**SE 308 Term Project 2**

**Onur Akalın 170706018 & Ahmet Arif Özçelik 170706016**

**Introduction**

We worked 2 people in this performance experiment, so we got different results from different

computers. In this way, we think our results are more accurate. We developed our program using

.Net Core, ADO.Net and MSSQL.

**Project**

We took our before query measurements in 2 different computer and We noticed that these queries run at different performance speeds on different computers and that the gap between them is huge, nearly 10 seconds. We are using different OS and system environments so I think, It is acceptable results for us.

**Average Times Before Indexıng (Second)**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 1** | 14.768 s | 21.383 s |
| **Query 2** | 9.796 s | 16.108 s |
| **Query 3** | 9.450 s | 14.563 s |

**Total Times Before Indexıng (Minute)**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 1** | 24.61 m | 35.63 m |
| **Query 2** | 16.32 m | 26.84 m |
| **Query 3** | 15.75 m | 24.27 m |

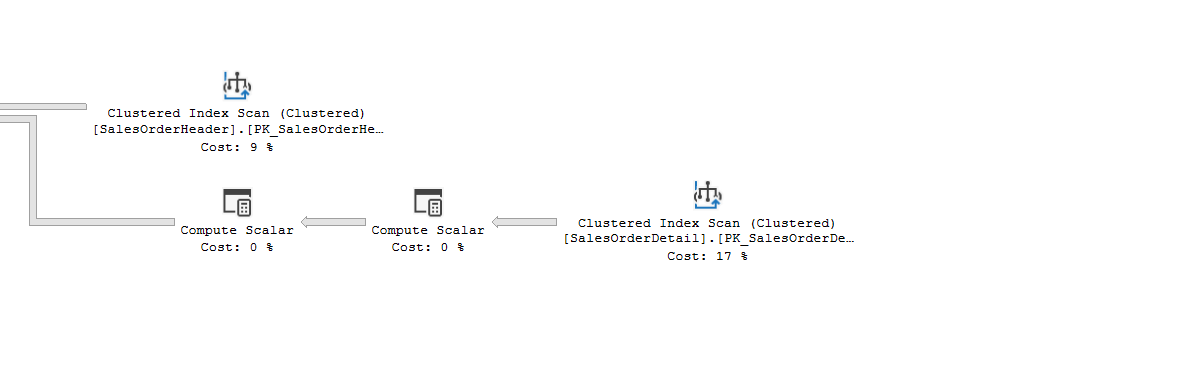
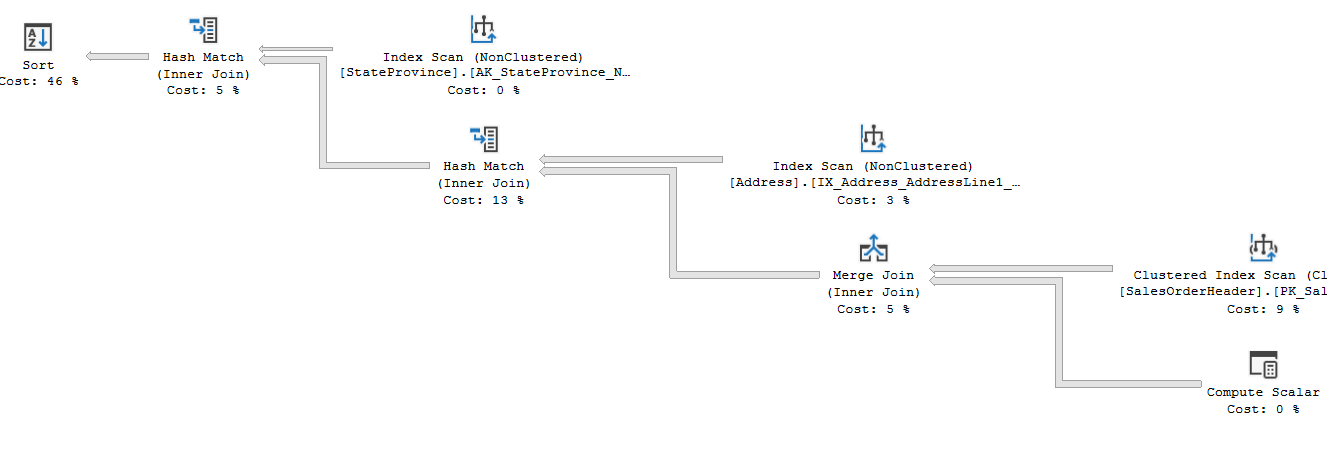
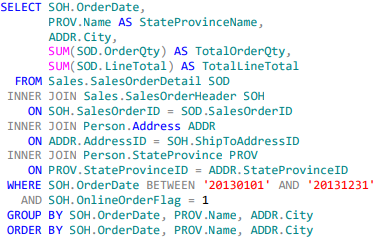
In our project, We run these queries totally 10.000 times, We defined 1 unit as 100 queries and execute totally 100 unit on each different query.

**Index**

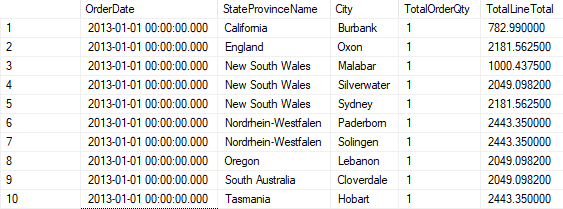
We are aimed to reduce cost of Indexes and transform them to Seeks to speed up executions but sometimes even if We reduce the costs, speed of execution would be decreased drastically so that We gave up some index solutions.

While creating indexes for each query, we deleted the indexes we created before. Since the indexes are similar to each other, we have prevented the undesired use of indexes in this way.

**Query-1:**



**TOTAL RETURNED ROWS IS 10899**



1. Firstly, We want to create index for SalesOrderHeader table because It’s affect nearly all parts of query.

* **SOH.OrderDate :** This column was in Select , Where , Group By and Order By parts so that it is important for reduce the costs.
* **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be first column in index.
* **SOH.ShipToAddressID :** This column was in right sides of Nested Inner Join so that It’s not important as before 2 columns and It will be in include part.
* **SOH.SalesOrderID:** We don’t include this column to the index because It’s primary key and It’s have already have Clustered Index.

**CREATE INDEX Q1IndexOnlineOrderFlagANDOrderDateWithShipToAddressId**

**ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)**

**INCLUDE (ShipToAddressID)**

**\*OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

1. We want to created index for SalesOrderDetail table because it’s affect on one Nested Inner Join and Select part

* **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It’s not enough. We want to create a new Non-Clustered index with some includes.
* **SOD.OrderQty:** OrderQTY is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.
* **SOD.LineTotal:** LineTotal is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.

**CREATE INDEX Q1ProductIdWithOrderQtyAndLineTotal**

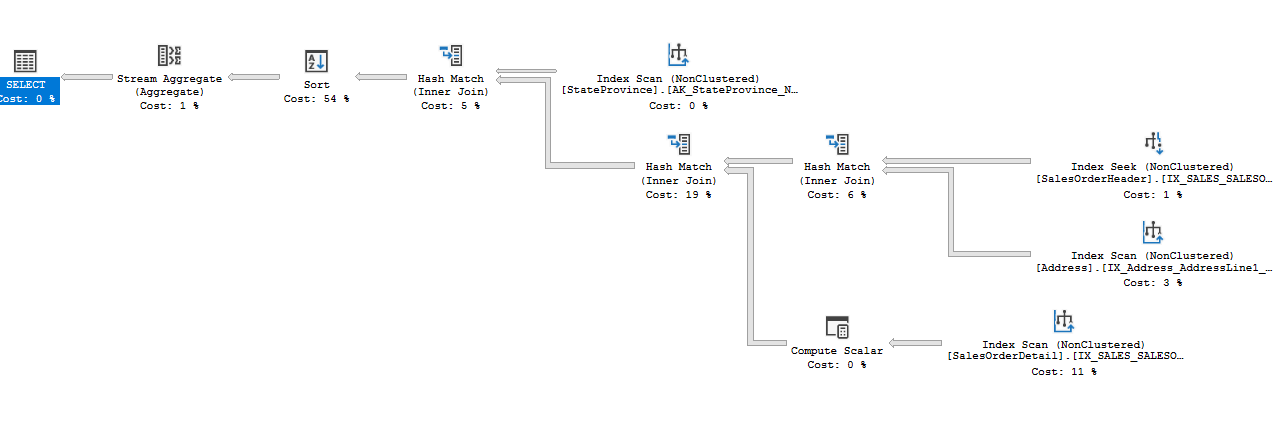
**ON Sales.SalesOrderDetail (ProductID)**

**INCLUDE (OrderQty, LineTotal)**

**\*If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

1. We don’t add any indexes for **PROV.Name** and **ADDR.City** because ADDR.City is become 0% cost after some indexes are added and PROV.Name has very low cost and has no affect on query with premade indexes for it.

**After Indexing**



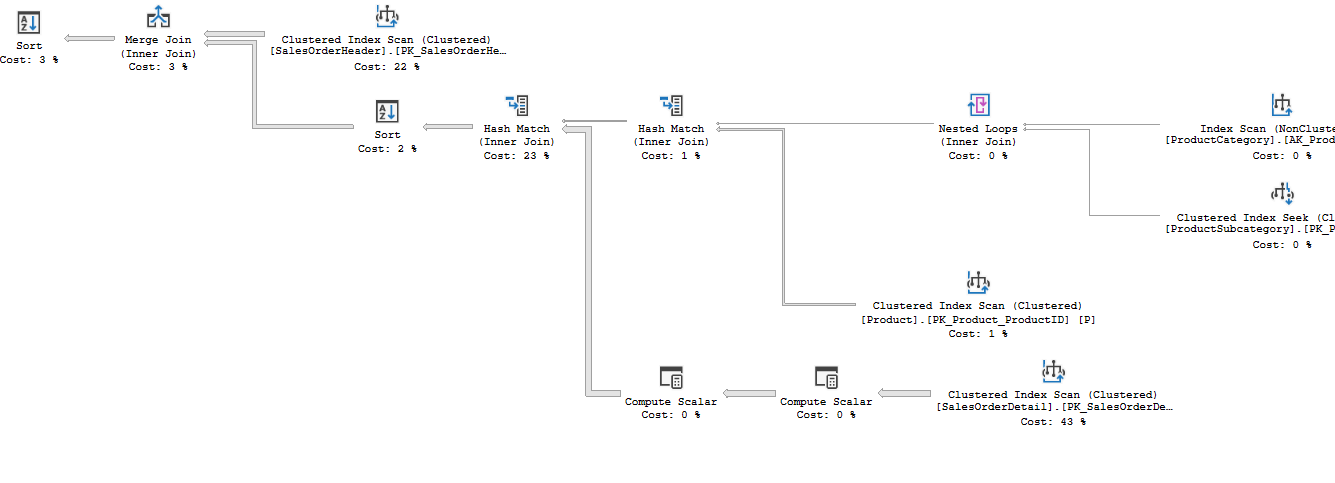
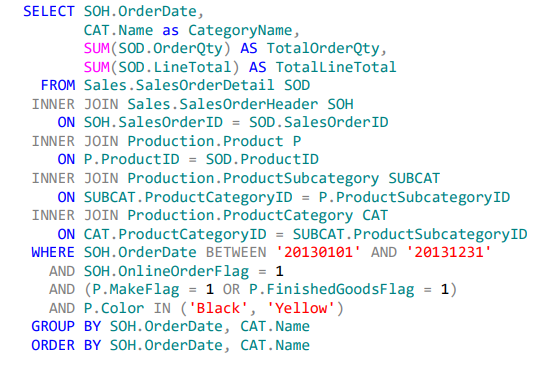
**Average times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 1** | 13.49 s | 19.98 s |
| **Remain** | +1.278 s | +1.403 s |

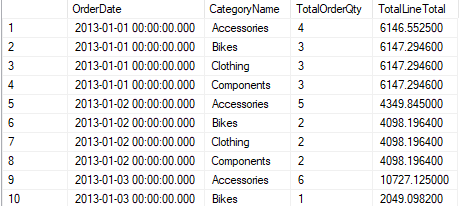
**Total times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 1** | 22.48 m | 33.30 m |
| **Remain** | +2.13 m | +2.33 m |

**Query-2 (Failure):**



**TOTAL RETURNED ROWS IS 1360**



1. Firstly, We want to create index for SalesOrderHeader table because It’s affect nearly all parts of query.

* **SOH.OrderDate :** This column was in Select , Where , Group By and Order By parts so that it is important for reduce the costs and increase speed.
* **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be first column in index, it’s important for index to work fast and properly.
* **SOH.ShipToAddressID :** This column was in right sides of Nested Inner Join so that It’s not important as before 2 columns and It will be in include part.
* **SOH.SalesOrderID:** We don’t include this column to the index because It’s primary key and It’s have already have Clustered Index.

**CREATE INDEX Q2IndexOnlineOrderFlagANDOrderDateWithShipToAddressId**

**ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)**

**INCLUDE (ShipToAddressID)**

**\*OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

1. We want to create index for SalesOrderDetail table because it’s affect on one Nested Inner Join and Select part

* **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It’s not enough. We want to create a new Non-Clustered index with some includes to make faster.
* **SOD.OrderQty:** OrderQTY is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.
* **SOD.LineTotal:** LineTotal is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.

**CREATE INDEX Q2ProductIdWithOrderQtyAndLineTotal**

**ON Sales.SalesOrderDetail (ProductID)**

**INCLUDE (OrderQty, LineTotal)**

**\*If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

1. We want to create index for Product table because it’s affect on one Nested Inner Join and Select part.

* **P.MakeFlag:** MakeFlag column was in Where and it is use for equation so that this part will be first column in index.
* **P.FinishedGoodsFlag:** FinishedGoodsFlag column was in Where and it is use for equation so that this part will be second column in index.
* **P.Color :** Color column was in Where and it is use for equation so that this part will be third column in index.
* **P.** **ProductSubcategoryID:** ProductSubcategoryID column was in right sides of Nested Inner Join so that It’s not important as before 3 columns and It will be in include part.

**CREATE INDEX Q2MakeFlagAndFinishedGoodsFlagAndColorWithProductSubcategoryID**

**ON [Production].[Product] (MakeFlag DESC , FinishedGoodsFlag DESC, Color DESC )**

**INCLUDE (ProductSubcategoryID)**

**\*This index will be reduce the sorting load of the process and execute the data in the table more faster with sorted data.**

1. We don’t create index for **CAT.NAME** because It’s have 0% cost and has no affect on query with premade indexes for it.

**Average times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 2** | 11.036 s | 18.253 s |
| **Remain** | -1.240 s | -2.145 s |

**Total times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 2** | 18.39 m | 30.42 m |
| **Remain** | -2.07 m | -3.58 m |

(When we add Index for 2 main columns ( SalesOrderDetail (43% Cost) and SalesOrderHeader (22% Cost) ), It reduce the speed drastically even their costs are decreased. We drop or change indexes to reduce the costs and improve speed but It didn’t work. )

\*We improved speed only with 3th index on this document.

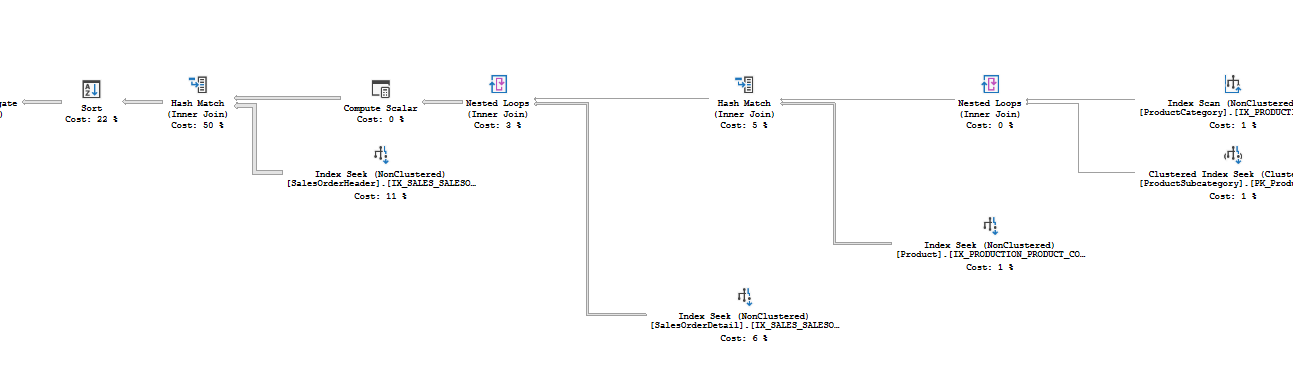
**Average times** **after Indexes (only with 3th index)**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 2** | 8.128 s | 14.126 s |
| **Remain** | +1.668 s | +1.982 s |

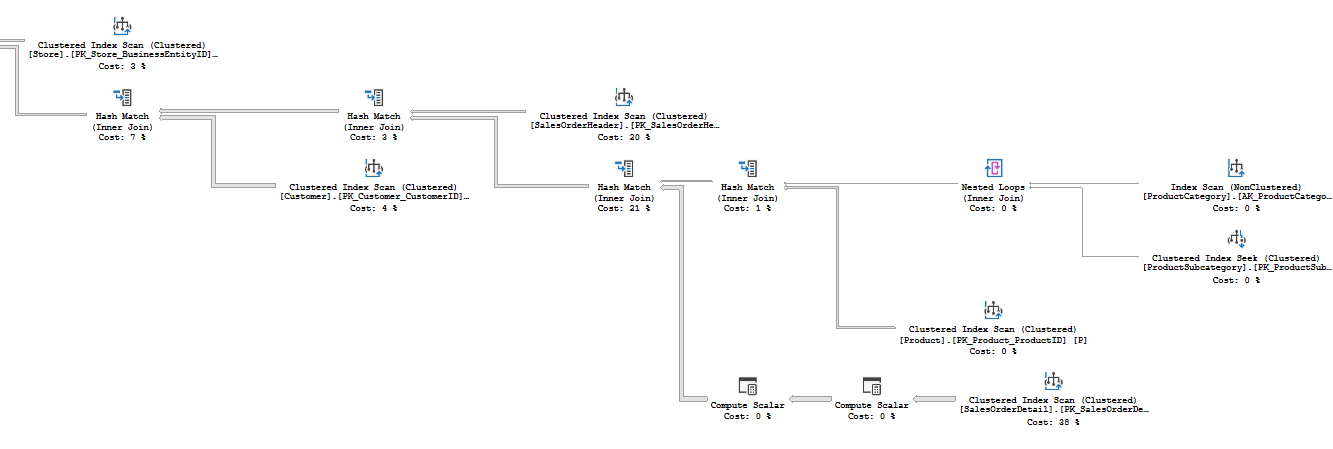
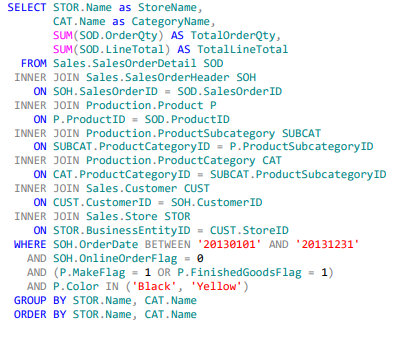
**Total times** **after Indexes (only with 3th index)**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 2** | 13.54 m | 23.54 m |
| **Remain** | +3.18 m | +3.30 m |

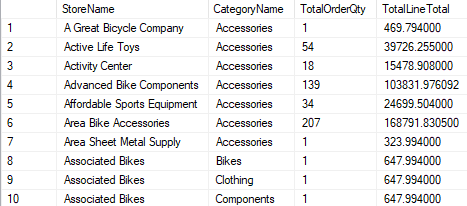
**After Indexing**



**Query-3:**



**TOTAL RETURNED ROWS IS 656**



1. Firstly, We want to create index for SalesOrderHeader table because It’s affect on where and inner join parts.

* **SOH.OrderDate :** This column was in Where and it is use for equation so that this part will be first column in index.
* **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be second column in index.
* **SOH.CustomerID :** This column was in right sides of Nested Inner Join so that It’s not important as before 2 columns and It will be in include part.
* **SOH.SalesOrderID:** We don’t include this column to the index because It’s primary key and It’s have already have Clustered Index.

**CREATE INDEX Q3IndexOnlineOrderFlagANDOrderDateWithShipToAddressId**

**ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)**

**INCLUDE (CustomerID)**

**\*OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

1. We want to create index for SalesOrderDetail table because it’s affect on one Nested Inner Join and Select part

* **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It’s not enough. We want to create a new Non-Clustered index with some includes to make faster.
* **SOD.OrderQty:** OrderQTY is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.
* **SOD.LineTotal:** LineTotal is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.

**CREATE INDEX Q3ProductIdWithOrderQtyAndLineTotal**

**ON Sales.SalesOrderDetail (ProductID)**

**INCLUDE (OrderQty, LineTotal)**

**\*If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

1. We want to create index for Store table because it’s affect on Select part.

* **STOR.BussinesEntityID:** is primary key so that It has index but in this query, It’s not enough. We want to create a new Non-Clustered index with some includes to make faster.
* **STOR.Name:** Name is used in Select part so that It’s not important as SalesOrderID, It will be placed in Include part.

**CREATE INDEX Q3IndexStoreName**

**ON Sales.Store (BusinessEntityID)**

**INCLUDE (Name)**

**\*This index will be reduce the sorting load of the process and execute the data in the table more faster with sorted data.**

**Average times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 3** | 7.024 s | 11.473 s |
| **Remain** | +2.226 s | +3.090 s |

**Total times** **after Indexes**

|  |  |  |
| --- | --- | --- |
|  | **Onur Akalın** | **Ahmet Arif Özçelik** |
| **Query 3** | 11.70 m | 19.12 m |
| **Remain** | +4.05 m | +5.15 m |

