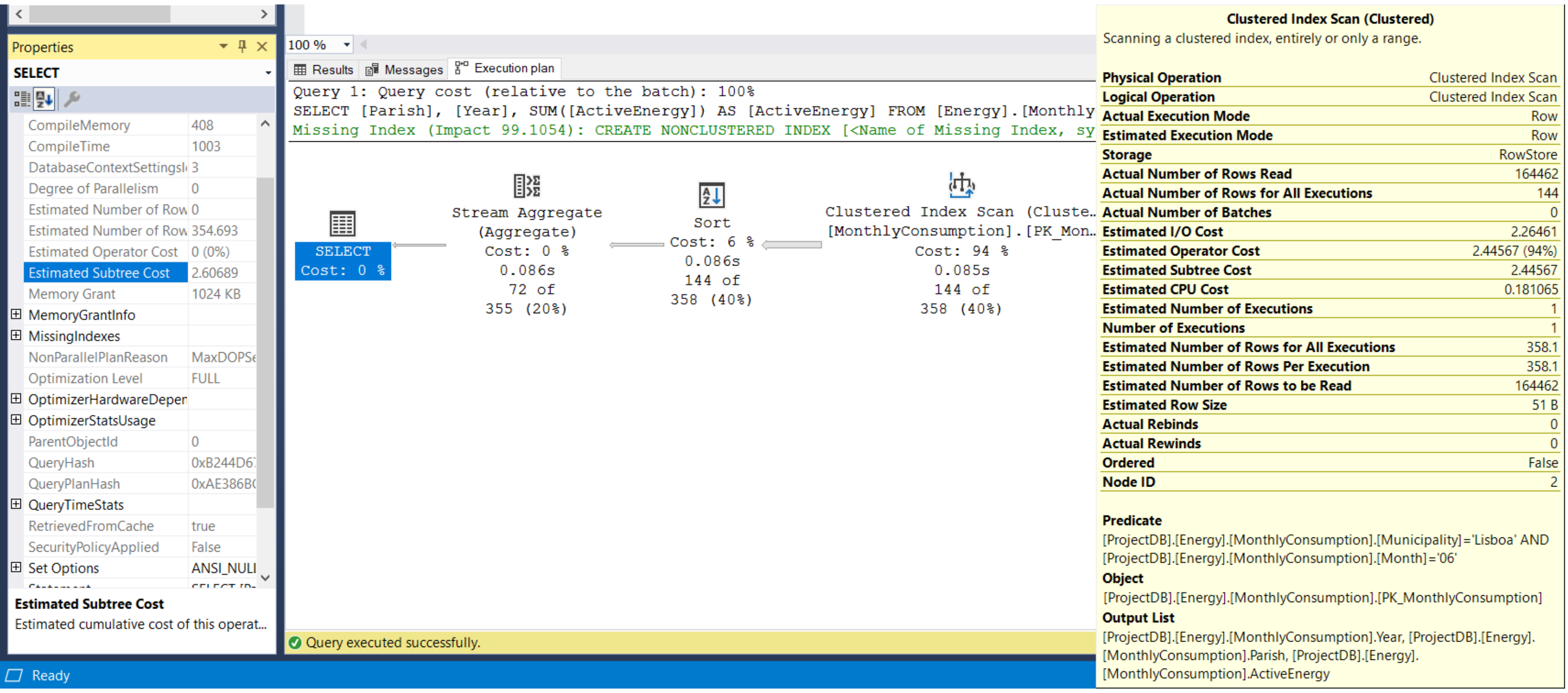
Task 2

Since the column Year in used in the primary key clustered index, its information is already present in the pointer to the data table. This means that its presence in not necessary in a new index.

It is also important to note that the table is already ordered by year due to the clustered index.

Because the column ActiveEnergy is only used inside a sum statement, it is only important to include their values into the end of the tree instead of sorting the index tree by them.

Initial state



CompileMemory	408
CompileTime	1003
DatabaseContextSettingsI	3
Degree of Parallelism	0
Estimated Number of Row	0
Estimated Number of Row	354.693
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2.60689
Memory Grant	1024 KB
⊞ MemoryGrantInfo	
⊞ MissingIndexes	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDeper	
⊞ OptimizerStatsUsage	
ParentObjectId	0
QueryHash	0xB244D6
QueryPlanHash	0xAE386B
⊞ QueryTimeStats	
RetrievedFromCache	true
SecurityPolicyApplied	False
⊞ Set Options	ANSI_NULI

Estimated Subtree Cost
Estimated cumulative cost of this operat...

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [ActiveEnergy] FROM [Energy].[Monthly
Missing Index (Impact 99.1054): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sy
```

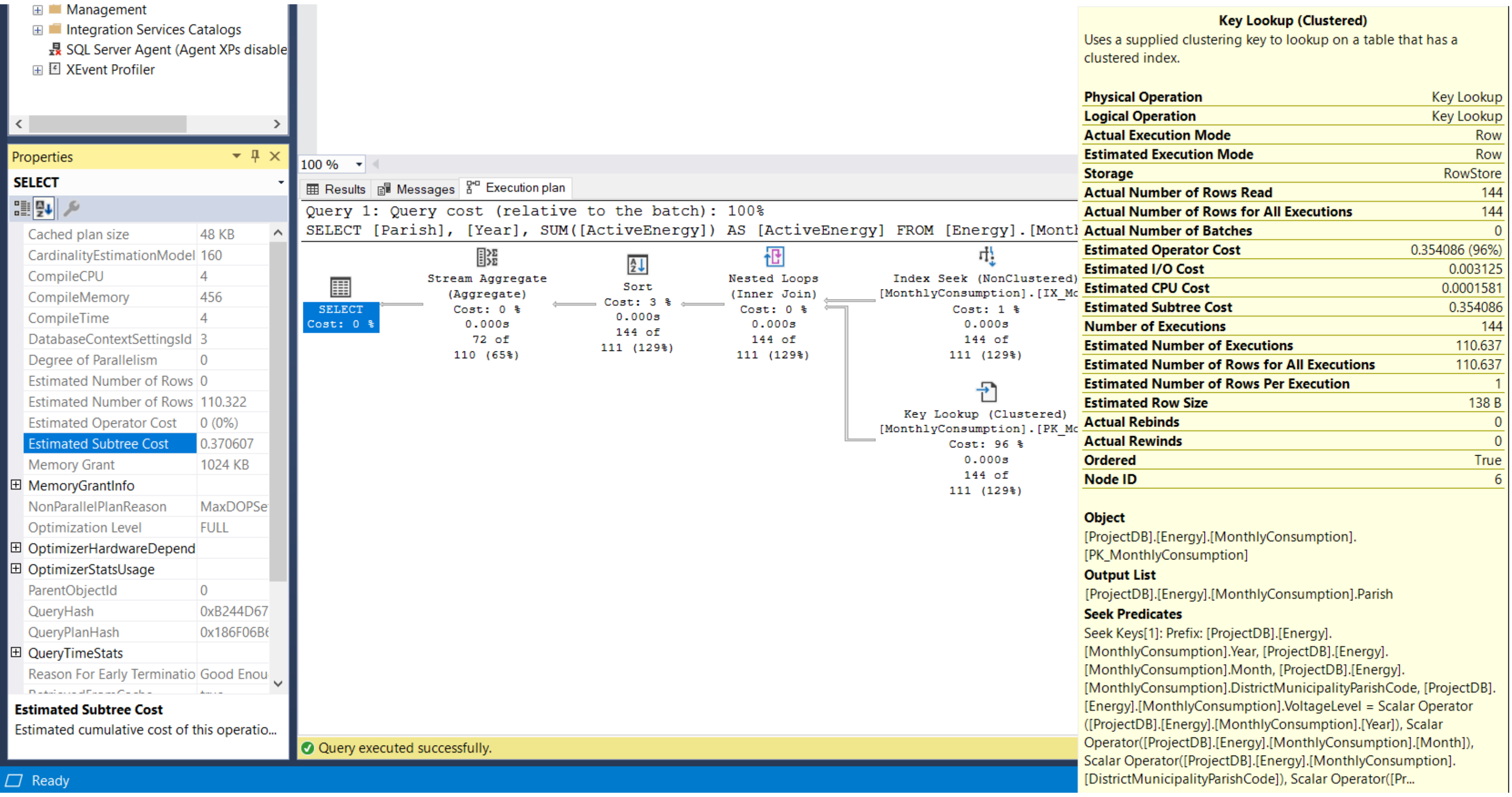
Clustered Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	164462
Actual Number of Rows for All Executions	144
Actual Number of Batches	0
Estimated I/O Cost	2.26461
Estimated Operator Cost	2.44567 (94%)
Estimated Subtree Cost	2.44567
Estimated CPU Cost	0.181065
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	358.1
Estimated Number of Rows Per Execution	358.1
Estimated Number of Rows to be Read	164462
Estimated Row Size	51 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	2

Predicate
[ProjectDB].[Energy].[MonthlyConsumption].[Municipality]='Lisboa' AND [ProjectDB].[Energy].[MonthlyConsumption].[Month]='06'
Object
[ProjectDB].[Energy].[MonthlyConsumption].[PK_MonthlyConsumption]
Output List
[ProjectDB].[Energy].[MonthlyConsumption].Year, [ProjectDB].[Energy].[MonthlyConsumption].Parish, [ProjectDB].[Energy].[MonthlyConsumption].ActiveEnergy

Since the main purpose of an index is to speed up the search of some given records, there is no doubt that an index with Municipality and Month would help greatly, speeding up the look up for the records with the given values, so the first index to try would be:

```sql
DROP INDEX IF EXISTS IX_Month_Municipality ON Energy.MonthlyConsumption
CREATE NONCLUSTERED INDEX IX_Month_Municipality ON
Energy.MonthlyConsumption (Month, Municipality) INCLUDE (ActiveEnergy)
```



Cached plan size	48 KB
CardinalityEstimationModel	160
CompileCPU	4
CompileMemory	456
CompileTime	4
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	110.322
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.370607
Memory Grant	1024 KB
MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
OptimizerHardwareDepend	
OptimizerStatsUsage	
ParentObjectId	0
QueryHash	0xB244D67
QueryPlanHash	0x186F06B6
QueryTimeStats	
Reason For Early Terminatio	Good Enou

Estimated Subtree Cost
Estimated cumulative cost of this operatio...

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [ActiveEnergy] FROM [Energy].[Montl
```

```
SELECT      Stream Aggregate   Sort          Nested Loops   Index Seek (NonClustered)
Cost: 0 %   (Aggregate)        Cost: 3 %     (Inner Join)   [MonthlyConsumption].[IX_Mc
            Cost: 0 %          0.000s        Cost: 0 %      Cost: 1 %
            0.000s             144 of        0.000s         0.000s
            72 of              111 (129%)    144 of         144 of
            110 (65%)                        111 (129%)     111 (129%)

                                                            Key Lookup (Clustered)
                                                            [MonthlyConsumption].[PK_Mc
                                                            Cost: 96 %
                                                            0.000s
                                                            144 of
                                                            111 (129%)
```

Query executed successfully.

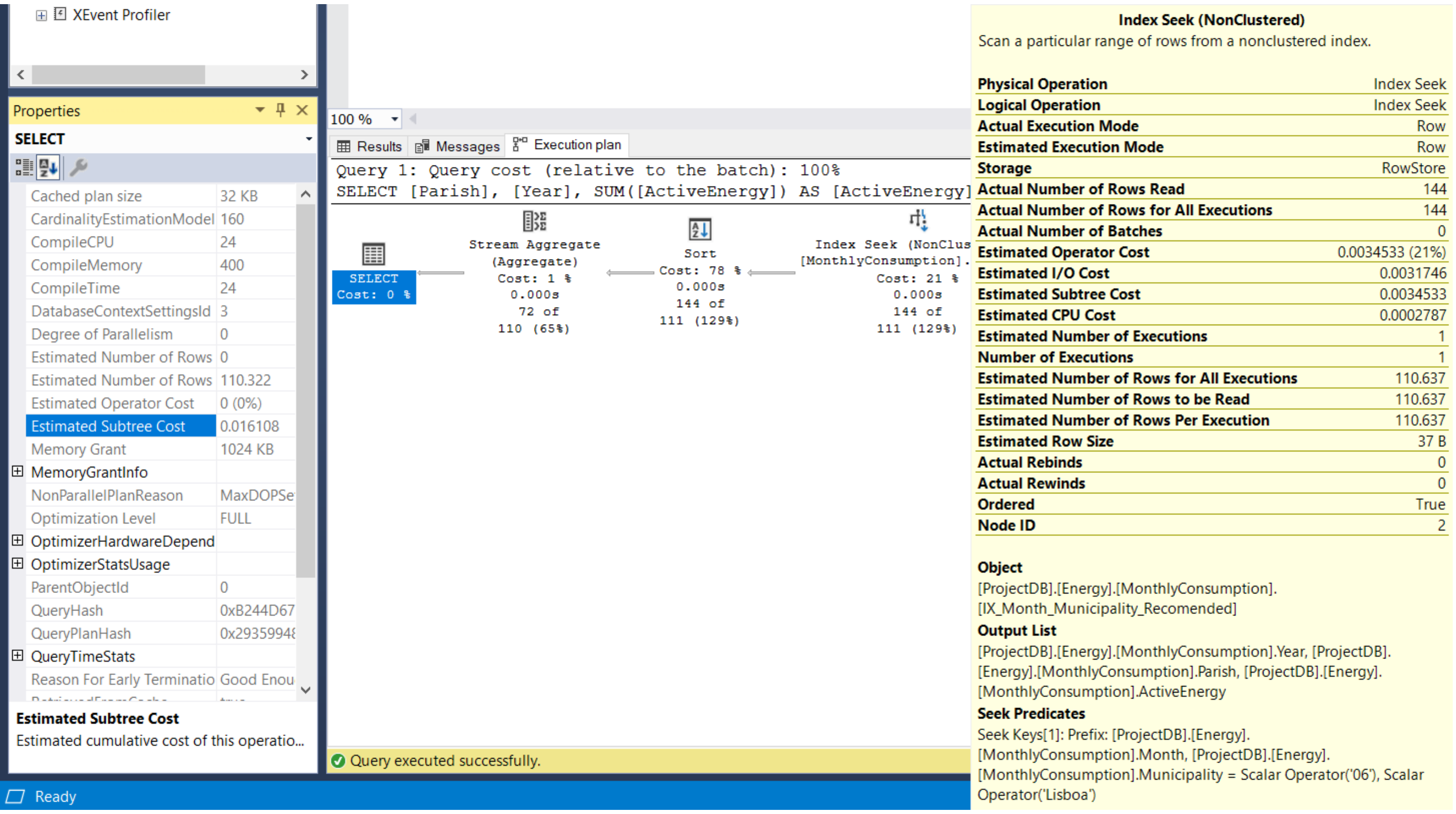
Key Lookup (Clustered)
Uses a supplied clustering key to lookup on a table that has a clustered index.

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	144
Actual Number of Rows for All Executions	144
Actual Number of Batches	0
Estimated Operator Cost	0.354086 (96%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.354086
Number of Executions	144
Estimated Number of Executions	110.637
Estimated Number of Rows for All Executions	110.637
Estimated Number of Rows Per Execution	1
Estimated Row Size	138 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	6

Object
[ProjectDB].[Energy].[MonthlyConsumption].[PK_MonthlyConsumption]
Output List
[ProjectDB].[Energy].[MonthlyConsumption].Parish
Seek Predicates
Seek Keys[1]: Prefix: [ProjectDB].[Energy].[MonthlyConsumption].Year, [ProjectDB].[Energy].[MonthlyConsumption].Month, [ProjectDB].[Energy].[MonthlyConsumption].DistrictMunicipalityParishCode, [ProjectDB].[Energy].[MonthlyConsumption].VoltageLevel = Scalar Operator([ProjectDB].[Energy].[MonthlyConsumption].[Year]), Scalar Operator([ProjectDB].[Energy].[MonthlyConsumption].[Month]), Scalar Operator([ProjectDB].[Energy].[MonthlyConsumption].[DistrictMunicipalityParishCode]), Scalar Operator([Pr...

There is still a key lookup being executed using 96% of the total cost. It's only used to get the data for the Parish column. Since it is on a group by and a order by statements, it should go inside the main index clause, but let's include it first, appending its values to the end of the tree.

```
DROP INDEX IF EXISTS IX_Month_Municipality_Recomended ON
Energy.MonthlyConsumption
CREATE NONCLUSTERED INDEX IX_Month_Municipality_Recomended ON
Energy.MonthlyConsumption (Month, Municipality) INCLUDE (Parish,
ActiveEnergy)
```



Cached plan size	32 KB
CardinalityEstimationModel	160
CompileCPU	24
CompileMemory	400
CompileTime	24
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	110.322
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.016108
Memory Grant	1024 KB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0
QueryHash	0xB244D67
QueryPlanHash	0x2935994
⊞ QueryTimeStats	
Reason For Early Terminatio	Good Enou

Estimated Subtree Cost
Estimated cumulative cost of this operatio...

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [ActiveEnergy]
```

Query executed successfully.

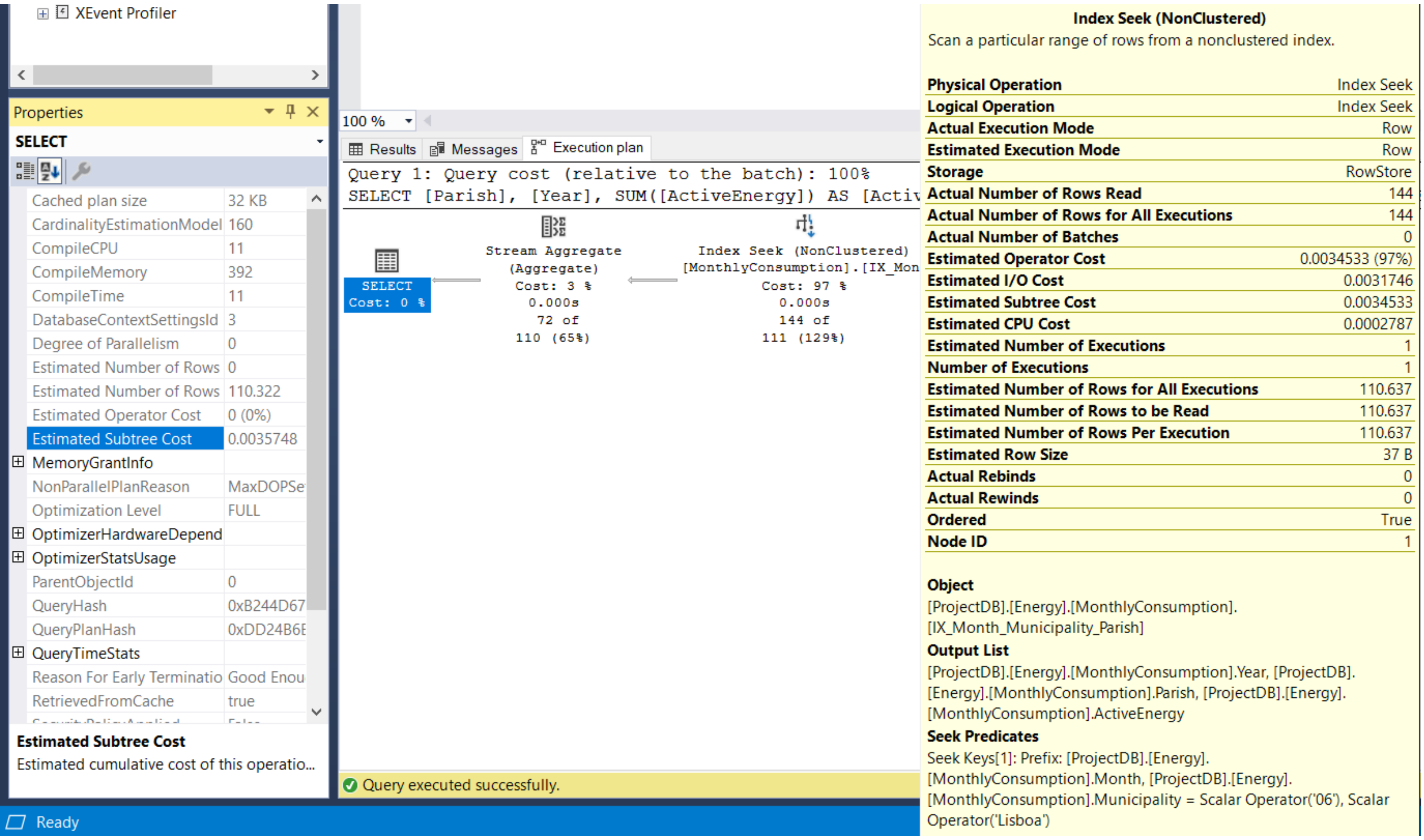
Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	144
Actual Number of Rows for All Executions	144
Actual Number of Batches	0
Estimated Operator Cost	0.0034533 (21%)
Estimated I/O Cost	0.0031746
Estimated Subtree Cost	0.0034533
Estimated CPU Cost	0.0002787
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	110.637
Estimated Number of Rows to be Read	110.637
Estimated Number of Rows Per Execution	110.637
Estimated Row Size	37 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	2

Object
[ProjectDB].[Energy].[MonthlyConsumption].[IX_Month_Municipality_Recomended]
Output List
[ProjectDB].[Energy].[MonthlyConsumption].Year, [ProjectDB].[Energy].[MonthlyConsumption].Parish, [ProjectDB].[Energy].[MonthlyConsumption].ActiveEnergy
Seek Predicates
Seek Keys[1]: Prefix: [ProjectDB].[Energy].[MonthlyConsumption].Month, [ProjectDB].[Energy].[MonthlyConsumption].Municipality = Scalar Operator('06'), Scalar Operator('Lisboa')

There was a clear benefit on removing that Key Lookup, turning its 0.354 estimated cost into the 0.000345 estimated cost of the hole Index Seek. But now the Sort is the step that is slowing down the most. Moving the Parish column to be used in the tree will automatically sort its values, removing completely the need for a Sort block.

```sql
DROP INDEX IF EXISTS IX_Month_Municipality_Parish ON
Energy.MonthlyConsumption
CREATE NONCLUSTERED INDEX IX_Month_Municipality_Parish ON
Energy.MonthlyConsumption (Month, Municipality, Parish) INCLUDE
(ActiveEnergy)
```



SELECT

Cached plan size	32 KB
CardinalityEstimationModel	160
CompileCPU	11
CompileMemory	392
CompileTime	11
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	110.322
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0035748
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0
QueryHash	0xB244D67
QueryPlanHash	0xDD24B6E
⊞ QueryTimeStats	
Reason For Early Terminatio	Good Enou
RetrievedFromCache	true

Estimated Subtree Cost
Estimated cumulative cost of this operatio...

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [Activ
```

```
SELECT       Stream Aggregate    Index Seek (NonClustered)
Cost: 0 %    (Aggregate)         [MonthlyConsumption].[IX_Mon
             Cost: 3 %           Cost: 97 %
             0.000s              0.000s
             72 of               144 of
             110 (65%)           111 (129%)
```

Query executed successfully.

Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	144
Actual Number of Rows for All Executions	144
Actual Number of Batches	0
Estimated Operator Cost	0.0034533 (97%)
Estimated I/O Cost	0.0031746
Estimated Subtree Cost	0.0034533
Estimated CPU Cost	0.0002787
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	110.637
Estimated Number of Rows to be Read	110.637
Estimated Number of Rows Per Execution	110.637
Estimated Row Size	37 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object
[ProjectDB].[Energy].[MonthlyConsumption].[IX_Month_Municipality_Parish]
Output List
[ProjectDB].[Energy].[MonthlyConsumption].Year, [ProjectDB].[Energy].[MonthlyConsumption].Parish, [ProjectDB].[Energy].[MonthlyConsumption].ActiveEnergy
Seek Predicates
Seek Keys[1]: Prefix: [ProjectDB].[Energy].[MonthlyConsumption].Month, [ProjectDB].[Energy].[MonthlyConsumption].Municipality = Scalar Operator('06'), Scalar Operator('Lisboa')

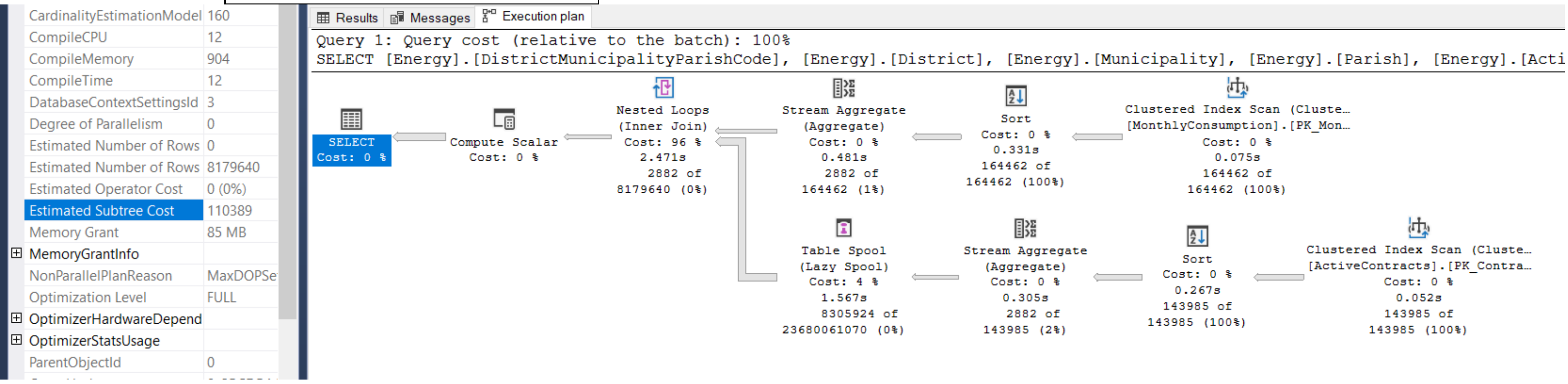
Concluding, the best Index found uses the columns Month, Municipality and Parish, in this order, so after filtering the given Month and Municipality, the Parish is already sorted in the tree, and accesses the Year and ActiveEnergy data from the primary key pointer and the include clause, respectively.

Task 3

The following SQL instructions were added at the end of the given query.

```
OPTION (LOOP JOIN)
```

110389

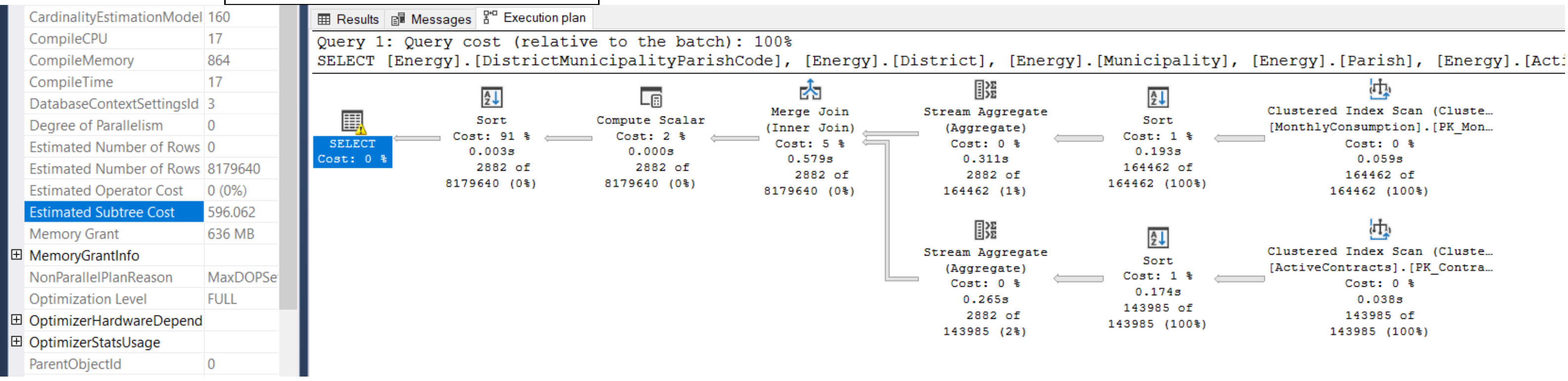


CardinalityEstimationModel	160
CompileCPU	12
CompileMemory	904
CompileTime	12
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	110389
Memory Grant	85 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality], [Energy].[Parish], [Energy].[Acti
```

```
OPTION (MERGE JOIN)
```

596.062

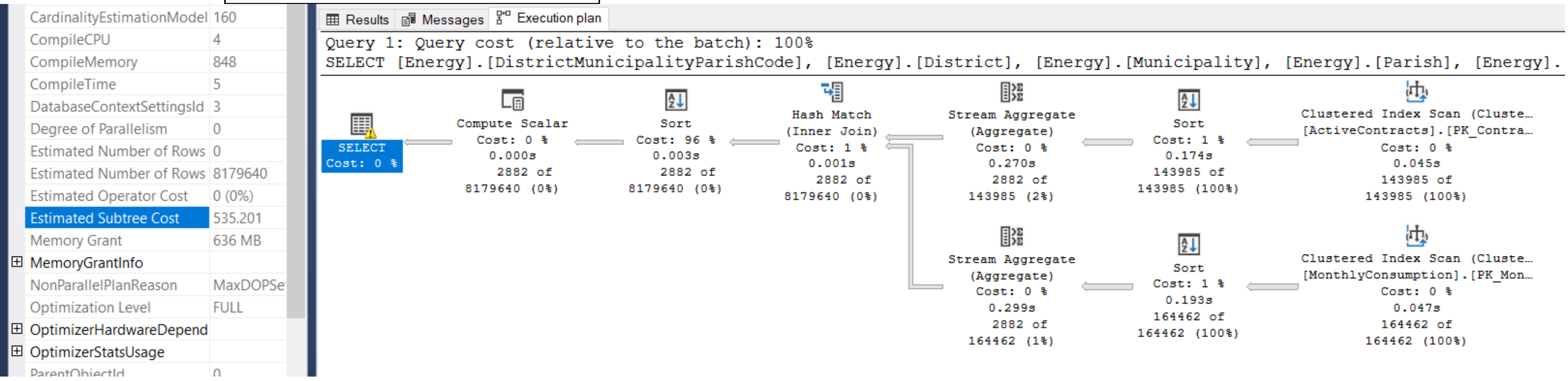


CardinalityEstimationModel	160
CompileCPU	17
CompileMemory	864
CompileTime	17
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	596.062
Memory Grant	636 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality], [Energy].[Parish], [Energy].[Act:
```

```
OPTION (HASH JOIN)
```

535.201

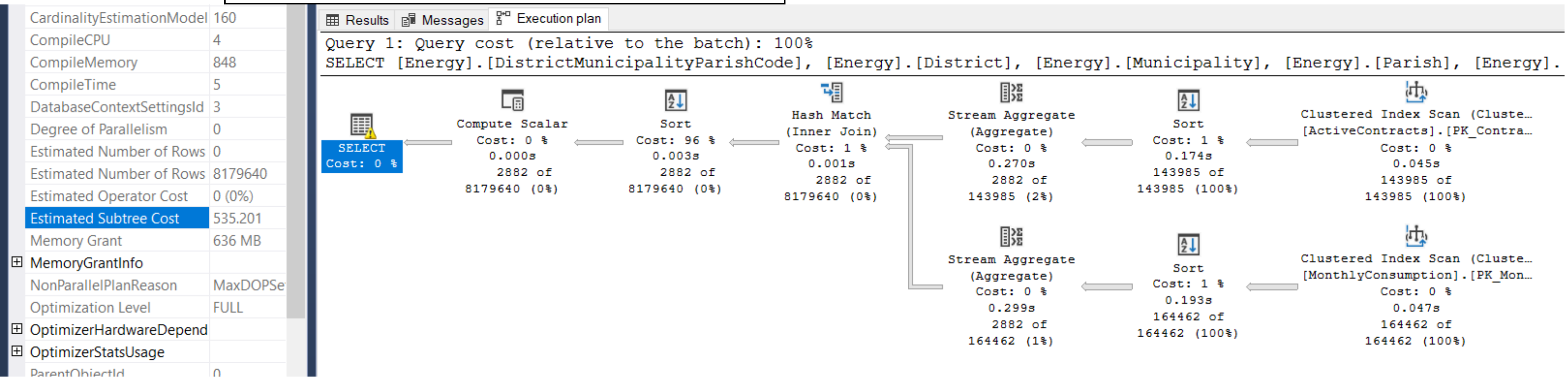


CardinalityEstimationModel	160
CompileCPU	4
CompileMemory	848
CompileTime	5
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	535.201
Memory Grant	636 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality], [Energy].[Parish], [Energy].
```

```
OPTION (STREAM AGGREGATE)
```

535.201



CardinalityEstimationModel	160
CompileCPU	4
CompileMemory	848
CompileTime	5
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	535.201
Memory Grant	636 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

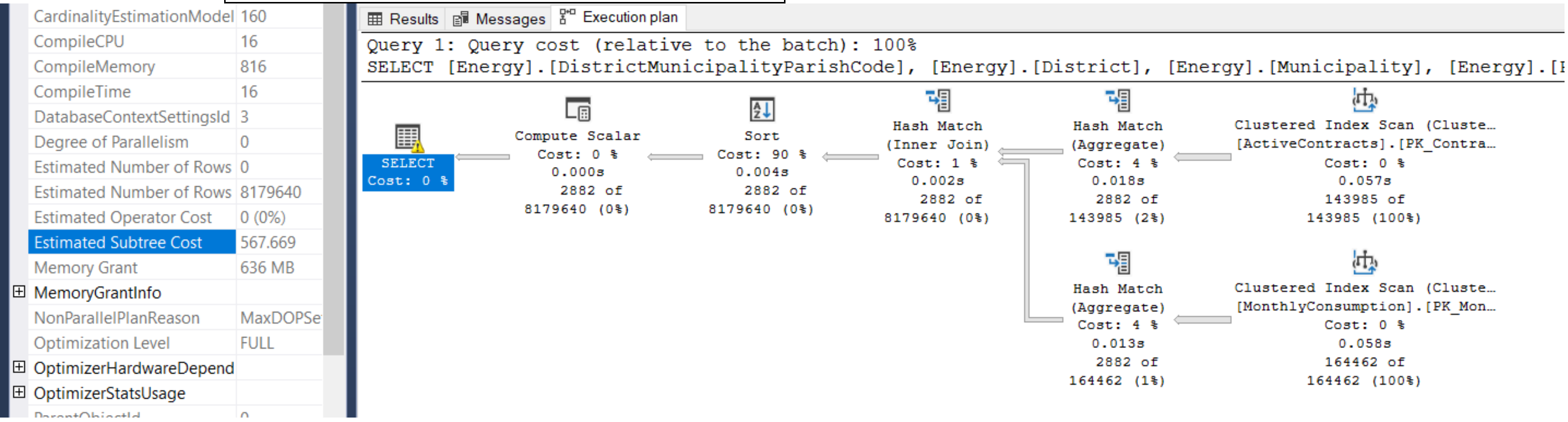
```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality], [Energy].[Parish], [Energy].
```

```
SELECT        Compute Scalar   Sort          Hash Match     Stream Aggregate  Sort           Clustered Index Scan (Cluste…
Cost: 0 %     Cost: 0 %        Cost: 96 %    (Inner Join)   (Aggregate)       Cost: 1 %      [ActiveContracts].[PK_Contra…
              0.000s           0.003s        Cost: 1 %      Cost: 0 %         0.174s         Cost: 0 %
              2882 of          2882 of       0.001s         0.270s            143985 of      0.045s
              8179640 (0%)     8179640 (0%)  2882 of        2882 of           143985 (100%)  143985 of
                                             8179640 (0%)   143985 (2%)                      143985 (100%)

                                                            Stream Aggregate  Sort           Clustered Index Scan (Cluste…
                                                            (Aggregate)       Cost: 1 %      [MonthlyConsumption].[PK_Mon…
                                                            Cost: 0 %         0.193s         Cost: 0 %
                                                            0.299s            164462 of      0.047s
                                                            2882 of           164462 (100%)  164462 of
                                                            164462 (1%)                      164462 (100%)
```

```
OPTION (HASH GROUP)
```

567.669



CardinalityEstimationModel	160
CompileCPU	16
CompileMemory	816
CompileTime	16
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	567.669
Memory Grant	636 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality], [Energy].[I
```

```
SELECT        Compute Scalar   Sort          Hash Match     Hash Match       Clustered Index Scan (Cluste…
Cost: 0 %     Cost: 0 %        Cost: 90 %    (Inner Join)   (Aggregate)      [ActiveContracts].[PK_Contra…
              0.000s           0.004s        Cost: 1 %      Cost: 4 %        Cost: 0 %
              2882 of          2882 of       0.002s         0.018s           0.057s
              8179640 (0%)     8179640 (0%)  2882 of        2882 of          143985 of
                                             8179640 (0%)   143985 (2%)      143985 (100%)

                                                            Hash Match       Clustered Index Scan (Cluste…
                                                            (Aggregate)      [MonthlyConsumption].[PK_Mon…
                                                            Cost: 4 %        Cost: 0 %
                                                            0.013s           0.058s
                                                            2882 of          164462 of
                                                            164462 (1%)      164462 (100%)
```

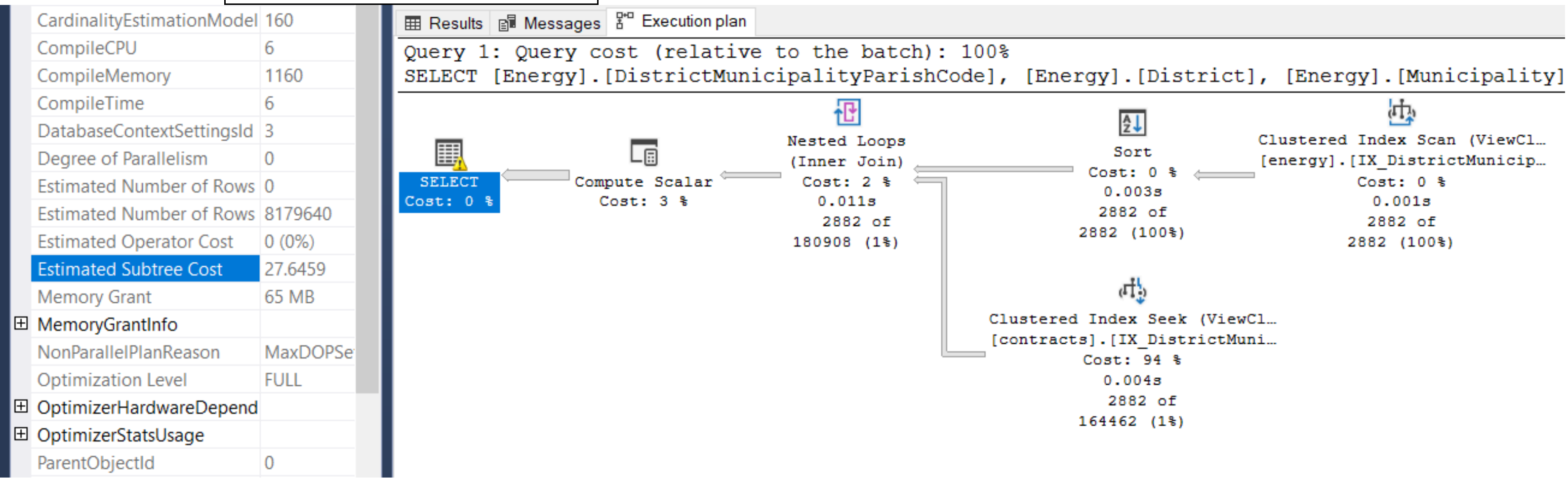
Task 4

```
IF OBJECT_ID ('Energy.energy', 'view') IS NOT NULL
   DROP VIEW Energy.energy;
GO
CREATE VIEW Energy.energy WITH SCHEMABINDING AS
    SELECT [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish],
        SUM([ActiveEnergy]) AS [ActiveEnergy],
        COUNT_BIG(*) AS COUNT
    FROM [Energy].[MonthlyConsumption]
    GROUP BY [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish];
GO
CREATE UNIQUE CLUSTERED INDEX IX_DistrictMunicipalityParishCode
   ON Energy.energy (DistrictMunicipalityParishCode);
GO
```

```
IF OBJECT_ID ('Energy.contracts', 'view') IS NOT NULL
   DROP VIEW Energy.contracts;
GO
CREATE VIEW Energy.contracts WITH SCHEMABINDING AS
    SELECT [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish],
        SUM([NumberContracts]) AS [NumberContracts],
        COUNT_BIG(*) AS COUNT
    FROM [Energy].[ActiveContracts]
    GROUP BY [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish];
GO
CREATE UNIQUE CLUSTERED INDEX IX_DistrictMunicipalityParishCode
   ON Energy.contracts (DistrictMunicipalityParishCode);
GO
```

```
SELECT [Energy].[DistrictMunicipalityParishCode],
    [Energy].[District],
    [Energy].[Municipality],
    [Energy].[Parish],
    [Energy].[ActiveEnergy],
    [Contracts].[NumberContracts],
    [Energy].[ActiveEnergy] / [Contracts].[NumberContracts] AS
EnergyPerContract
FROM Energy.energy as [Energy], Energy.contracts as [Contracts]
WHERE [Energy].[DistrictMunicipalityParishCode] =
    [Contracts].[DistrictMunicipalityParishCode]
ORDER BY [Energy].[District],
    [Energy].[Municipality],
    [Energy].[Parish]
```

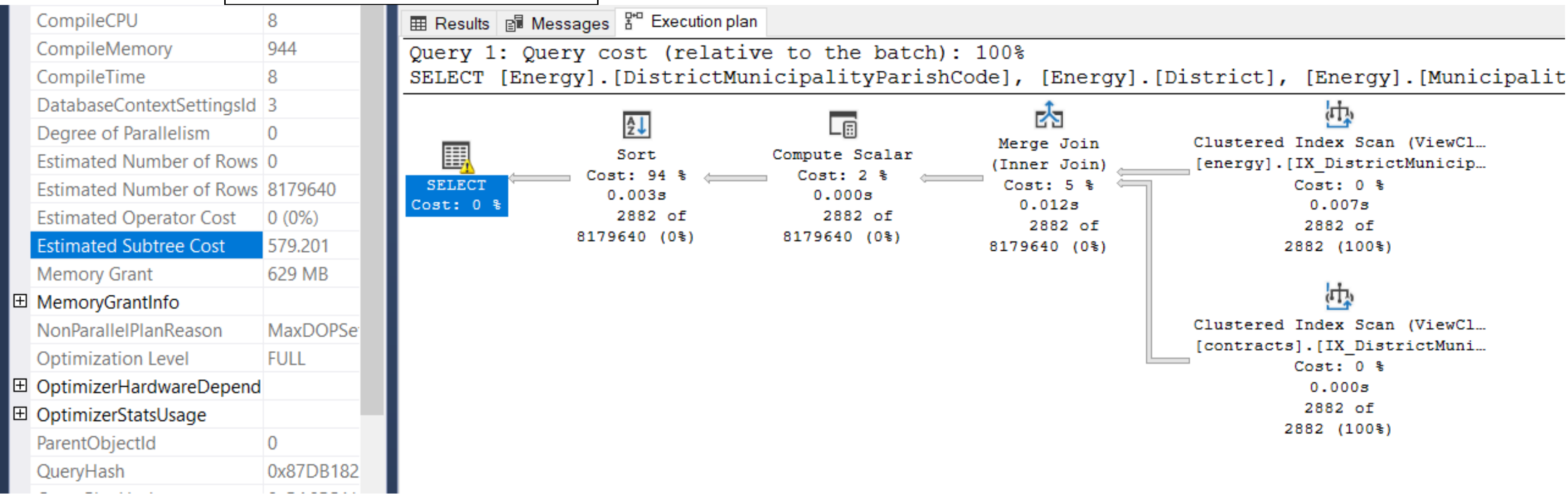
OPTION (LOOP JOIN)
27.6459



CardinalityEstimationModel	160
CompileCPU	6
CompileMemory	1160
CompileTime	6
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	27.6459
Memory Grant	65 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipality]
```

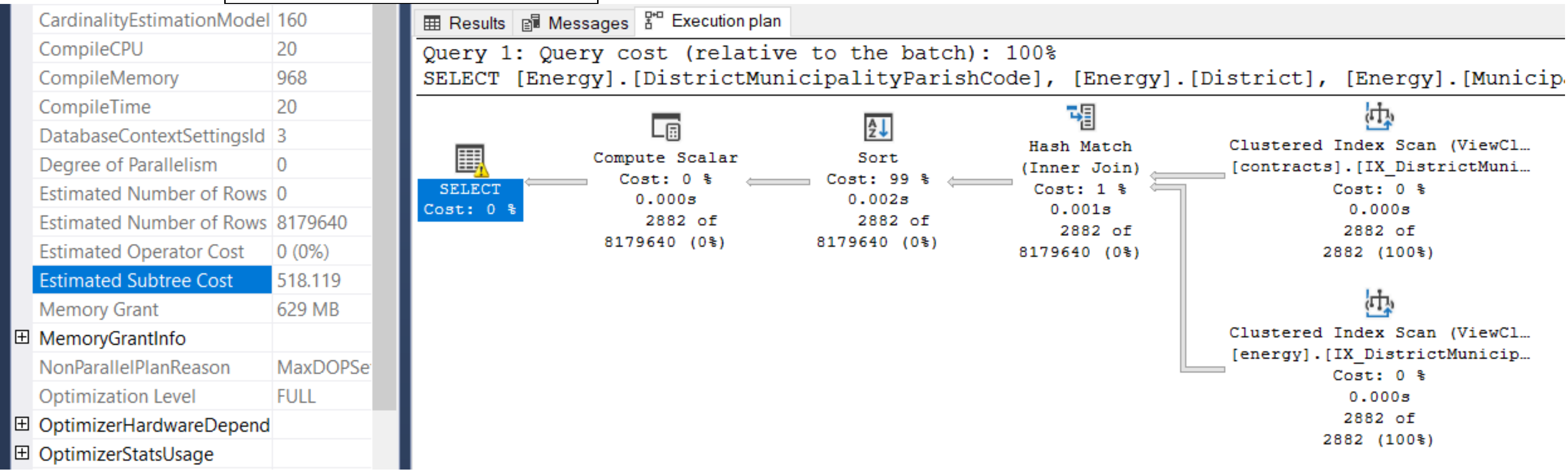
OPTION (MERGE JOIN)
579.201



CompileCPU	8
CompileMemory	944
CompileTime	8
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	579.201
Memory Grant	629 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	
ParentObjectId	0
QueryHash	0x87DB182

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municipalit
```

OPTION (HASH JOIN)
518.119



CardinalityEstimationModel	160
CompileCPU	20
CompileMemory	968
CompileTime	20
DatabaseContextSettingsId	3
Degree of Parallelism	0
Estimated Number of Rows	0
Estimated Number of Rows	8179640
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	518.119
Memory Grant	629 MB
⊞ MemoryGrantInfo	
NonParallelPlanReason	MaxDOPSe
Optimization Level	FULL
⊞ OptimizerHardwareDepend	
⊞ OptimizerStatsUsage	

```
Query 1: Query cost (relative to the batch): 100%
SELECT [Energy].[DistrictMunicipalityParishCode], [Energy].[District], [Energy].[Municip
```

Task 5

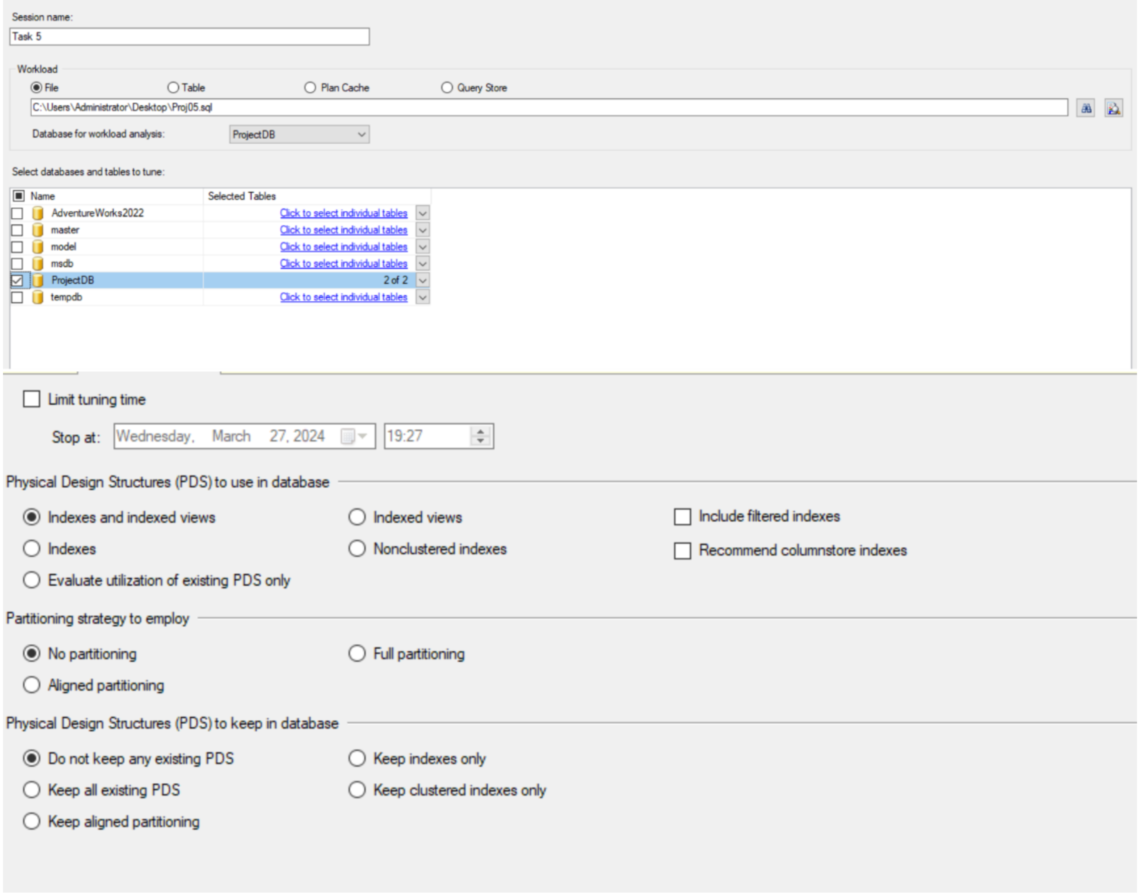
In this task in order to get the best results possible run both queries from task 2 and 3 in the same workload in order to get the best results possible that apply to both.

This is our workload:

```sql
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [ActiveEnergy]
FROM [Energy].[MonthlyConsumption]
WHERE [Municipality] = 'Lisboa'
AND [Month] = '06'
GROUP BY [Parish], [Year]
ORDER BY [Parish], [Year]

SELECT [Energy].[DistrictMunicipalityParishCode],
    [Energy].[District],
    [Energy].[Municipality],
    [Energy].[Parish],
    [Energy].[ActiveEnergy],
    [Contracts].[NumberContracts],
    [Energy].[ActiveEnergy] / [Contracts].[NumberContracts] AS
EnergyPerContract
FROM (
    SELECT [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish],
        SUM([ActiveEnergy]) AS [ActiveEnergy]
    FROM [Energy].[MonthlyConsumption]
    GROUP BY [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish]
) AS [Energy], (
    SELECT [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish],
    SUM([NumberContracts]) AS [NumberContracts]
    FROM [Energy].[ActiveContracts]
    GROUP BY [DistrictMunicipalityParishCode],
        [District],
        [Municipality],
        [Parish]
) AS [Contracts]
WHERE [Energy].[DistrictMunicipalityParishCode] =
    [Contracts].[DistrictMunicipalityParishCode]
ORDER BY [Energy].[District],
    [Energy].[Municipality],
    [Energy].[Parish]
```

The workload will be run on Database Engine Tuning Advisor with the following settings:



Session name:
Task 5

Workload
● File ○ Table ○ Plan Cache ○ Query Store
C:\Users\Administrator\Desktop\Proj05.sql

Database for workload analysis: ProjectDB

Select databases and tables to tune:

☐	Name	Selected Tables
☐	AdventureWorks2022	Click to select individual tables
☐	master	Click to select individual tables
☐	model	Click to select individual tables
☐	msdb	Click to select individual tables
☑	ProjectDB	2 of 2
☐	tempdb	Click to select individual tables

☐ Limit tuning time

Stop at: Wednesday, March 27, 2024 19:27

Physical Design Structures (PDS) to use in database

● Indexes and indexed views ○ Indexed views ☐ Include filtered indexes
○ Indexes ○ Nonclustered indexes ☐ Recommend columnstore indexes
○ Evaluate utilization of existing PDS only

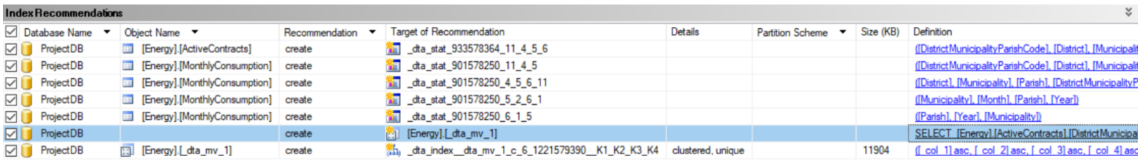
Partitioning strategy to employ

● No partitioning ○ Full partitioning
○ Aligned partitioning

Physical Design Structures (PDS) to keep in database

● Do not keep any existing PDS ○ Keep indexes only
○ Keep all existing PDS ○ Keep clustered indexes only
○ Keep aligned partitioning

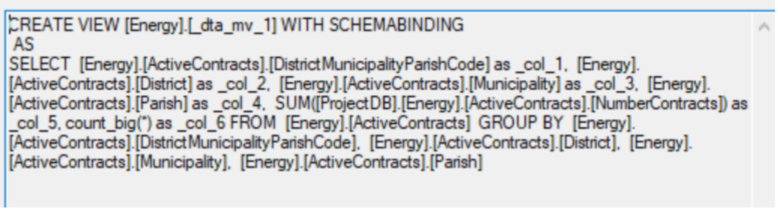
After running the analysis with these settings we get the following recommendations:



Index Recommendations

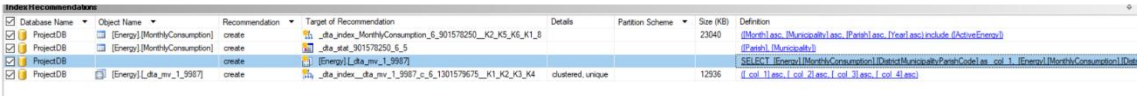
☑	Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
☑	ProjectDB	[Energy].[ActiveContracts]	create	_dta_stat_933578364_11_4_5_6				([DistrictMunicipalityParishCode], [District], [Municipalit
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_stat_901578250_11_4_5				([DistrictMunicipalityParishCode], [District], [Municipalit
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_stat_901578250_4_5_6_11				([District], [Municipality], [Parish], [DistrictMunicipalityP
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_stat_901578250_5_2_6_1				([Municipality], [Month], [Parish], [Year])
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_stat_901578250_6_1_5				([Parish], [Year], [Municipality])
☑	ProjectDB		create	[Energy].[_dta_mv_1]				SELECT [Energy].[ActiveContracts].[DistrictMunicipa
☑	ProjectDB	[Energy].[_dta_mv_1]	create	_dta_index__dta_mv_1_c_6_1221579390__K1_K2_K3_K4	clustered, unique		11904	([_col_1] asc, [_col_2] asc, [_col_3] asc, [_col_4] asc

As we can notice the second to last recommendation looks familiar and that is because that is the second materialized view we created in task 4



```
CREATE VIEW [Energy].[_dta_mv_1] WITH SCHEMABINDING
 AS
SELECT  [Energy].[ActiveContracts].[DistrictMunicipalityParishCode] as _col_1,  [Energy].
[ActiveContracts].[District] as _col_2,  [Energy].[ActiveContracts].[Municipality] as _col_3,  [Energy].
[ActiveContracts].[Parish] as _col_4,  SUM([ProjectDB].[Energy].[ActiveContracts].[NumberContracts]) as
_col_5, count_big(*) as _col_6 FROM  [Energy].[ActiveContracts]  GROUP BY  [Energy].
[ActiveContracts].[DistrictMunicipalityParishCode],  [Energy].[ActiveContracts].[District],  [Energy].
[ActiveContracts].[Municipality],  [Energy].[ActiveContracts].[Parish]
```

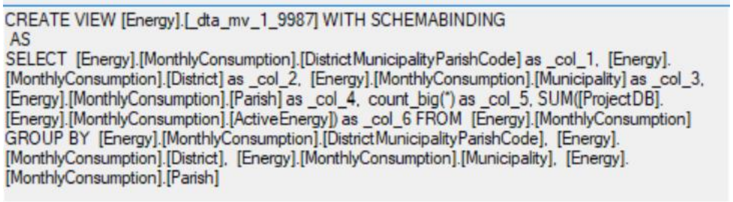
By applying these recommendations and running a new analysis (with the same settings) we will get new recommendations



Index Recommendations

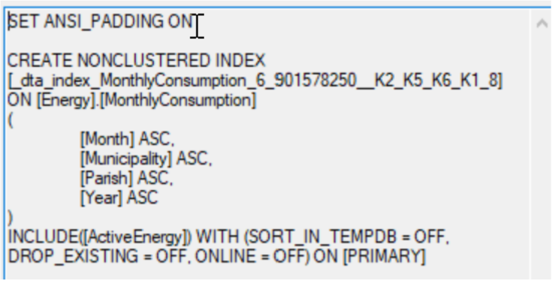
☑	Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_index_MonthlyConsumption_6_901578250__K2_K5_K6_K1_8			23040	([Month] asc, [Municipality] asc, [Parish] asc, [Year] asc) include ([ActiveEnergy])
☑	ProjectDB	[Energy].[MonthlyConsumption]	create	_dta_stat_901578250_6_5				([Parish], [Municipality])
☑	ProjectDB		create	[Energy].[_dta_mv_1_9987]				SELECT [Energy].[MonthlyConsumption].[DistrictMunicipalityParishCode] as _col_1, [Energy].[MonthlyConsumption].[Dist
☑	ProjectDB	[Energy].[_dta_mv_1_9987]	create	_dta_index__dta_mv_1_9987_c_6_1301579675__K1_K2_K3_K4	clustered, unique		12936	([_col_1] asc, [_col_2] asc, [_col_3] asc, [_col_4] asc)

Again, as we can notice the third recommendation is the other materialized view we created in task 4



```
CREATE VIEW [Energy].[_dta_mv_1_9987] WITH SCHEMABINDING
 AS
SELECT  [Energy].[MonthlyConsumption].[DistrictMunicipalityParishCode] as _col_1,  [Energy].
[MonthlyConsumption].[District] as _col_2,  [Energy].[MonthlyConsumption].[Municipality] as _col_3,
[Energy].[MonthlyConsumption].[Parish] as _col_4,  count_big(*) as _col_5, SUM([ProjectDB].
[Energy].[MonthlyConsumption].[ActiveEnergy]) as _col_6 FROM  [Energy].[MonthlyConsumption]
GROUP BY  [Energy].[MonthlyConsumption].[DistrictMunicipalityParishCode],  [Energy].
[MonthlyConsumption].[District],  [Energy].[MonthlyConsumption].[Municipality],  [Energy].
[MonthlyConsumption].[Parish]
```

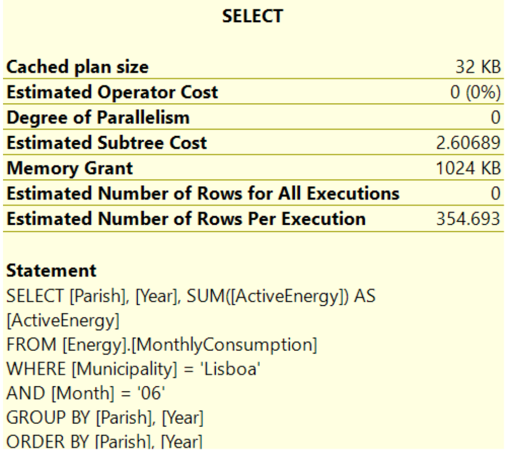
Also, the first recommendation is a new index that we can apply to the query on task 2 to get improved performance.



```
SET ANSI_PADDING ON

CREATE NONCLUSTERED INDEX
[_dta_index_MonthlyConsumption_6_901578250__K2_K5_K6_K1_8]
ON [Energy].[MonthlyConsumption]
(
            [Month] ASC,
            [Municipality] ASC,
            [Parish] ASC,
            [Year] ASC
)
INCLUDE([ActiveEnergy]) WITH (SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF) ON [PRIMARY]
```

After creating this index we can compare the estimated subtree cost before and after

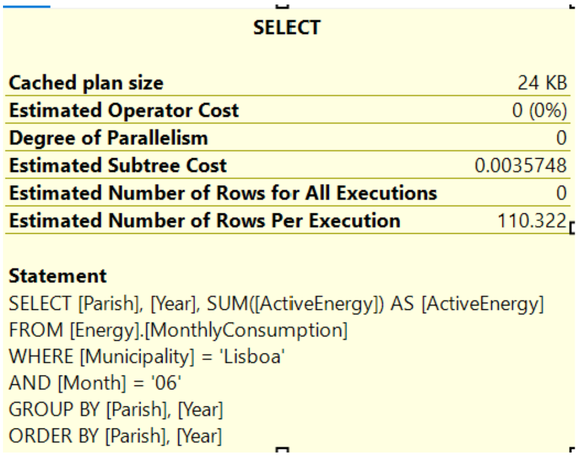


SELECT

Cached plan size	32 KB
Estimated Operator Cost	0 (0%)
Degree of Parallelism	0
Estimated Subtree Cost	2.60689
Memory Grant	1024 KB
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	354.693

Statement

```
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS
[ActiveEnergy]
FROM [Energy].[MonthlyConsumption]
WHERE [Municipality] = 'Lisboa'
AND [Month] = '06'
GROUP BY [Parish], [Year]
ORDER BY [Parish], [Year]
```



SELECT

Cached plan size	24 KB
Estimated Operator Cost	0 (0%)
Degree of Parallelism	0
Estimated Subtree Cost	0.0035748
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	110.322

Statement

```
SELECT [Parish], [Year], SUM([ActiveEnergy]) AS [ActiveEnergy]
FROM [Energy].[MonthlyConsumption]
WHERE [Municipality] = 'Lisboa'
AND [Month] = '06'
GROUP BY [Parish], [Year]
ORDER BY [Parish], [Year]
```