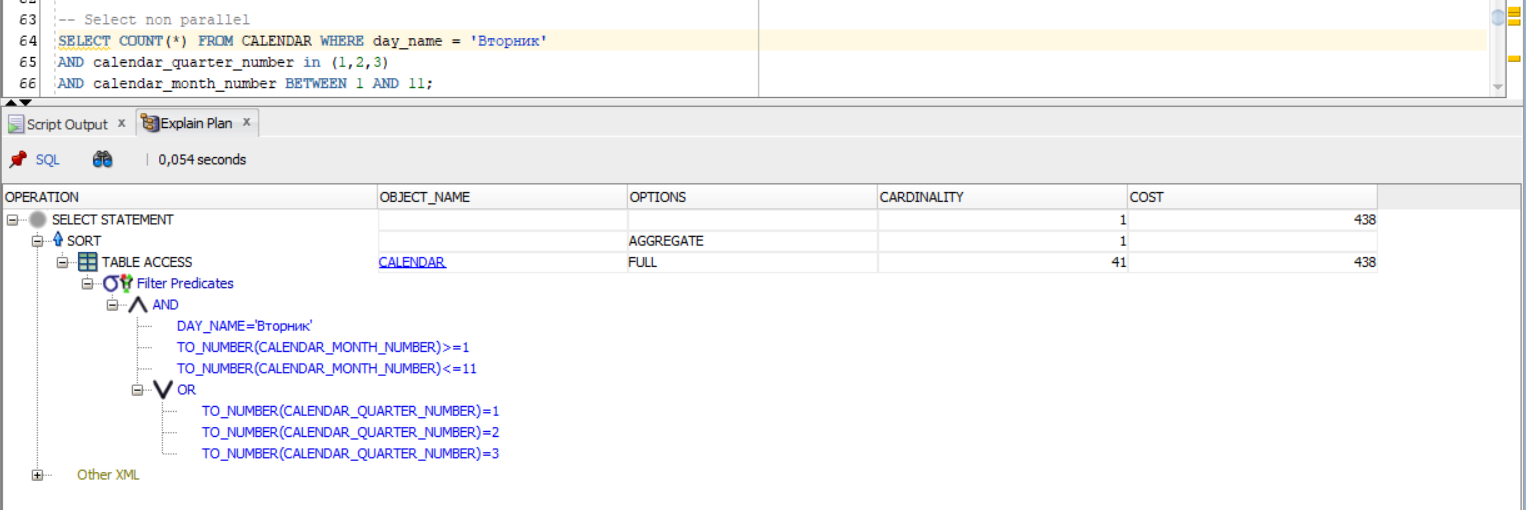
**Anton Slizh’s**

**U1M10.LW.Basic Parallel Execution**

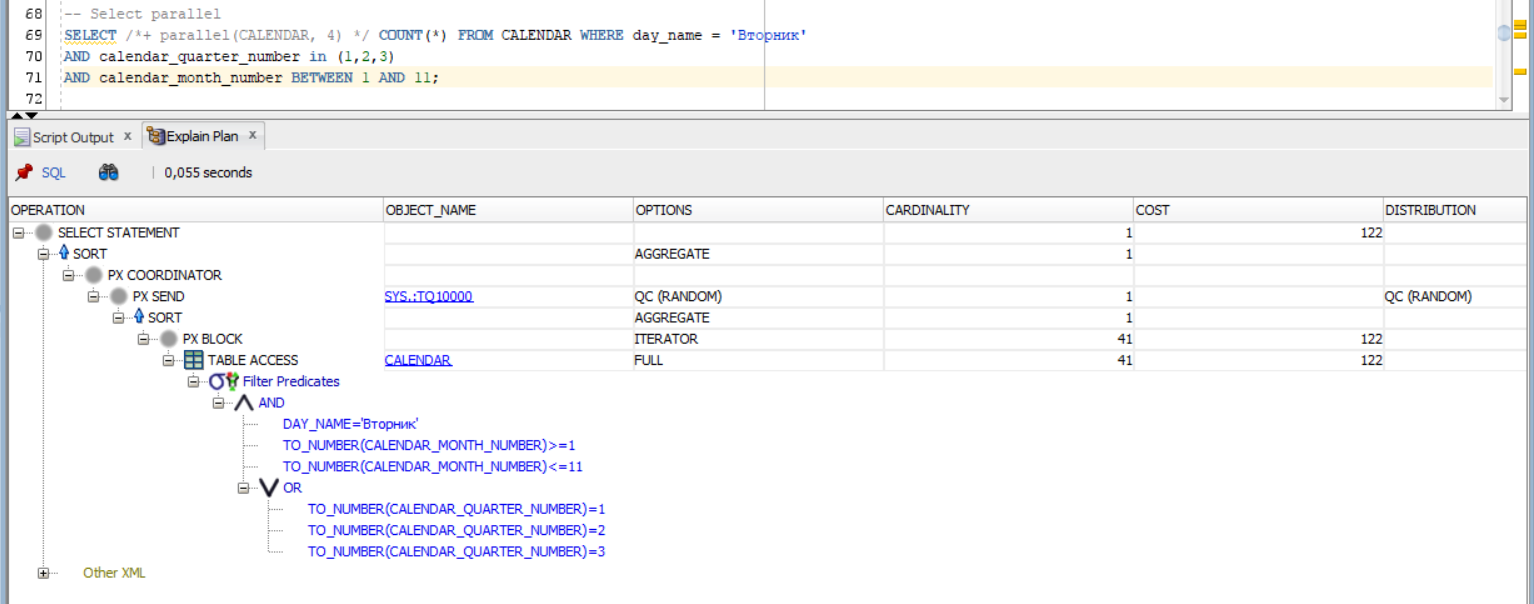
*GitHub: https://github.com/drapejny/DataCamp2022*

## 2.1. Task 01: CREATE Example of Select Parallel execution

Non parallel SELECT – 54ms

****

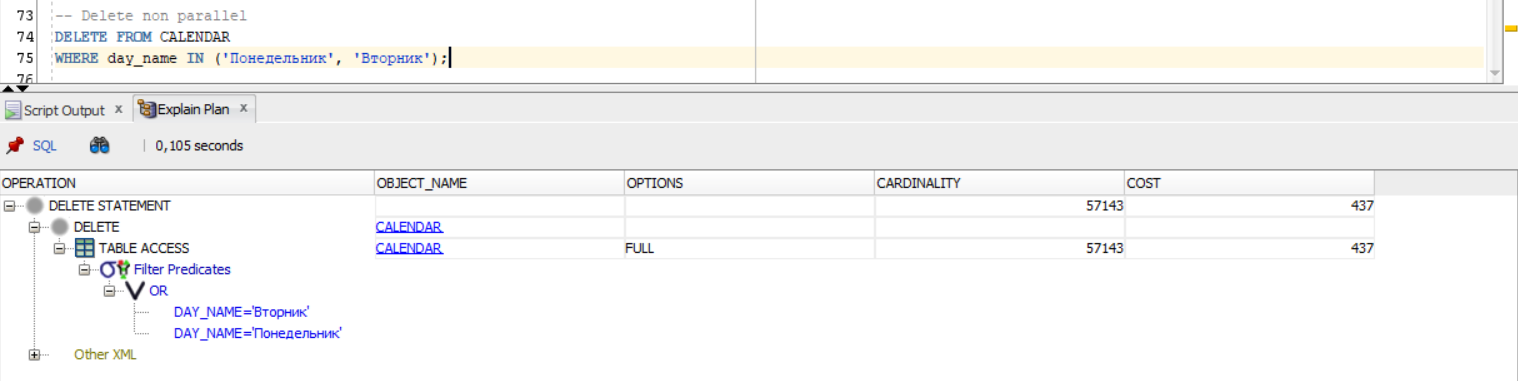
Parallel SELECT – 55ms

****

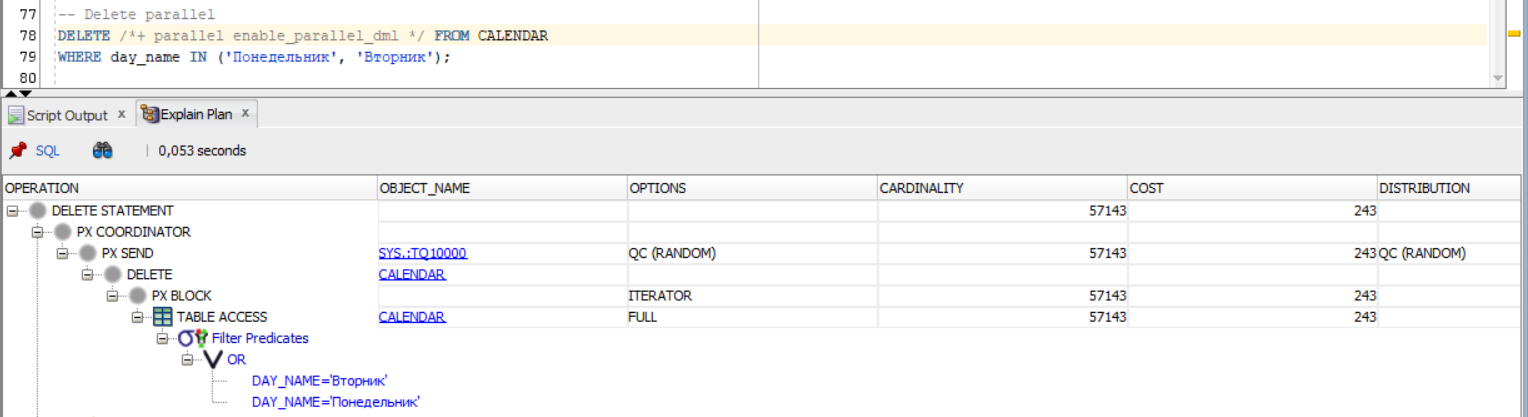
As we see, in our example there is no performance increase on SELECT statements.

## 2.2. Task 02: CREATE Example of Parallel DML

Non parallel DELETE – 105ms

****

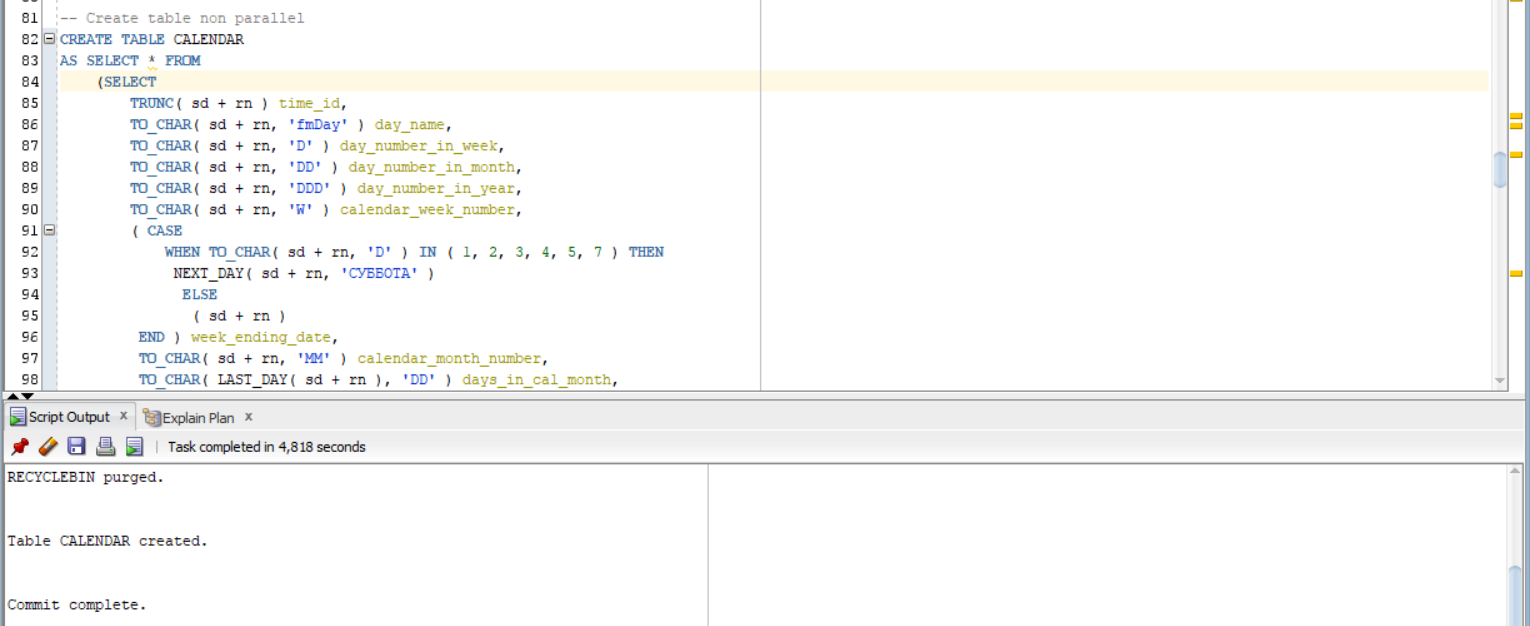
Parallel DELETE – 53ms



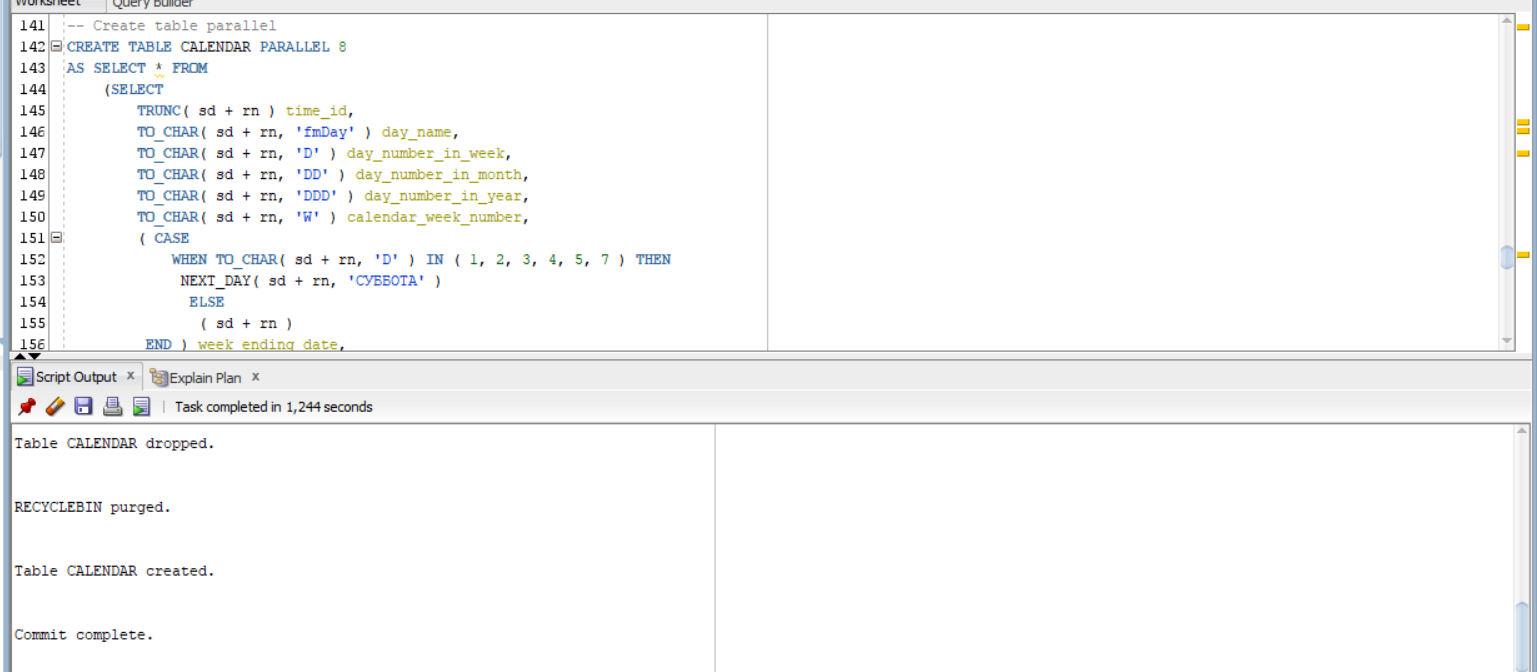
As we see, there is a slight increase in performance.

## 2.3. Task 03: CREATE Example of Parallel DDL

Non parallel CREATE TABLE – 4,818 s



Parallel CREATE TABLE – 1,244



As we see, the parallelization significantly increase performance on DDL operations.

|  |  |  |  |
| --- | --- | --- | --- |
|  | SELECT | DML | DDL |
| Non parallel | 54ms | 105ms | 4,818s |
| Parallel | 55ms | 53ms | 1,244s |

## 3.1. Task 03: CREATE Strategy of Parallel execution

Parallel query:

I’m completely sure, that in developing DWH can be used the parallel execution to optimize large analytical queries. The queries for large tables like FCT\_SALES can be performed using many processes to increase performance and reduce the execution time.

Parallel DML:

There are large tables in developing DWH which must be periodically updated. Using parallelization of DML operations we can increase performance while updating large tables with sales, customers, products data.

Parallel DDL:

According to results of first task, we can make conclusion that parallelization on DDL operations has significant impact on execution time. I’m sure that involving this functionality in developing DWH system will allow us to perform managing tasks faster. Also, it significantly reduces the time spend to recovery in case of any failure. The high availability is one of the main customer requirements.