SQL IMPLEMENTATION

The logic behind the sql implementation is identical to the logic of the python implementation.

In the sql_implementation folder one can find files with scripts that create tables , views to setup the data according to the three hypotheses discussed along with insert statements to fill the tables. Sql statements that solve the three business questions are also provided.

The data are entered in the tb_stg_rawdata table.

A new stage table is introduced to normalize the data according to the three hypotheses.

stg_rawdata

And

tb_STG_rawdata_SQL_ordered_reduced.

stg_rawdata stores the raw dataset . After the data are injected in the table the first and last START and STOP statuses are updated. The script can be found in update_start_stop.sql. Only the rows with status START and STOP are ordered and injected to the table tb_STG_SQL_ordered_reduced. Create and insert statements can be found in the create_tb_STG_rawdata_SQL_ordered_reduced.sql and fill_tb_STG_rawdata_SQL_ordered_reduced.sql files.

As a next step the model table is filled exactly as discussed via view_start_stop_data.

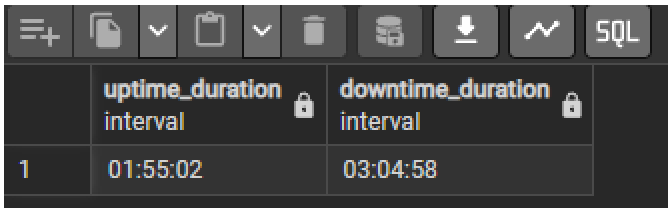
To achieve this data from tb_STG_rawdata_SQL_ordered_reduced is partitioned by production_line_id and ordered by timestamp. The result is self joined on the next row_number and start and stop combination. That way start time and end time end up in the same tuple. At last duration is calculated and the result is injected in the tb_mdl_linedata table. Create_view_start_stop_data.sql contains the sql statement to create the view, fill_tb_mdl_linedata.sql contains the insert statements and Create_mdl_linedata.sql contains the create statement for the table.

tb_mdl_linedata is the base on which every bussiness question is answered.

Bq1: Create_view_line_47.sql contains the create statement for the view that answers question 1. It is a simple filter statement.

Bq2: bq2_with_sp.sql contains a stored procedure chain to answer the second business question. The sp chain follows the logic discussed, with score – event strategy. As a last step is contains a select statement to calculate the total uptime and downtime for the tuples with 4 production_lines running.

The results produced were the same as the python implementation :



	uptime_duration interval	downtime_duration interval
1	01:55:02	03:04:58

Bq3: is answered the view in the Create_view_most_downtime.sql

Nikos Nteits