

AS321478

Open-Source License Dashboard and User's Real-Time Data

Fabio Roberti – Head of BIM
WilkinsonEyre

Alexandros Bertzouanis – BIM Coordinator
WilkinsonEyre

Learning Objectives

- Learn how to prepare the open-source database to receive data from FlexLM License Manager
- Learn how to identify how many licenses a single person is using after opening multiple applications
- Learn how to track the license usage per user to identify consumption and demand
- Learn how to create a license dashboard with an open-source database

Description

This class will analyze data from the FlexLM License Manager to create a live license dashboard using an open-source set of tools. We'll show how by using the Power BI (Business Intelligence) platform and MySQL database we can identify the demand and consumption of licenses per application, group of applications, and group of users per design team. The interactive dashboard will provide valuable statistics of the license usage per software, which helps to predict the future license need.

Speakers:



Fabio Roberti is Head of BIM at WilkinsonEyre, where he leads the technology and processes for BIM according to the ISO 19650 series. He is responsible for developing and coordinating a more integrated project delivery between the structural, environmental and architectural teams through BIM and the deliverable packages utilising cross platform workflows and parametric optimisations. He has been supporting the BIM process in innovative ways for large architectural practices for many years and has a thorough understanding of British Standards, PAS 1192 and ISO 19650 framework.

[in /in/froberti](https://www.linkedin.com/in/froberti)

[@F_Roberti_](https://twitter.com/F_Roberti_)

[@ f.roberti@wilkinsoneyre.com](mailto:f.roberti@wilkinsoneyre.com)



Alexandros Bertzouanis is BIM Coordinator at WilkinsonEyre. He is an Architect, a BIM Expert and a coder. His architectural works include residential, mixed use and educational projects in all stages in design and construction. He is an expert user of Revit, Dynamo and has excellent programming skills in a variety of languages including C#, .Net, Python and Java. He uses Revit API, integrating parametric design and other computation techniques into Architecture.

[in /in/alexandros-bertzouanis](https://www.linkedin.com/in/alexandros-bertzouanis)

[@alex_berd_](https://twitter.com/alex_berd)

[@ a.bertzouanis@wilkinsoneyre.com](mailto:a.bertzouanis@wilkinsoneyre.com)

Contents:

Introduction

1.1	Learning objectives.....	4
1.2	Why we create this workflow.....	4
1.3	Data workflow.....	7

Decoding the Flex License data

2.1	Exporting a Log file.....	13
2.2	Decode & Structure – Flex License Data Log.....	14

Scrubbing Data

3.1	Plugins development.....	15
3.2	License Service plugin.....	16
3.3	Revit plugin.....	16

Open Source Database - MySQL

4.1	MySQL database (Sample Data No 1).....	18
4.2	Creating MySQL data tables.....	20
4.3	Transform data – MySQL triggers.....	24
4.4	Importing sample data to MySQL.....	28

License Analytics

5.1	Connecting Power BI to MySQL.....	28
5.2	Importing the Data from a TXT file (Sample Data No 2).....	30
5.3	License Dashboard Analysis (Sample Data No 3).....	35
5.4	License Analytics and Project Usage.....	40
5.5	License Cost per Project.....	41
5.6	Identify When Extra Licenses Are Required	44

Conclusion.....	48
------------------------	-----------

Introduction

This class will analyze and structure the data from the FlexLM License Manager to create a live license dashboard using an open-source set of tools.

We will use Business Intelligence platforms (BI) and MY SQL to identify the demand and consumption of licenses per application, group of applications, and users per design team.

The interactive dashboards will provide valuable statistics of the license usage, which help to predict the license consumption for future projects.

1.1 Learning Objectives

These are the learning objectives for this class:

- Learn how to prepare the open-source database to receive data from the FlexLM License Manager.
- Learn how to identify how many licenses a single person is using after opening multiple applications.
- Learn how to track the license usage per user to identify consumption and demand.
- Learn how to create a license dashboard with an open-source database.

1.2 Why we create this workflow

License insights are critical for business management to understand and predict when would be necessary to purchase more Autodesk licenses and avoid projects teams running out of licenses.

Currently, the license insights are not available to use on network licenses, and the unique option is third party plugins.

There are third-party plugins to create license analyses, but if you develop your own application, you can have control of your data, create multiple dashboards and save money in the long term as you don't need to pay for the plugin.

Every company needs a better understanding when additional licenses are required and have a process to demonstrate the license usage on projects and identify the users.

Benefits of Analysis Tool:

- Better management of software resources
- Maximizing license utilization

- Software license optimization
- Improve the project budgets
- Identify when extra licenses are required

When licenses are not available, the users may send emails to all staff requesting a license and ask to close the software if it is not in use.

Typical email:



When users start to receive this type of email, there are two typical outcomes.

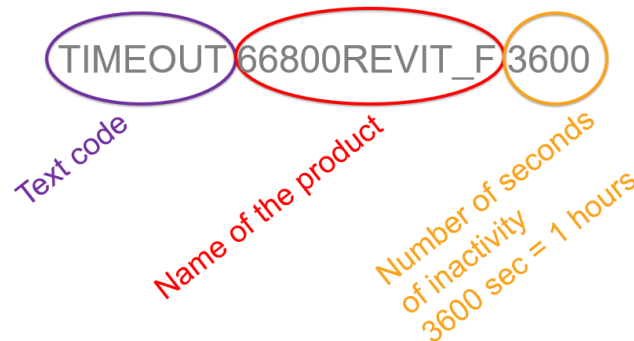
- Users will close the software if they are not using it.
- Users will keep the software always open until the license issue has been resolved.

If the design team have a deadline, they will tend to keep the software open, which increases the problem.

In this case, we advise introducing the license “Timeout” from 1 hour to 30 minutes or the time frame that is suitable for your company.

We won’t cover in detail the License Timeout in this class, but it is straightforward to implement.

You will need to open the Notepad and edit the file adskflex.opt. The file “.opt” could have a different name in your organisation.



The Autodesk software will take a license back when you return to your software.

Cascade Licensing

Companies purchase licenses over time from multiple software packages such as Single AutoCAD license, Revit Architecture, Building Design Suite or the AEC collection.

The cascade licensing with multiple software packages may utilize more licenses than if you have only a single package such as the AEC Collection.

The objective of this class is not to explain in detail the Cascade Licensing, but we will give one example to show the benefits of utilizing the AEC Collection software package.

Examples of Cascade Licensing:

1- Mixed pool of single products and Suite product licenses

A customer has two AutoCAD licenses, two Building Design Suite Premium licenses, for a total of four licenses.

- User A runs AutoCAD and pulls one AutoCAD license.
- User A then runs Revit, pulling one Revit license from the Building Design Suite.

At this point, User A has consumed **two licenses** on his machine.

If the company upgrades the 4 licenses for the AEC Collection, the cascading will work as follows:

- User A runs AutoCAD and pulls one AEC Collection license.
- User A then runs Revit and it will use the same license from the AEC Collection.

At this point, User A has consumed **one license** on his machine.

There are clear benefits to upgrading the licenses to the AEC collection as it will use fewer licenses.

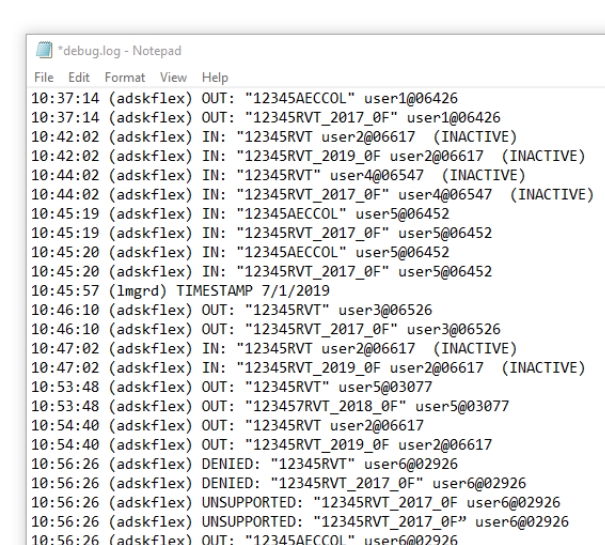
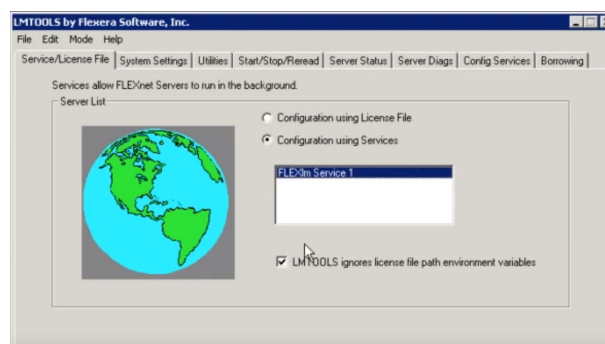
Overall, it is critical to have a tool that provides a better understanding of the license usage to plan and predict when it would be necessary to purchase more licenses.

1.3 Data Workflow

Understanding the data workflow will help you to replicate this process in your company.

Our workflow is divided into five parts:

1. The primary data source is the FlexLM License Manager that will export the data log.



2. The Revit Synchronization is the data source to identify users in the projects. The other alternative if you don't use Revit is to use Excel to create a list of projects and assign users.



Revit Synchronization
Data source to identify
users in projects

Alternative: Excel list



3. WilkinsonEyre has created two plugins to organize and push data to MySQL.

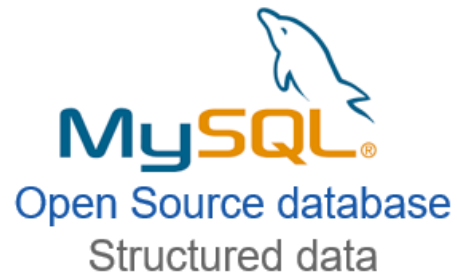
License Service plugin:

WilkinsonEyre License Service
Organize & Push data

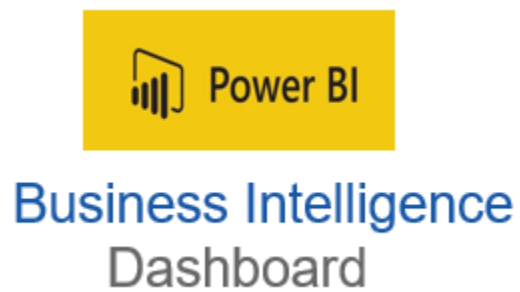
Revit Plugin:

WilkinsonEyre Revit Plugin
Organize & Push data

4. MySQL is the open source database that will be used to structure data.



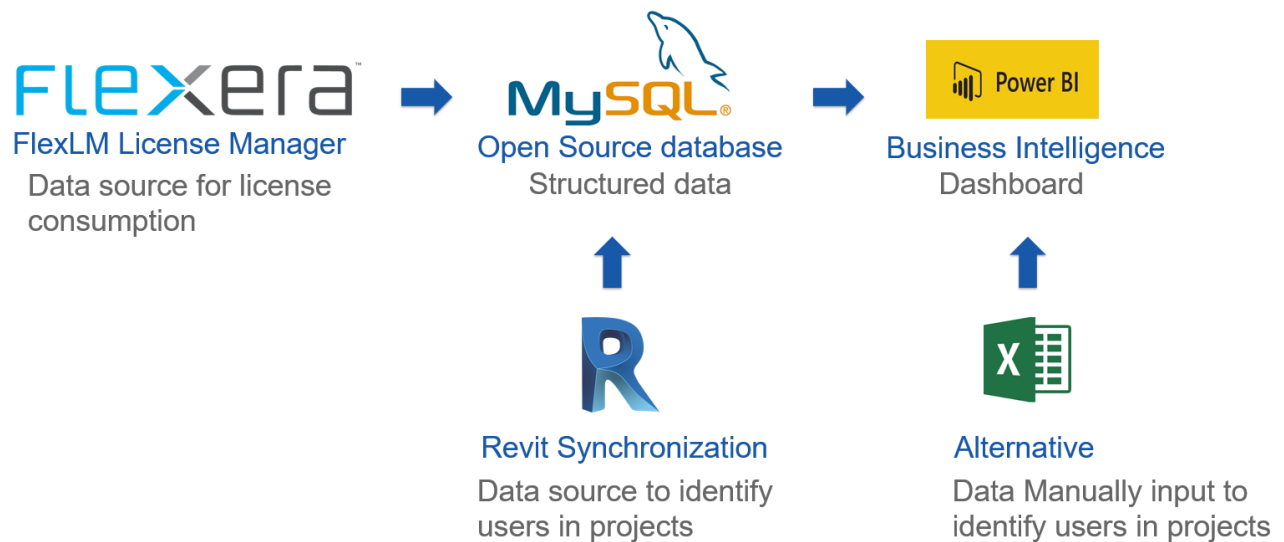
5. We are using Power BI to create the dashboards.



Data Workflow Diagram

The FlexLM License Manager and the Revit Synchronization or the Excel project/user list can provide data that will be structured in the open-source MySQL.

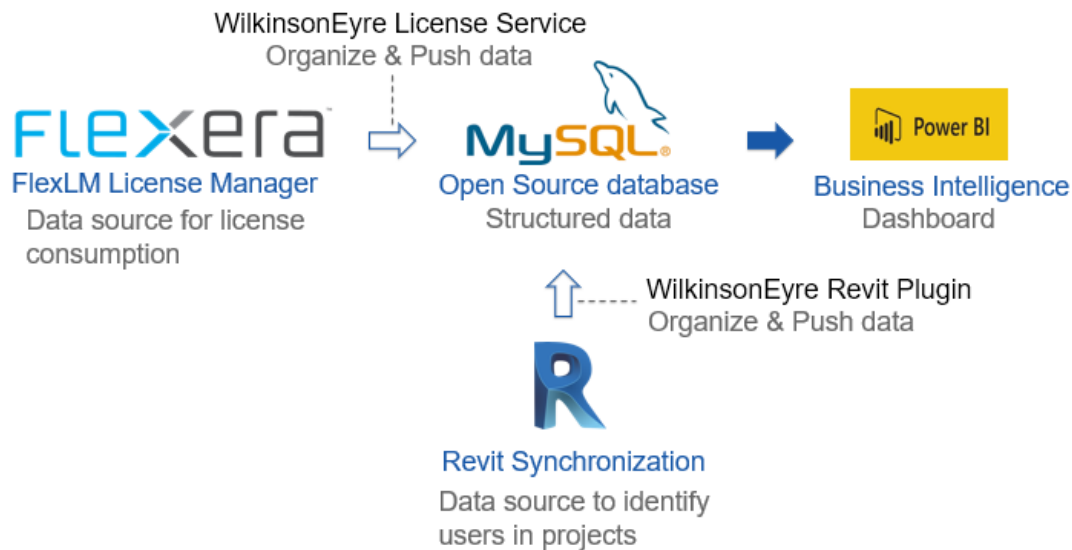
Power BI will be used to create the dashboards.



There is a problem in the diagram above as the FlexLM License Manager and the Revit Synchronization do not push data automatically to MySQL.

We had to create two plugins to organize and automatically push data to MySQL. This process provides the live updates to our dashboards.

The image below shows the complete data workflow of our class which includes the two plugins.



The data from MySQL will be used to create the following dashboards with PowerBI.

- Current license usage
- Historical license usage
- Number of license per user
- License usage time per user
- Software version
- License usage per project
- Predict when licenses are needed

We can divide the dashboards into two sets for you to understand which plugin is required to organizing and pushing the data to MySQL.

WilkinsonEyre License Service Organize & Push data

The WilkinsonEyre License Service plugin is required to produce the following dashboards:

- Current license usage
- Historical license usage
- Number of license per user
- License usage time per user
- Software version
- Predict when licenses are needed

WilkinsonEyre Revit Plugin Organize & Push data

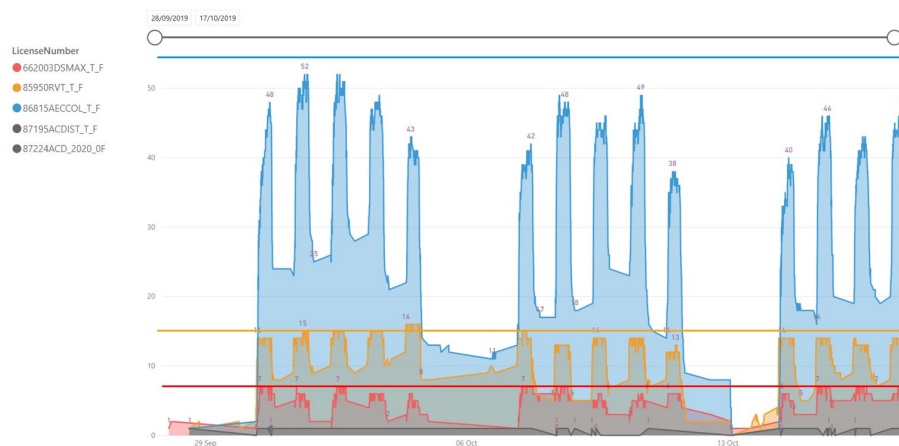
The WilkinsonEyre Revit plugin is required to produce the following dashboard:

- License usage per project

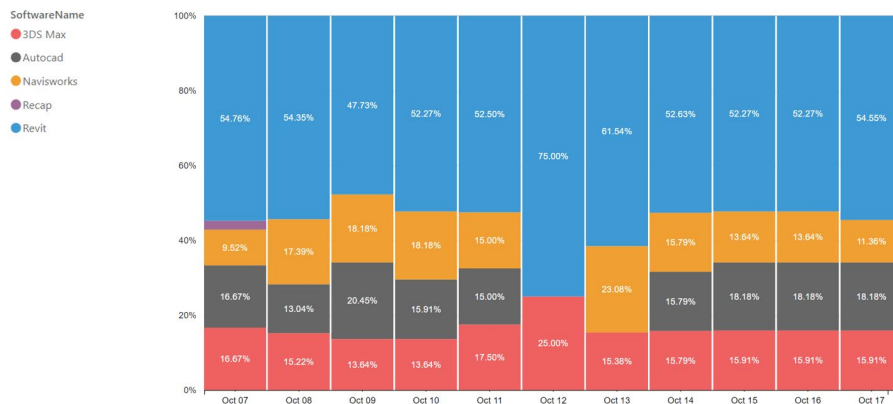
The Revit plugin is used to provide information related to users in the project. You can use Excel as an alternative to assigning users to projects.

The combination of Structured Data and Power BI, during this session, you will learn to create dashboards as the images below.

Historical License Usage



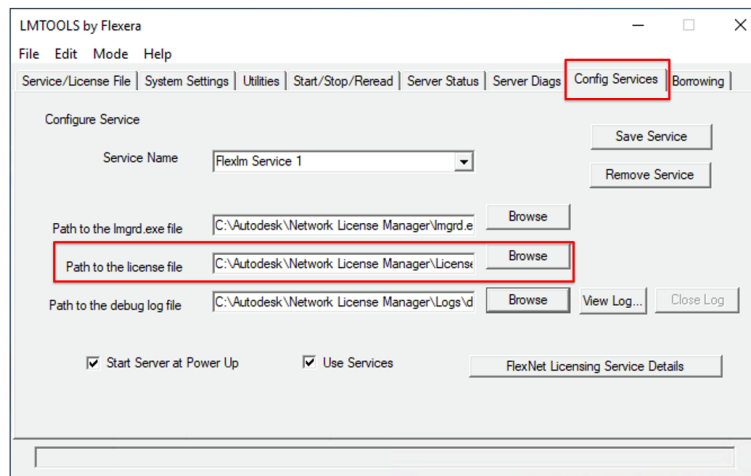
Software Version Usage



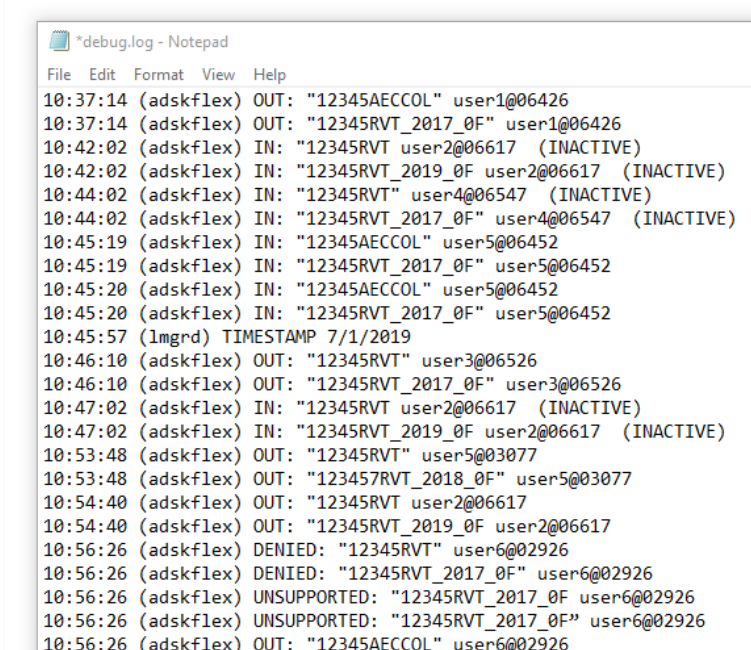
Decoding the Flex License data

2.1 Exporting a Log file

We can start getting information about the Autodesk licenses by utilizing the LMTools License software and setting up the path to a specific license file.



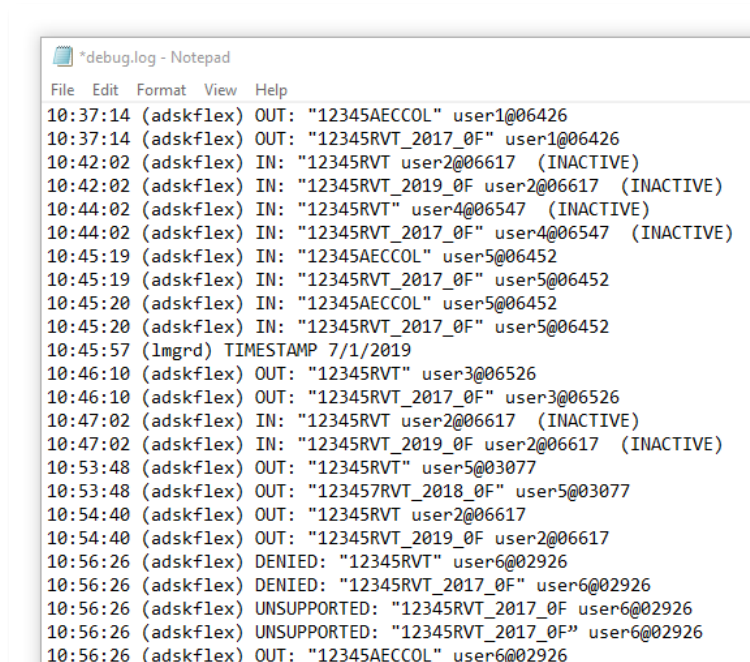
Once, we restart the License Server, license server's actions and all users actions will be tracked at the log file. If we open the log file with notepad, it will look similar to the image below.



2.2 Decode & Structure – Flex License Data Log

The data from the LOG file need to be decoded and structured before pushing the information to MySQL.

Data Log File:

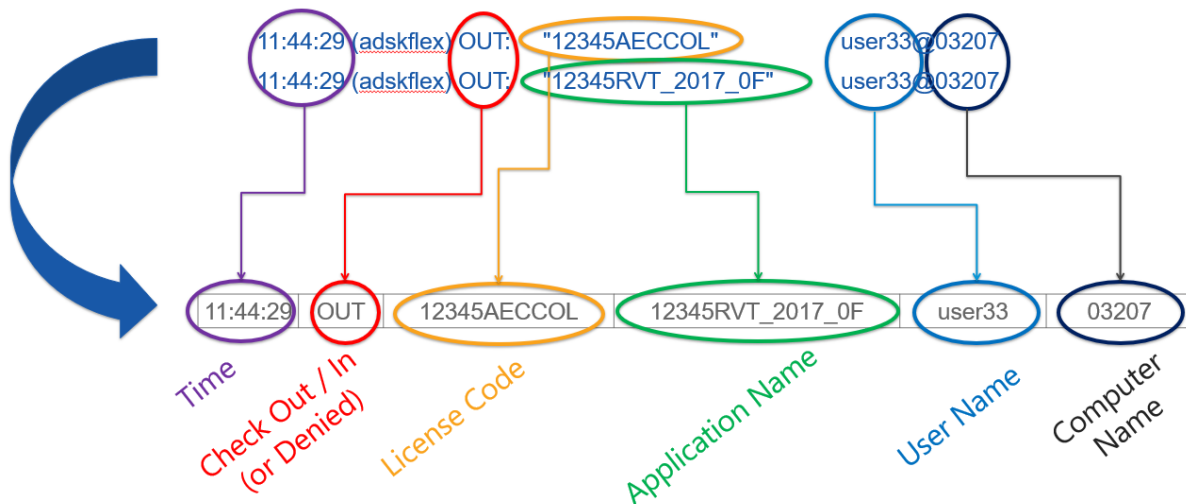


```
*debug.log - Notepad
File Edit Format View Help
10:37:14 (adskflex) OUT: "12345AECCOL" user1@06426
10:37:14 (adskflex) OUT: "12345RVT_2017_0F" user1@06426
10:42:02 (adskflex) IN: "12345RVT user2@06617 (INACTIVE)
10:42:02 (adskflex) IN: "12345RVT_2019_0F user2@06617 (INACTIVE)
10:44:02 (adskflex) IN: "12345RVT" user4@06547 (INACTIVE)
10:44:02 (adskflex) IN: "12345RVT_2017_0F" user4@06547 (INACTIVE)
10:45:19 (adskflex) IN: "12345AECCOL" user5@06452
10:45:19 (adskflex) IN: "12345RVT_2017_0F" user5@06452
10:45:20 (adskflex) IN: "12345AECCOL" user5@06452
10:45:20 (adskflex) IN: "12345RVT_2017_0F" user5@06452
10:45:57 (lmgrd) TIMESTAMP 7/1/2019
10:46:10 (adskflex) OUT: "12345RVT" user3@06526
10:46:10 (adskflex) OUT: "12345RVT_2017_0F" user3@06526
10:47:02 (adskflex) IN: "12345RVT user2@06617 (INACTIVE)
10:47:02 (adskflex) IN: "12345RVT_2019_0F user2@06617 (INACTIVE)
10:53:48 (adskflex) OUT: "12345RVT" user5@03077
10:53:48 (adskflex) OUT: "12345RVT_2018_0F" user5@03077
10:54:40 (adskflex) OUT: "12345RVT user2@06617
10:54:40 (adskflex) OUT: "12345RVT_2019_0F user2@06617
10:56:26 (adskflex) DENIED: "12345RVT" user6@02926
10:56:26 (adskflex) DENIED: "12345RVT_2017_0F" user6@02926
10:56:26 (adskflex) UNSUPPORTED: "12345RVT_2017_0F" user6@02926
10:56:26 (adskflex) UNSUPPORTED: "12345RVT_2017_0F" user6@02926
10:56:26 (adskflex) OUT: "12345AECCOL" user6@02926
```

The following image reveals how the log text file is structured.

```
11:38:02 (adskflex) IN: "12345RVT" user101@06577 (INACTIVE)
11:38:02 (adskflex) IN: "12345RVT_2019_0F" user101@06577 (INACTIVE)
11:44:29 (adskflex) DENIED: "12345RVT" user56@03207
11:44:29 (adskflex) DENIED: "12345RVT_2017_0F" user56@03207
11:44:29 (adskflex) OUT: "12345AECCOL" user33@03207
11:44:29 (adskflex) OUT: "12345RVT_2017_0F" user33@03207
```

Looking more carefully, we can understand better what the above information means. Storing this data could help us understand and analyse the license usage. We name the different data parts according to the following diagram.

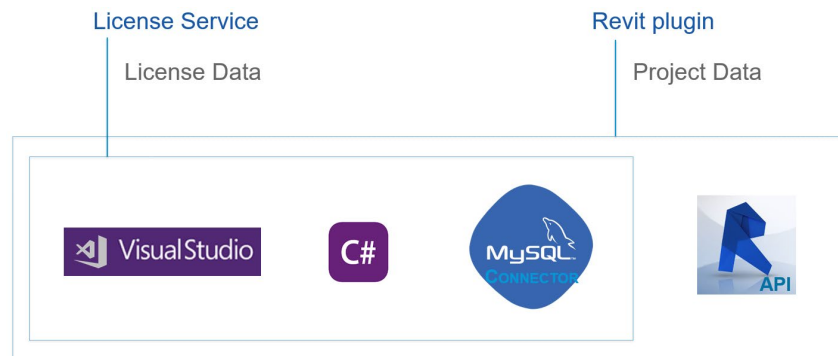


Organize Data

3.1 Application development

We developed in-house, two separate applications which are the License Service and the Revit Plugin.

The License Service plugin scrubs the data from the Flexera Log File and stores them in the MySQL server and the Revit Plugin daily stores the Revit project numbers and user names.



3.2 License Service plugin

In order to understand the license usage, we use the LM Tools' license log file to parse any relevant information according to section 2.2. We split the data into separate parts by a parser service created using Visual Studio and C#. After the service splits the data, it sends them into a MySQL database. This parser processes the following commands:

1. It reads the text file.
2. It excludes all unnecessary lines.
3. It splits every line into the information that we would like to keep, by keeping lines in doublets.
4. It places those data into separate parameters to be ready to send to the MySQL database.
5. It sends the data to the MySQL database in a relevant table.
6. It repeats the process every 1 minute reading only the new added lines of the log file.

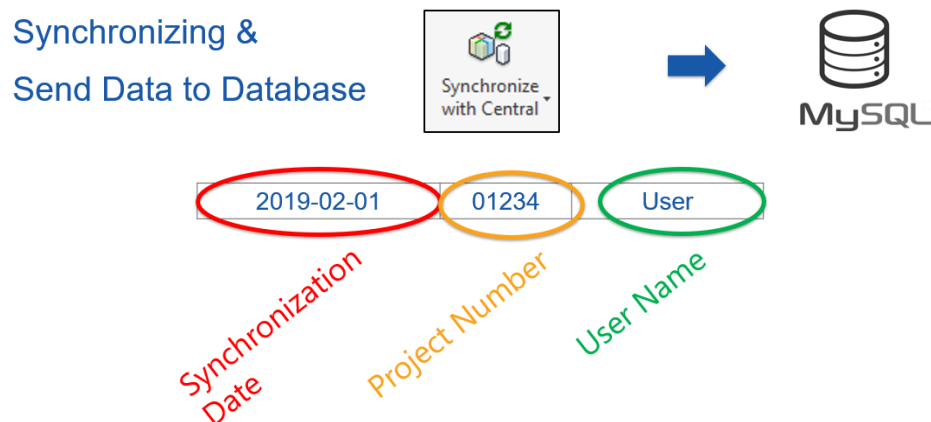
3.3 Revit plugin

In order to understand the license usage per project, we need to know in which project every user works and try to correlate them to the license usage. If users are related to specific projects, it is easy to get this information i.e. request those data from the HR department. However, if users are working across different project over the year, this is more challenging to track.

To overcome this problem, we created a Revit plugin that reveals the different projects that users are working in Revit.

For that reason, we only concentrated on users who synchronize models and excluded for instance all those who accidentally leave their Revit models open.

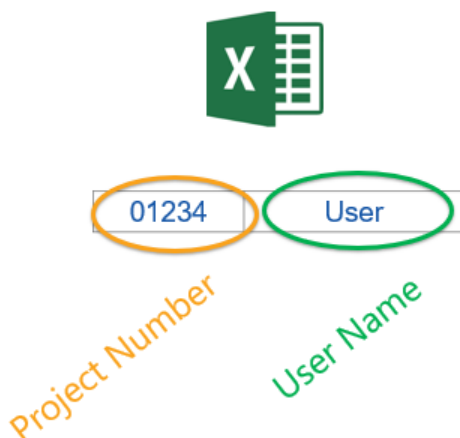
Revit plugin is developed in C#, and similarly to the License Service, it sends the acquired data to the MySQL database which can be picked up later by Power BI.



The Revit plugin we developed uses the following Revit API commands. If you don't use Revit, the alternative is to create an Excel list with Project Number and User Name.

Use MySQL Connector	using MySql.Data.MySqlClient;
Connection settings	static string connectionString = "SERVER=" + ServerIP / Name + ";" + "DATABASE=" + " license_schema " + ";" + "UID=" + UserName + ";" + "PASSWORD=" + Password ;
Create connection	MySqlConnection connection = new MySqlConnection(connectionString); connection.Open();
Define command	MySqlCommand command = new MySqlCommand("INSERT INTO revit_licenses_Autodesk ('DateTime', ...)", connection);
Execute command	command.ExecuteNonQuery();
Close Connection	connection.Close();

Alternative

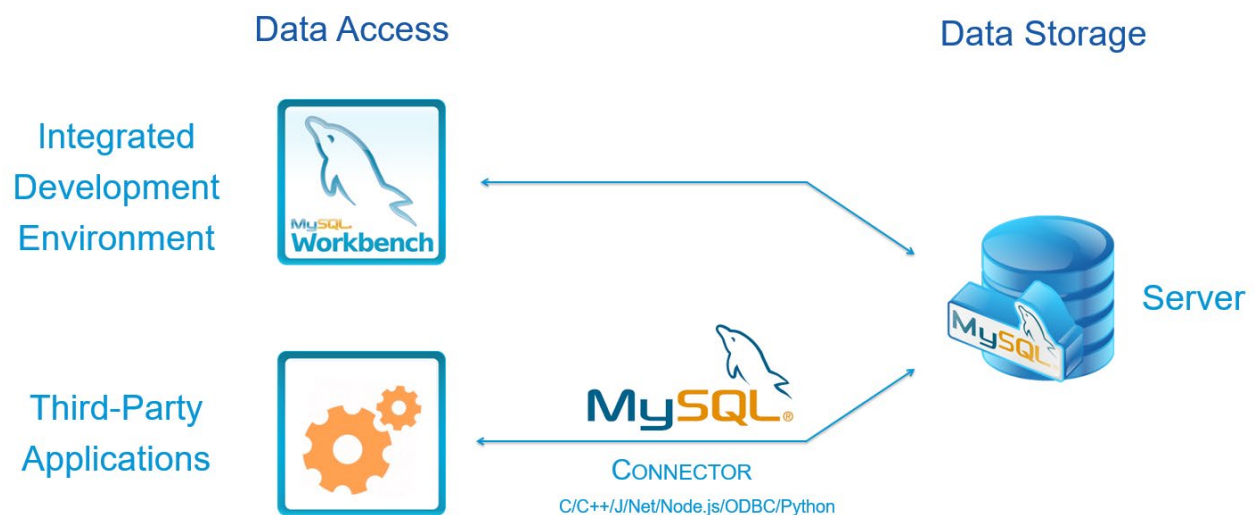


Open Source Database - MySQL

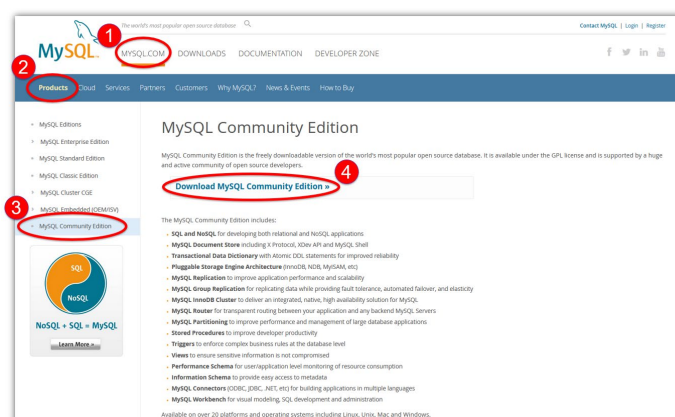
4.1 MySQL database

MySQL is a free and open-source relational database management system which could store structured and Non-Structured Data. It has a set of different tools for developing and managing data, and due to its large community, it is very well documented and easy to learn and to fix errors.

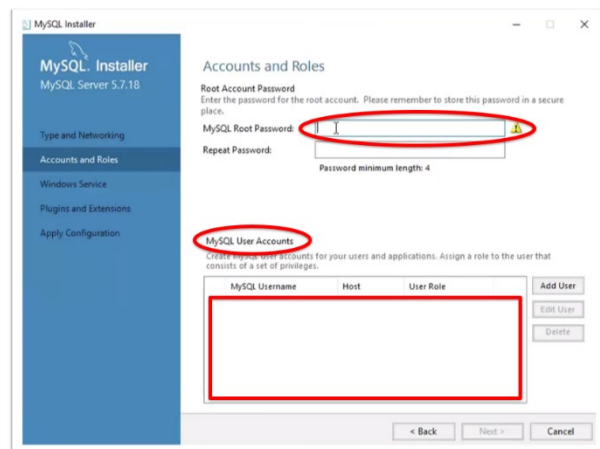
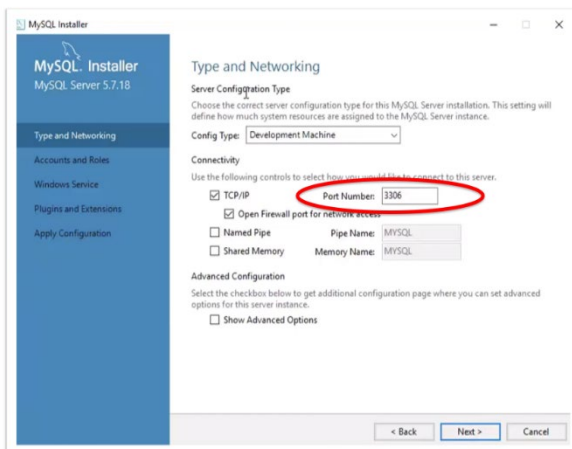
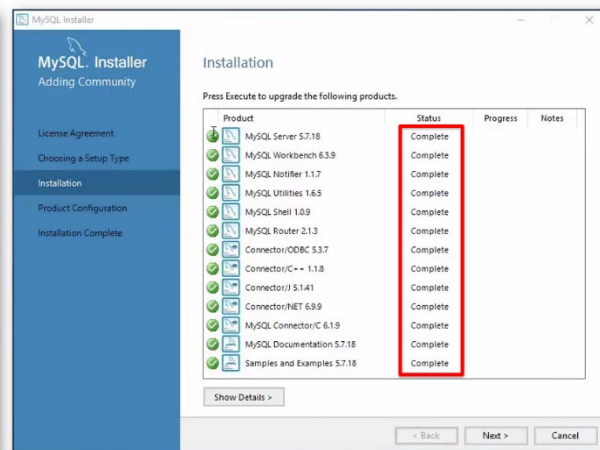
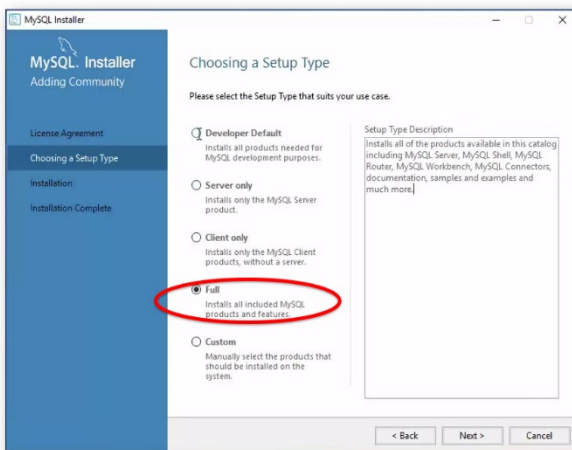
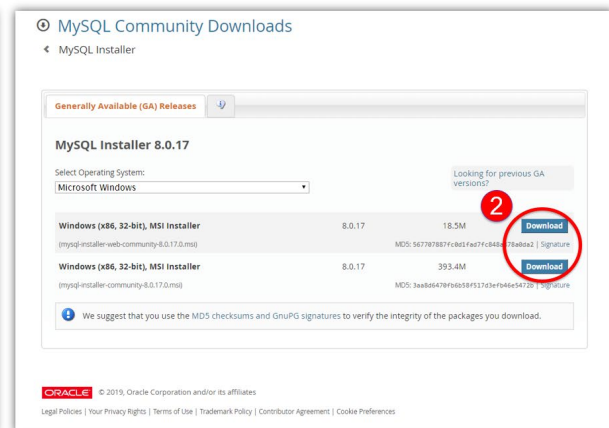
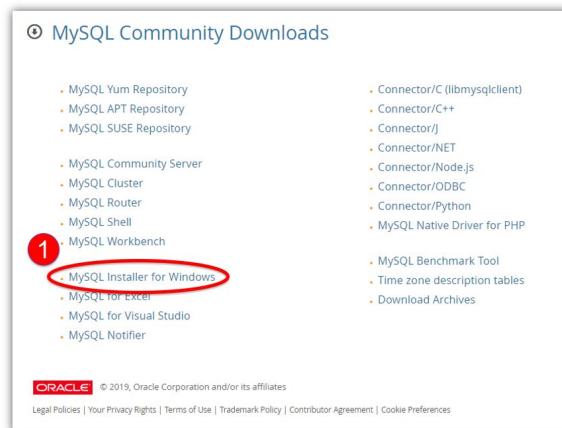
The basic components of MySQL are MySQL Server which stores the data and the MySQL Workbench which manages the data. Third-party applications can also access MySQL server data by using the MySQL connector component.



The Installation of MySQL is quick and easy. You can download and install it from the MySQL website <https://www.mysql.com/>.



Installation Steps are easy to follow. Make sure that you perform a full installation, including all the MySQL components and define a password to be able to access it later.

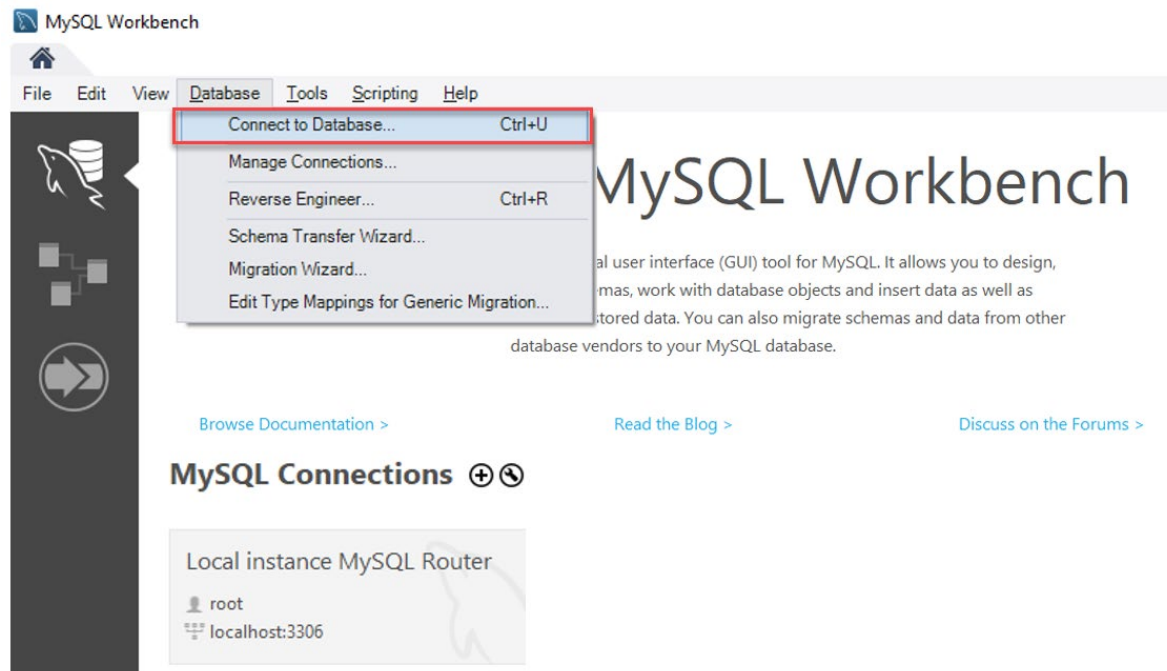


4.2 Creating MySQL data tables

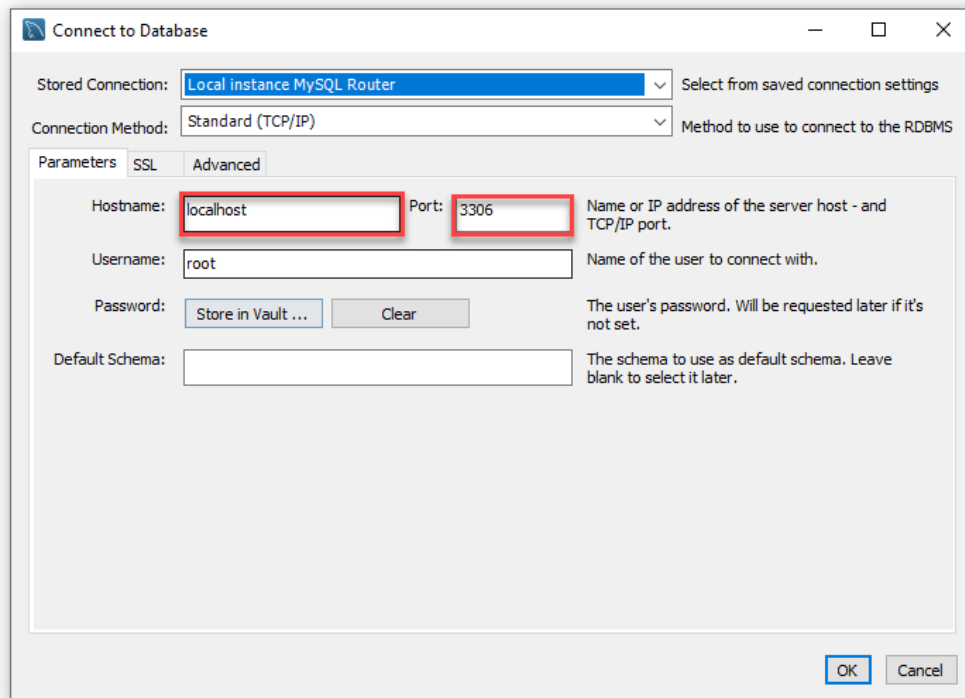
Start here to use the “Dataset No 1”

It is not too challenging to create few tables in MySQL database. We can access MySQL server using MySQL Workbench, which is an easy-to-use visual database design tool. This tool performs most of the MySQL operations, including database design, development and administration.

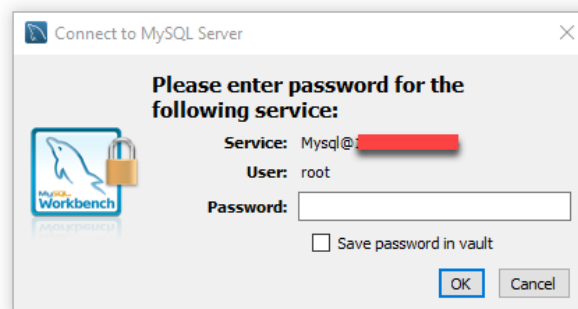
In the first page of MySQL Workbench, we create a connection.



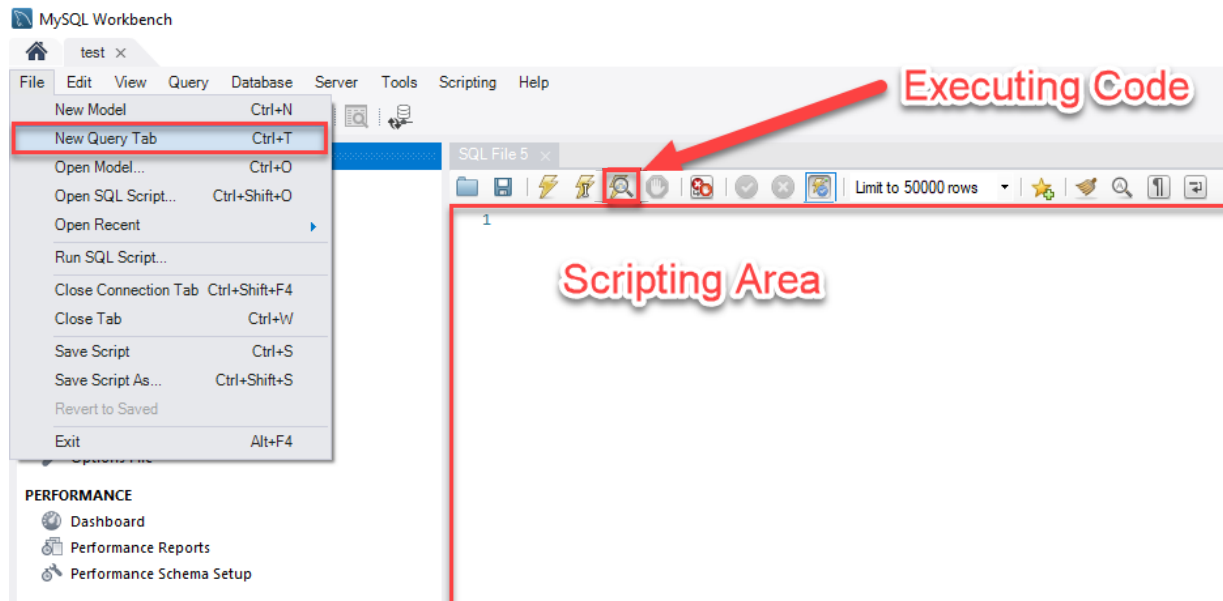
If the database is installed locally (at the same computer), we can access it by using Hostname “localhost”, otherwise we need to provide the relevant IP. We also need to provide server’s port, the default value is 3306.



Provide the credentials given during the installation process.



Connect to the database and create a new query tab. Every time we want to execute code, we copy-past the given code to the “Scripting Area”, and then we click the “Execute Code” using the icon shown below.



We need first to create a database by providing a name. For this exercise, we create the “license_schema” database using the following command. Create this database executing the SQL script below in the MySQL Workbench.

```
CREATE SCHEMA `license_schema`
```

To push the data from the two plugins we previously created, we need to create a few tables having the same columns as the given data.

For the License Usage Data, we create a table to store the following data. Create this table executing the SQL script below in the MySQL Workbench.

1.Input Table



```
CREATE TABLE `license_schema`.`input_table` (  
  `ID` INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  `DateTime` DATETIME(3) NOT NULL UNIQUE,  
  `Action` VARCHAR(45) NULL,  
  `LicenseCode` VARCHAR(45) NULL,  
  `ApplicationName` VARCHAR(45) NULL,  
  `User` VARCHAR(45) NULL,  
  `Computer` VARCHAR(45) NULL  
)
```

4.3 Transform data – MySQL triggers

The table we created at the previous step holds the input data from the License Parsing Service.

However, to extract important information for our analysis through Power BI, many calculations are necessary.

To speed up our later analysis in Power BI, we instruct MySQL to pre-calculate and save most of the calculations needed. For this reason, we will create a MySQL trigger. The trigger will run every time a new entry is imported to our license database. Trigger calculations will generate all necessary statistics we need for our visualizations, and it will be stored to a separate table.

We create a second table named “Triggered Table” storing all the calculated license statistics shown below. Create this table executing the SQL script below in the MySQL Workbench.

2.Triggered Table



```
CREATE TABLE `license_schema`.`triggered_table` (
  `ID` INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  `DateTime` DATETIME(3) NOT NULL UNIQUE,
  `Action` VARCHAR(45) NULL,
  `LicenseCode` VARCHAR(45) NULL,
  `ApplicationName` VARCHAR(45) NULL,
  `User` VARCHAR(45) NULL,
  `Computer` VARCHAR(45) NULL,
  `TotalApplicationInstancesPerLicense` INT(3) NULL,
  `LicenseDuration` TIME NOT NULL,
  `TotalLicensesPerUser` INT(3) NULL,
  `TotalLicenses` INT(3) NULL,
  `InstancesPerApplication` INT(3) NULL,
  `ApplicationInstancesPerLicense` INT(3) NULL,
  `ApplicationDuration` TIME NOT NULL,
  `TotalApplications` INT(3) NULL,
  `TotalApplicationsPerUser` INT(3) NULL
)
```


Before we continue, we need to explain what the “Action” parameter created by the LM Tools Flexera Log File means.

Action	Meaning
• OUT:	The software license is acquired by the User.
• IN:	The software license is released by the User.
• DENIED:	User requested a license but there are not enough licenses in the Selected license pool. Server searches for license availability in the next license pool.

We will create the actual trigger to make all the necessary calculations and save the required statistics in the “Triggered” Table.

The code of the MySQL trigger is provided as a separate file name called “MySQL Trigger Code” which is written in SQL Code saved in a txt file. We need to open this file with notepad and copy-paste the code into the MySQL Scripting Area illustrated in Section 4.2. After that, we execute the code, and the trigger is ready for action.

The Trigger runs every time when a new entry comes into the database. Below we will not go into the coding, but we will explain the basic principles of how it calculates the statistics. We will illustrate only a few of the script’s operations.

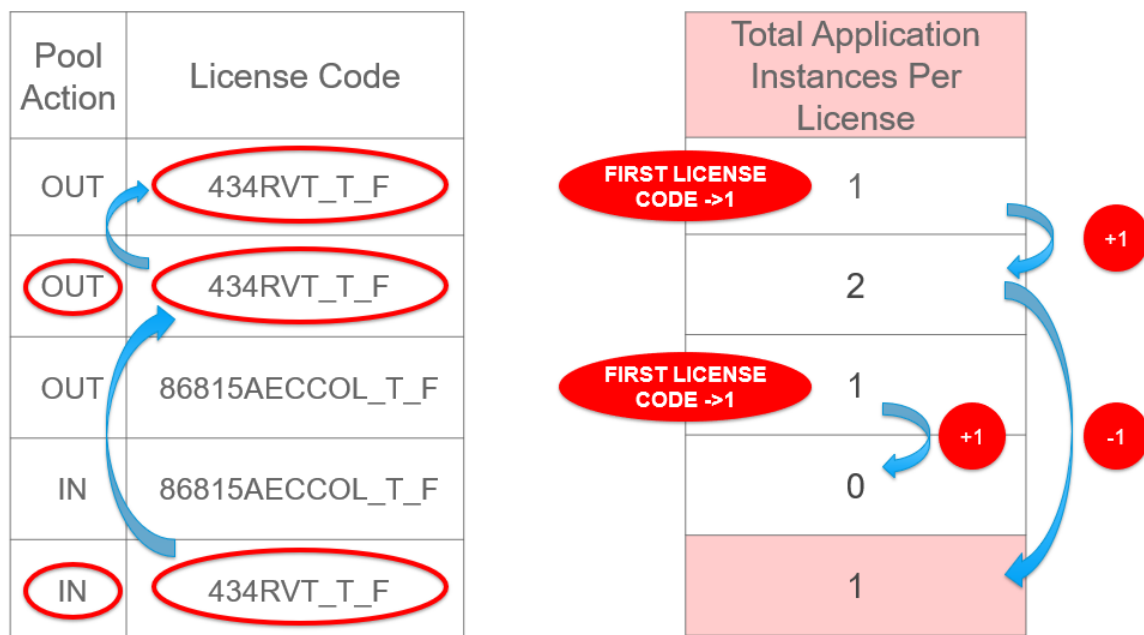
The logic is equivalent to the Pseudo Code of the script. However, if you would like to go through the code, you can navigate and read the “MySQL Trigger” text file.

Total Application Instances Per License:

To calculate this process, the trigger searched the whole database and trying to find previous entries of the same user utilizing the same license.

If there is no previous entry, which means that it is the first time this license is used, so the value is defined as 1.

If not, the value incrementally increases or decreases, depending on the pool's action. If the pool's action is "out", value increases by 1. If action is "in" value decreases by 1.

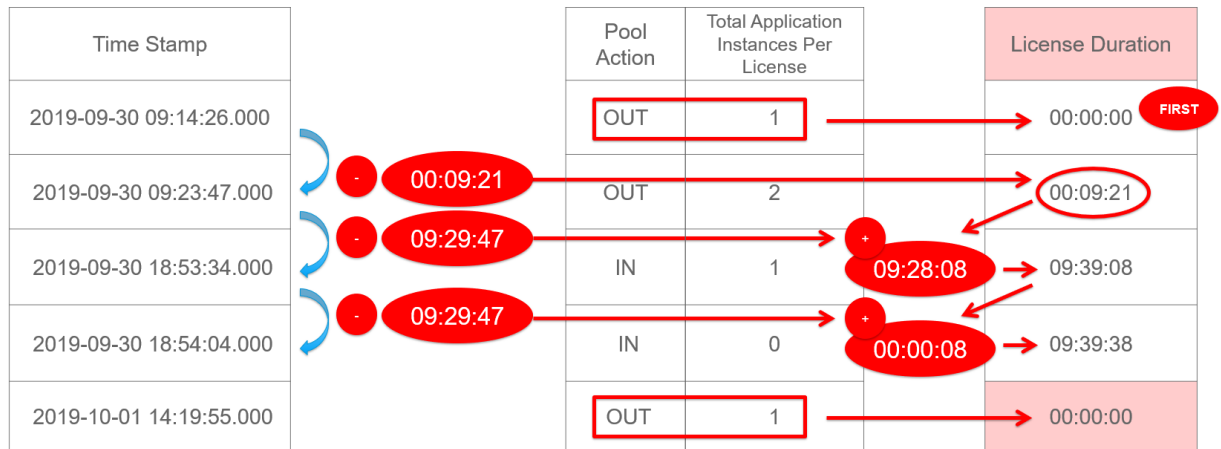


License Duration

To calculate the License Duration, the trigger looks at the "Total Application Instances Per License" value that we previously created and the current pool action.

If the previous value of "Total Application Instance Per License" is 0, it means that license is not used for this user. Then the license Duration calculated will be 0:00.

If "Total Application Instance Per License" is other than 0 the license duration increases by the difference between the current and the previous entry of the same license code and user.

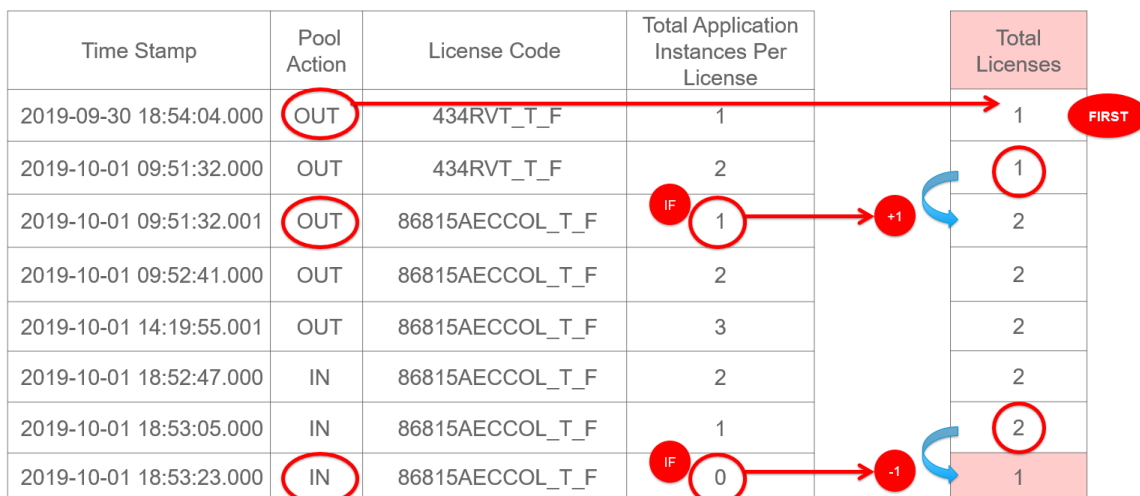


Total Licenses

To calculate the Total Licenses, the trigger looks at the previous “Total Application Instances Per License” and the Pool action value separately for every user.

If Pool Action is “out” and “Total Application Instances Per License” value is 1 then Total Licenses increase by 1.

If Pool Action is “in” and “Total Application Instances Per License” value is 0 then Total Licenses decreased by 1. In all other instances, Total Licenses value remain the same.



Using similar methodologies, we calculate all other statistics “Total Licenses per User”, “Instances per Application”, “Application Instances Per License”, Application Duration”, “Total Applications”, and “Total Applications per User”.

4.4 Importing sample data to MySQL

After parsing the LMTOOLS license log, created in section 3.2, every time we send new license data to the MySQL database at the input table, the MySQL trigger will calculate all the statistics discussed in the previous section (4.3). Rows of input and triggered tables are relative to each other.

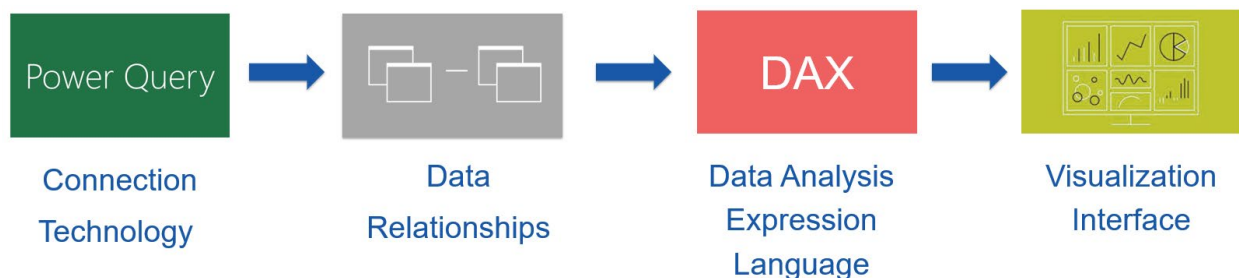
For the purpose of this exercise, we prepared an example dataset with parsed license data ready to use as MySQL inputs. This file includes a variety of users worked in a two-week period. To import those data in MySQL database in the tables we generated at the previous section, copy the “Parsed LMTOOLS License Data.txt” files at the “C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\” location. Then, execute the following SQL Script in the MySQL Workbench.

```
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Parsed
LMTOOLS License Data.txt"
INTO TABLE `license_schema`.`input_table`
FIELDS TERMINATED BY "\t";
```

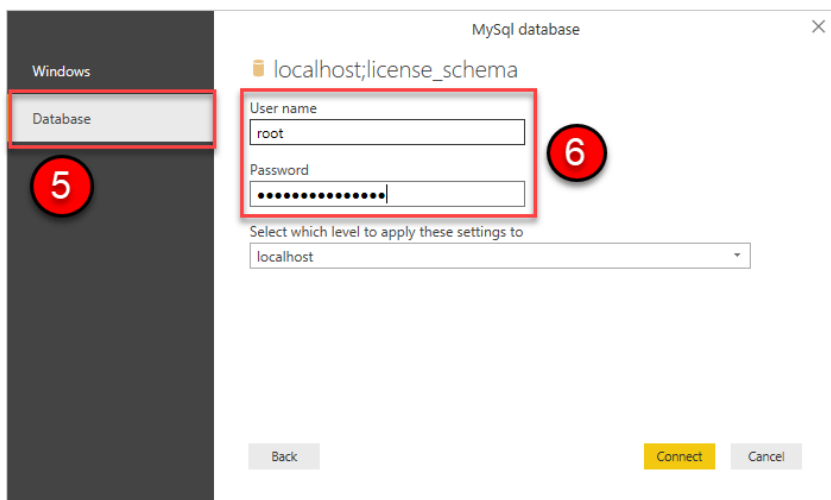
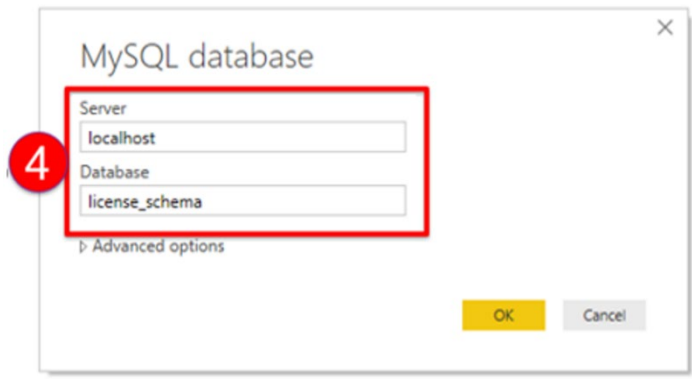
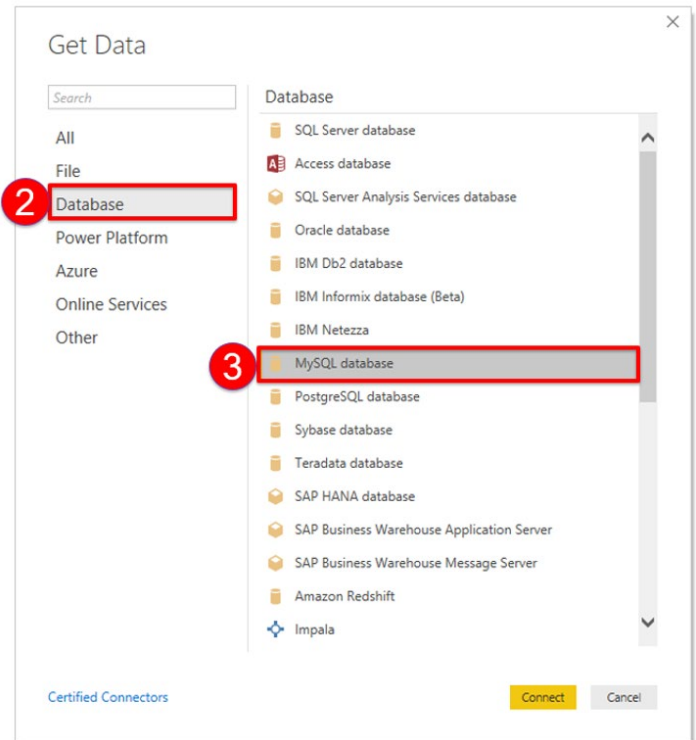
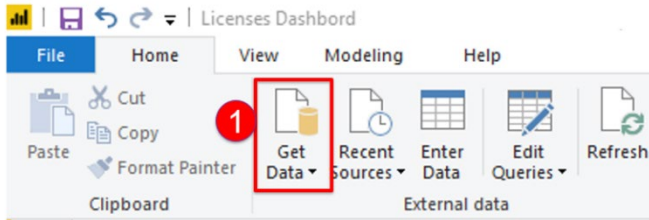
License Analytics using Power BI

5.1 Connecting Power BI to MySQL

Power BI Desktop is a business analytics service which provides interactive visualizations and advanced analytics to allow users easily create their own reports and dashboards. It includes the following four interfaces: Power Query technology, Data Relationships (Model Interface), Data Analysis Expression Language and the Visualization Interface (Report Interface).



Firstly, we need to let Power BI access the data we previously stored in the MySQL database. Use the “Get Data” command from the Home toolbar and provide the Server IP “localhost”, the Database name “license_schema”, User Name “root” and the Password credentials used in the MySQL installation. Then select the “license_schema.triggered_table” table we previously created.



Navigator

Display Options

- localhost: license_schema [2]
 - license_schema.input_table
 - license_schema.triggered_table**

7

license_schema.triggered_table

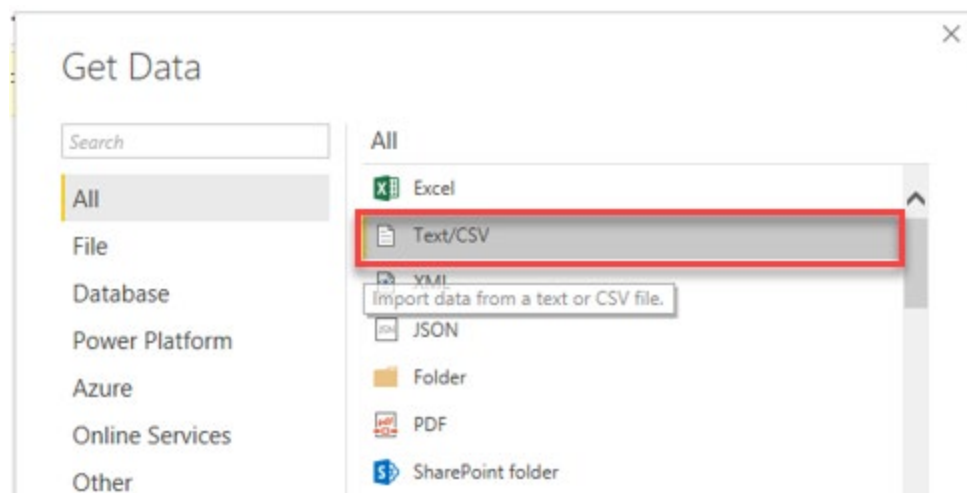
ID	DateTime	Action	LicenseCode	ApplicationName
1	28/09/2019 13:19:53	OUT	AUTOCAD	87084ACD_2019_1
2	30/09/2019 08:40:48	IN	AUTOCAD	87084ACD_2019_1
3	30/09/2019 09:23:41	OUT	AEC COLLECTION	86706RVT_2017_1
4	30/09/2019 09:23:46	OUT	AEC COLLECTION	86706RVT_2017_1
5	30/09/2019 09:28:00	OUT	AEC COLLECTION	86706RVT_2017_1
6	30/09/2019 09:30:30	OUT	AEC COLLECTION	86706RVT_2017_1
7	30/09/2019 09:35:59	OUT	AEC COLLECTION	86706RVT_2017_1
8	30/09/2019 09:39:49	OUT	AEC COLLECTION	86604ACD_2017_1
9	30/09/2019 09:40:27	OUT	AEC COLLECTION	87149RVT_2019_1
10	30/09/2019 09:41:29	OUT	AEC COLLECTION	86706RVT_2017_1
11	30/09/2019 09:42:31	OUT	AEC COLLECTION	86706RVT_2017_1
12	30/09/2019 09:43:09	OUT	AEC COLLECTION	86706RVT_2017_1
13	30/09/2019 09:43:46	OUT	AEC COLLECTION	87084ACD_2019_1
14	30/09/2019 09:46:33	OUT	AEC COLLECTION	86706RVT_2017_1
15	30/09/2019 09:46:49	IN	AEC COLLECTION	86706RVT_2017_1
16	30/09/2019 09:46:53	OUT	AEC COLLECTION	86706RVT_2017_1
17	30/09/2019 09:48:21	OUT	AEC COLLECTION	87149RVT_2019_1
18	30/09/2019 09:48:52	IN	AEC COLLECTION	86706RVT_2017_1
19	30/09/2019 09:52:33	OUT	AEC COLLECTION	86766NAVMAN_2
20	30/09/2019 09:53:22	OUT	AEC COLLECTION	86706RVT_2017_1
21	30/09/2019 09:54:38	IN	AEC COLLECTION	86706RVT_2017_1
22	30/09/2019 09:55:36	OUT	AEC COLLECTION	86766NAVMAN_2
23	30/09/2019 09:56:36	OUT	AEC COLLECTION	86706RVT_2017_1

Select Related Tables Load Transform Data Cancel

5.2 Importing the Data from a TXT file (Sample data No. 2)

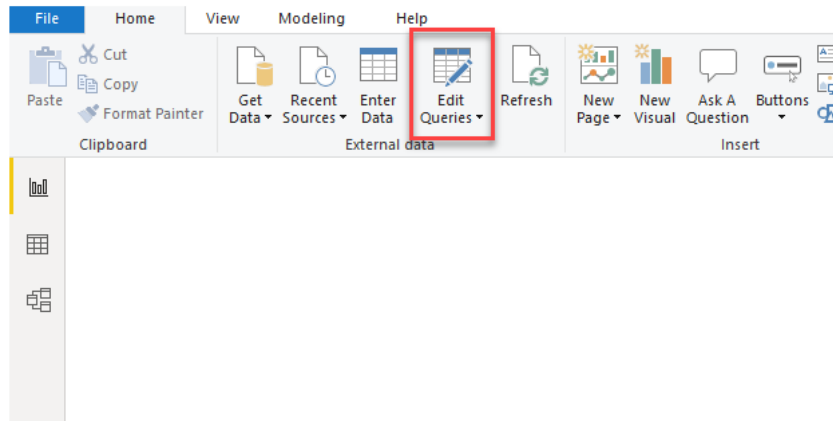
Start here to use the “Dataset No 2”

To analyse project information we need to utilise the “project usage” information. Those data are stored in a separate table in the MySQL database. For the convenience of this exercise, we provided an equivalent txt file that we can use. We can import it using the “Get Data Text/CSV” commands below.

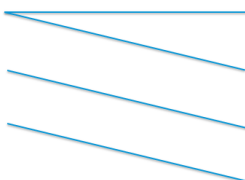


Only for those who skipped section 4 and want to experiment just with Power BI, need to repeat the “Get Data Text/CSV” command using the second dataset and importing the license usage data from the “license_schema triggered_table” text file.

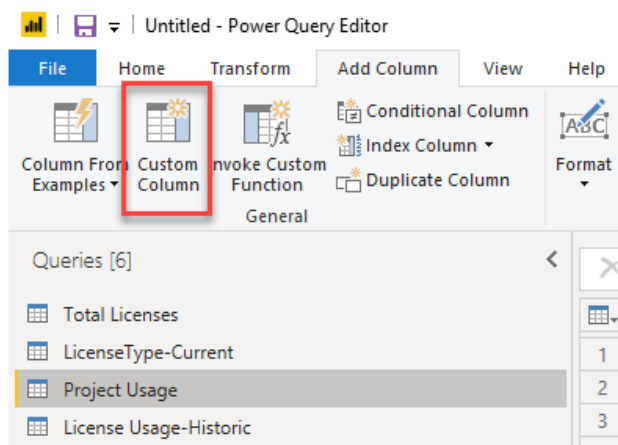
Use the Power Query interface clicking at the “Edit Queries” button to transform the data and make them ready to visualize.



In the Project Usage table, we need to create a new “Date&User” column to join the license usage and the project usage and later to be able to calculate the project cost.

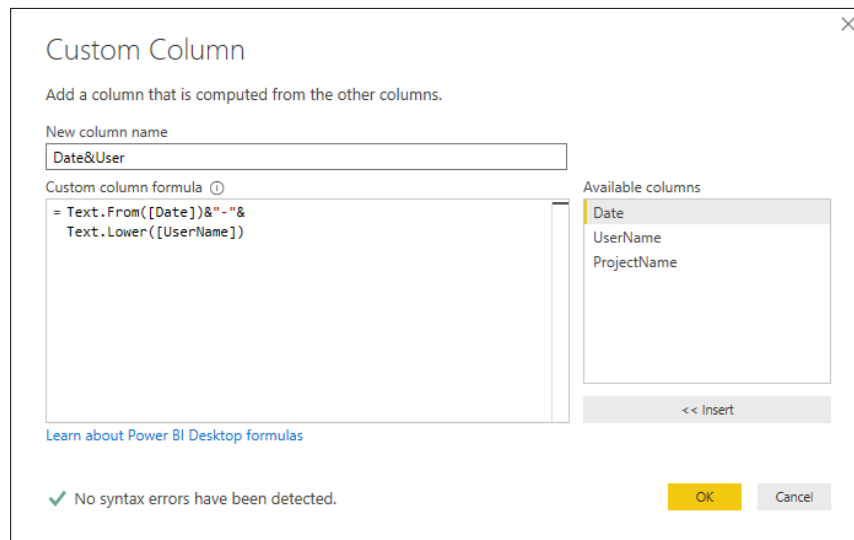
Project	Date&User		Date&User	Working Time
0005	20/10/2019-User1		20/10/2019-User1	6:00
0013	20/10/2019-User2		20/10/2019-User1	4:30
			20/10/2019-User2	7:00
0005	21/10/2019-User2		20/10/2019-User2	2:00

Add a calculated custom column at the “Project Usage” data using the Custom Column command from the Add Column Panel.



Paste the following script creating the “Date&User” Column.

```
Text.From([Date])&"-"&Text.Lower([UserName])
```



Custom Column

Add a column that is computed from the other columns.

New column name
Date&User

Custom column formula ⓘ
= Text.From([Date])&"-"&
Text.Lower([UserName])

Available columns
Date
UserName
ProjectName

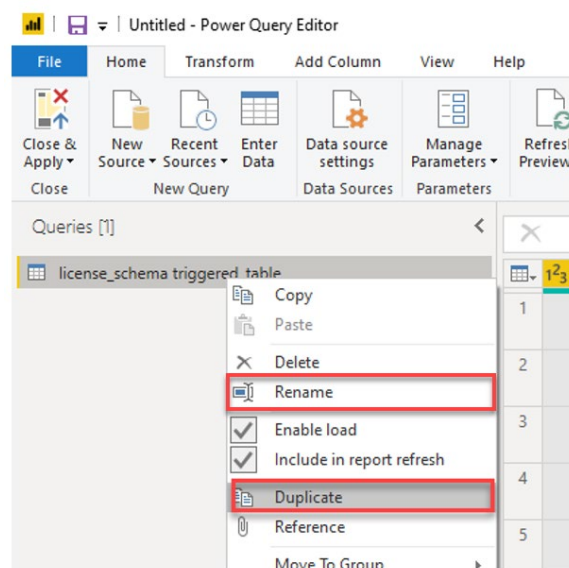
<< Insert

[Learn about Power BI Desktop formulas](#)

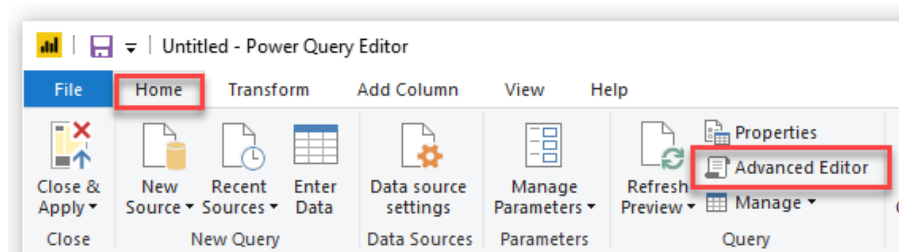
✓ No syntax errors have been detected.

OK Cancel

In the Power Query Interface, duplicate the “license_schema triggered_table” four times and rename it to “License Usage-Costing”, “License Usage-Historic”, “LicenseType-Current” and “Total Licenses”, “Users”, “Total Licenses” and “Software Names” accordingly.

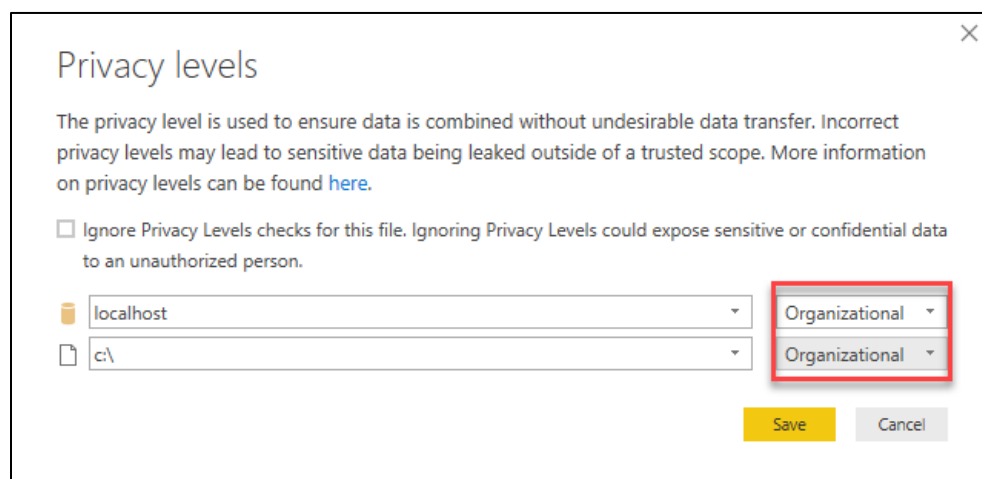
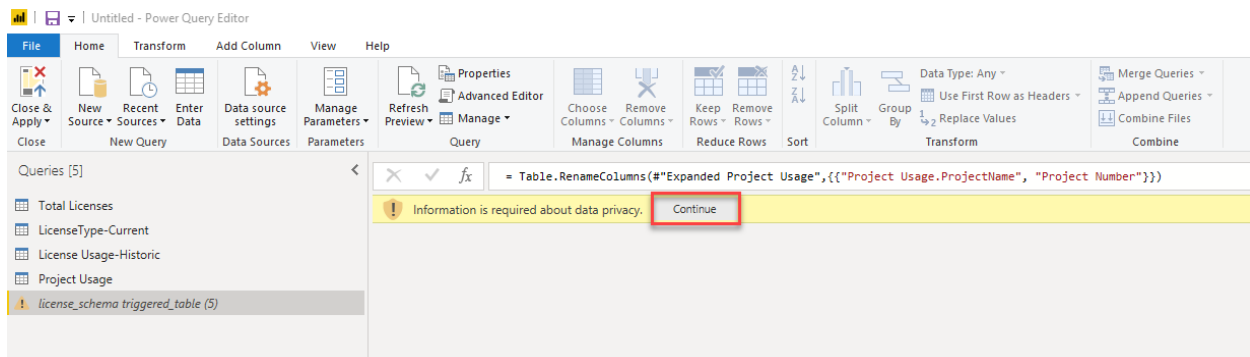


For every one of those tables, use the “Advanced Editor” and replace all existing code by the code provided in the “Power Query” subfolder. There are two “Power Query” subfolders, one for those who started from Dataset No 1 and one for those who started from Dataset No 2.

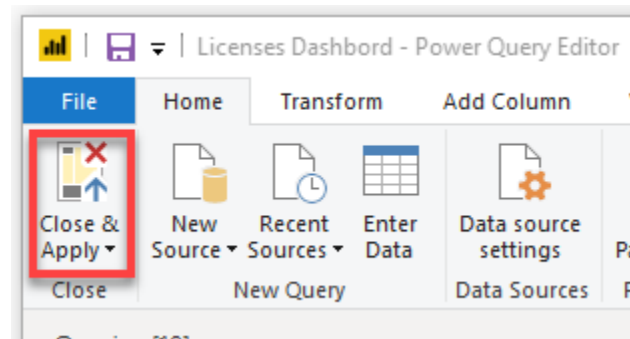


If started using the “Dataset No-2” use the advanced editor and keep the three first lines of code (up to the line starting by #“Promoted Headers”). Then, replace the rest of the code with every one of provided code at the “Data set No – 2” “Power Query” subfolder respectively.

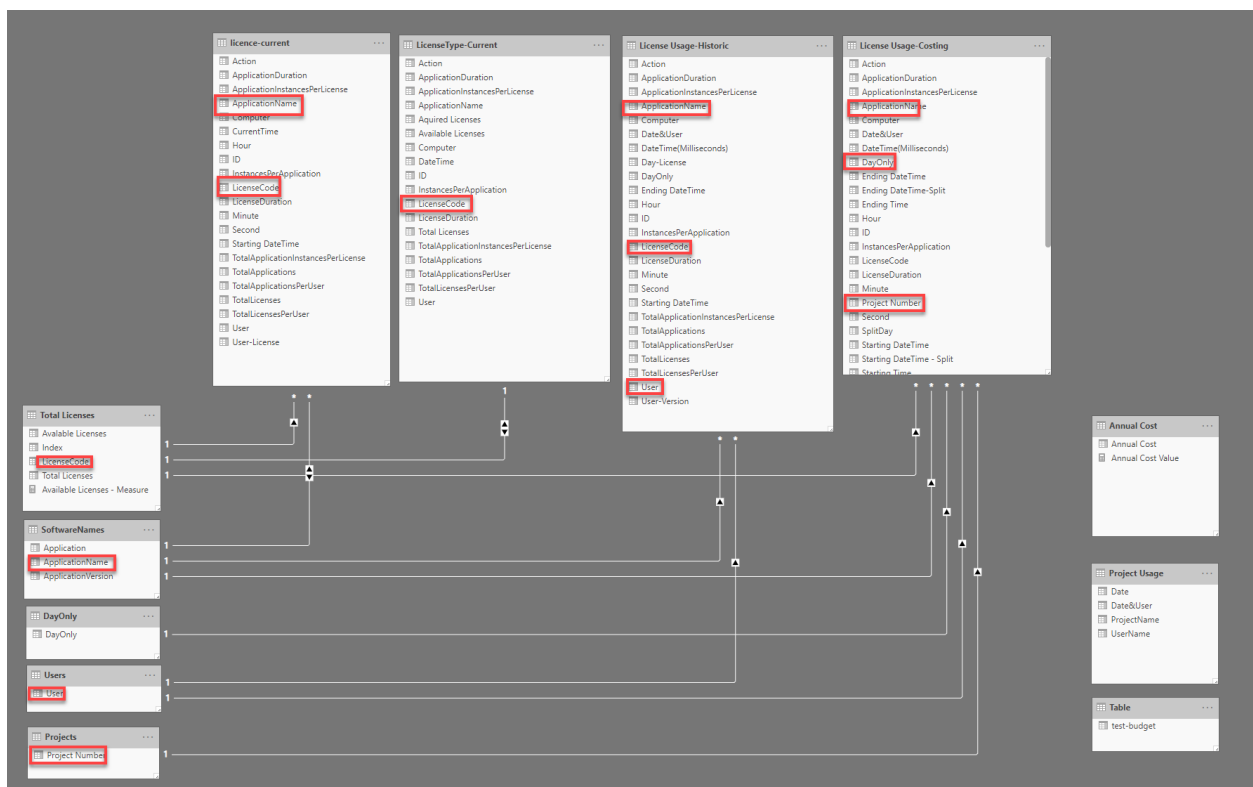
Select the “License Usage-Costing” table and press continue. This will allow data to join between the two different tables. This is initially prevented by Power BI for security reasons. Select Organizational and save.



By clicking at the close and apply button, we apply the data changes we made at the Power BI report data and model interfaces.

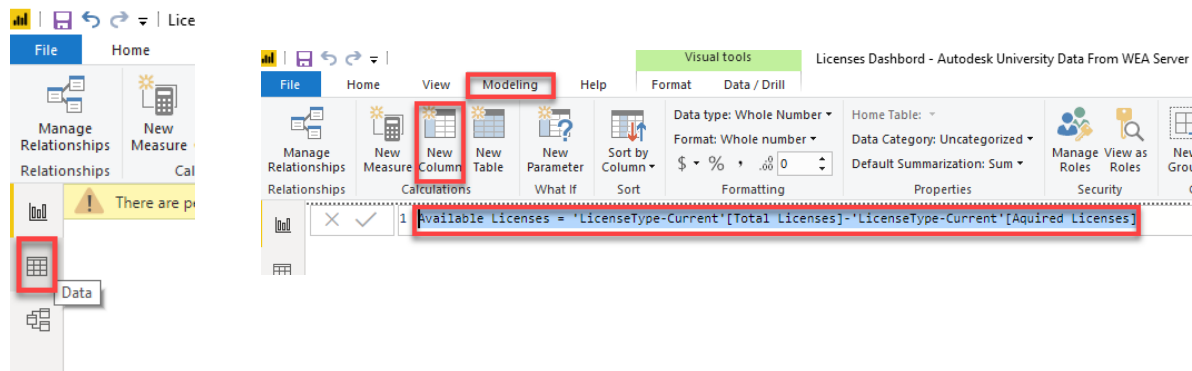


Using the Model Interface, we create the data relationships between tables. Drag and drop the same columns between different table to create the connections. Those data connections will help us to interact easily with the Power BI reports (visualizations).



Using the data interface and selecting table “LicenseType-Current” we create a new column to calculate the license availability.

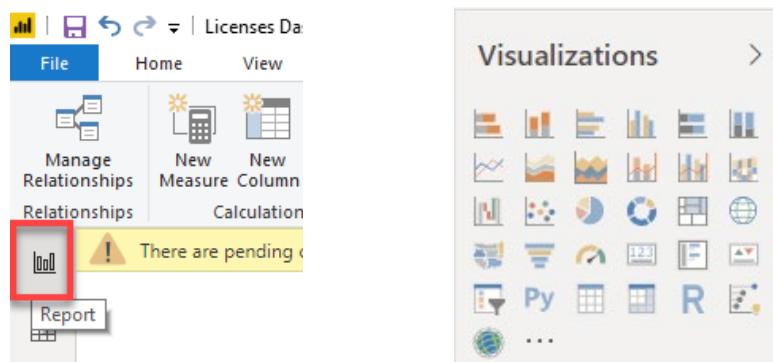
```
Available Licenses = 'LicenseType-Current'[Total Licenses]-'LicenseType-Current'[Acquired Licenses]
```



5.3 License Dashboard Analysis (Sample Data No.3)

Dataset No 3 includes elements existing at this section.

The workflow presented in this class will enable you to create the following dashboards to analyze the license consumption in your company. As the license data are organized, you will be able to create dashboards in Power BI using the report Interface (Visualizations). Using the Visualization buttons, we first select the type of graph we would like to create and we then from the Fields panel we drag and drop the data we would like to visualize.



Below all graph’s visualizations are extensively illustrated, together with the data needed to build the interactive dashboards.

Graphs:

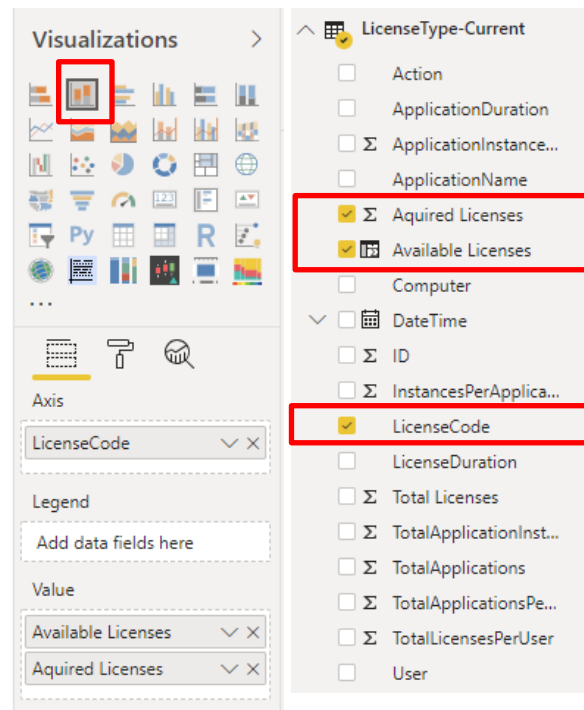
- Current license usage

Objective:

Identify the acquired, available and the total number of licenses.

Visualization:

Stacked Column Chart



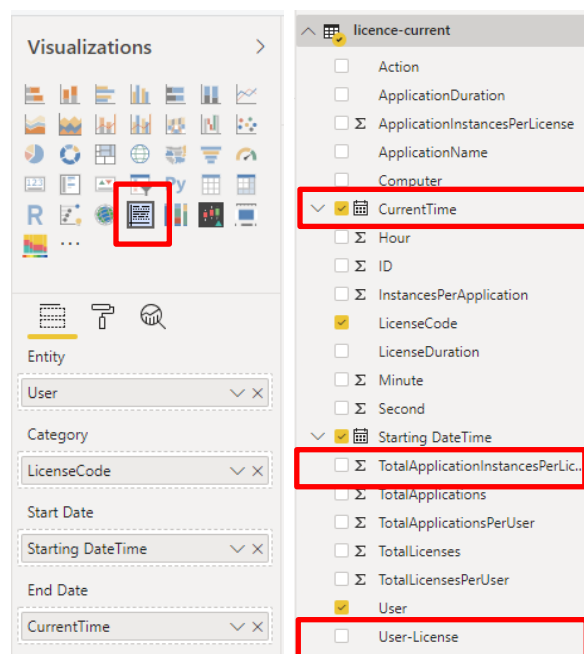
- Users Occupying a License

Objective:

Identify who is using a license and the duration user is using the license

Visualization:

As TimeLine
(download from Power BI marketplace)



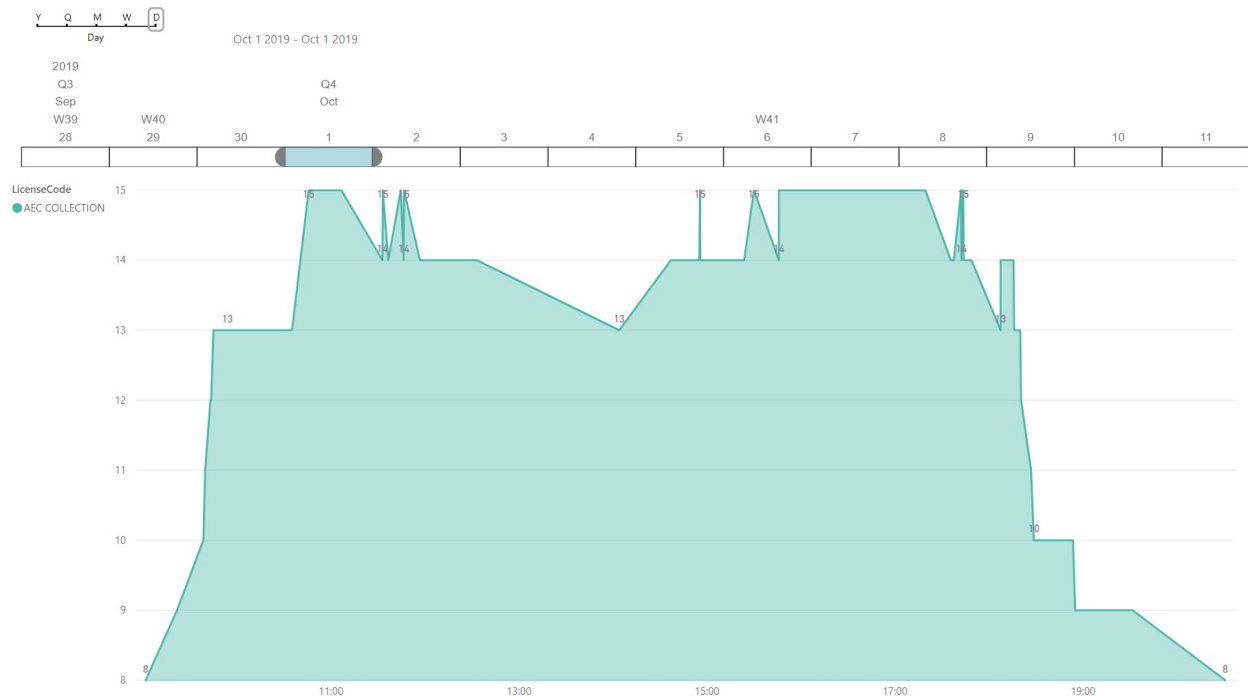
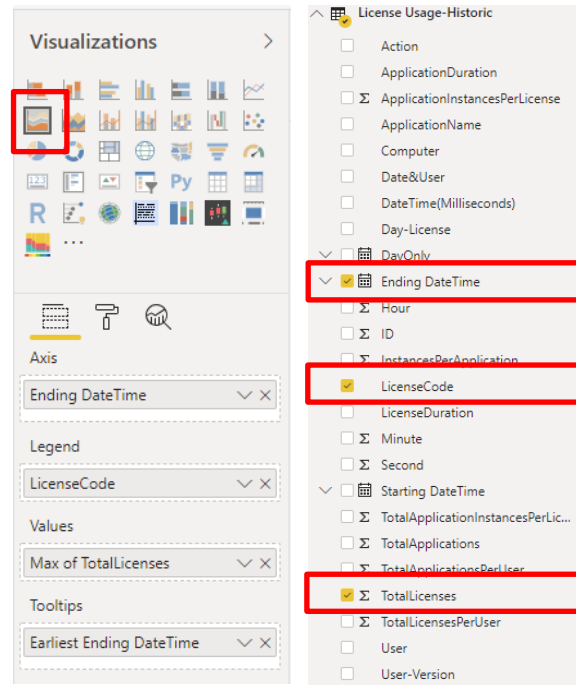
- Historical license usage

Objective:

Identify the maximum license usage in a time period that you can change in Power BI. You will be able to see the daily maximum and minimum utilization of licenses.

Visualization:

Area Chart



- Application Versions

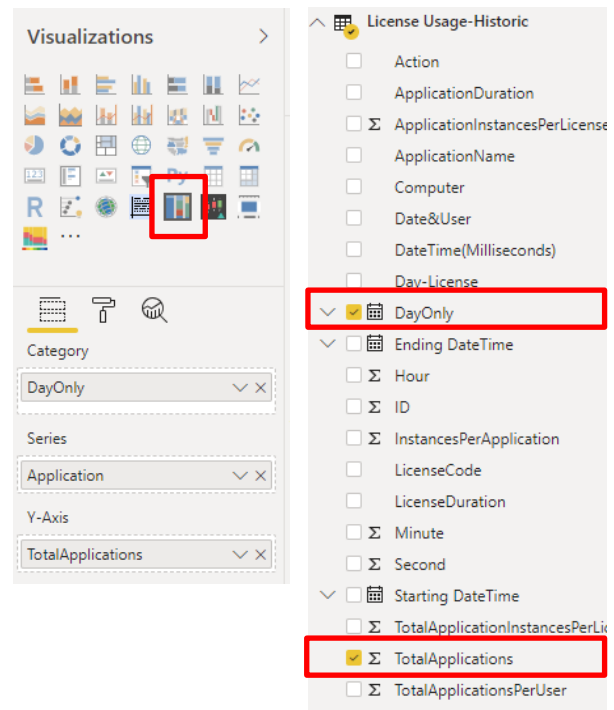
Objective:

Identify the most used software and Revit versions.

Visualization:

Mekko Chart 3.2.1

(download from Power BI marketplace)



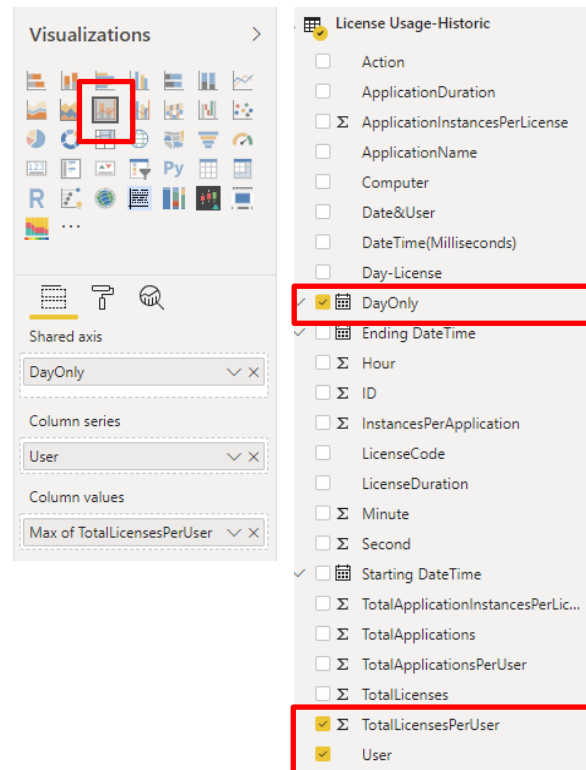
- Number of licenses per user

Objective:

Identify users that are using more than one license so you can review the license consumption.

Visualization:

Line Chart



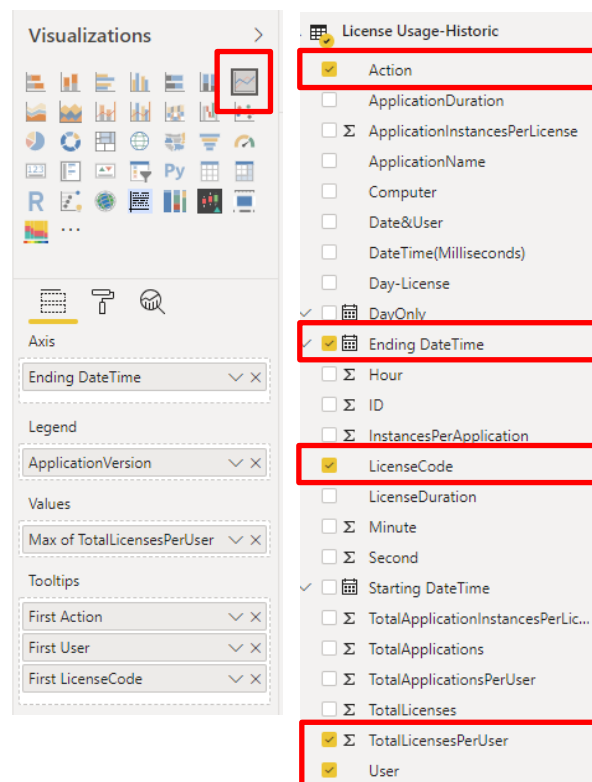
- License usage time per user

Objective:

Identify how long a user is using more than one license.

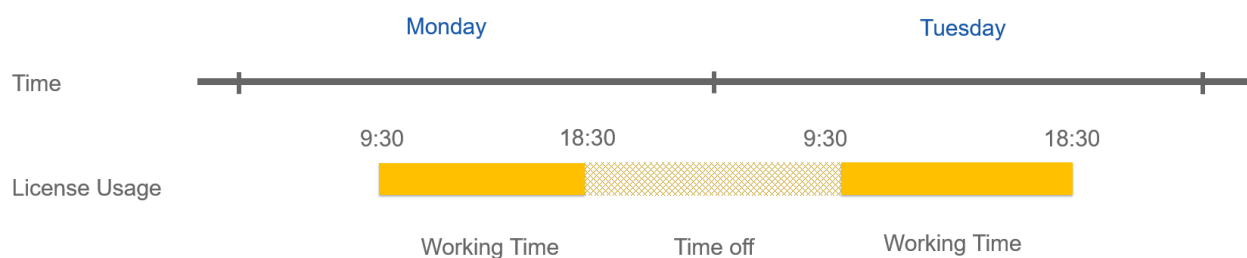
Visualization:

Line Chart



5.4 License Analytics and Project Usage

We will calculate the License usage per project, using the “License Usage-Costing” table previously created in Power Query. This table splits license usage into days. Considering that users’ working time is between 9:30 and 18:30, table includes values which exclude the time period earlier 9:30 in the morning and later that 18:30. Using the following graphs, we will visualise the license usage for every user and every project. We calculate the license time usage as a percentage across different projects and we calculate the license cost per project based on a license budget user input.



Graphs:

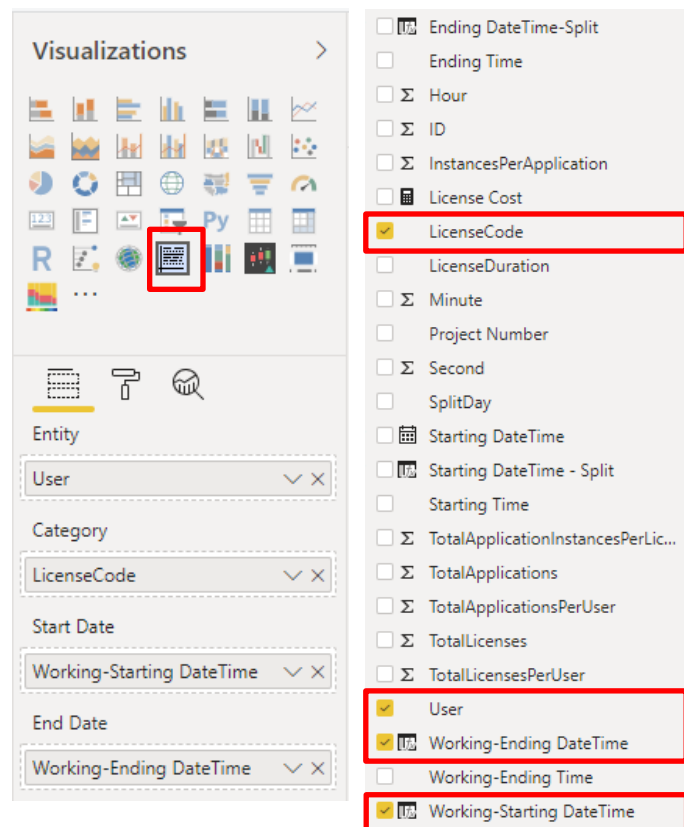
- Working Time - Usage per User

Objective:

Identify the duration of users acquired license during the Working Time period.

Visualization:

As TimeLine
(download from Power BI marketplace)



The screenshot shows the Power BI Visualizations pane for a chart titled 'Working Time - Usage per User'. The chart is configured as a 'Timeline' visualization. The 'Entity' is set to 'User', the 'Category' is 'LicenseCode', the 'Start Date' is 'Working-Starting DateTime', and the 'End Date' is 'Working-Ending DateTime'. The 'Visualizations' pane on the right shows the following fields selected (indicated by a red box):

- Ending DateTime-Split
- Ending Time
- Hour
- ID
- InstancesPerApplication
- License Cost
- LicenseCode
- LicenseDuration
- Minute
- Project Number
- Second
- SplitDay
- Starting DateTime
- Starting DateTime - Split
- Starting Time
- TotalApplicationInstancesPerLic...
- TotalApplications
- TotalApplicationsPerUser
- TotalLicenses
- TotalLicensesPerUser
- User
- Working-Ending DateTime
- Working-Ending Time
- Working-Starting DateTime

5.5 License Cost per Project

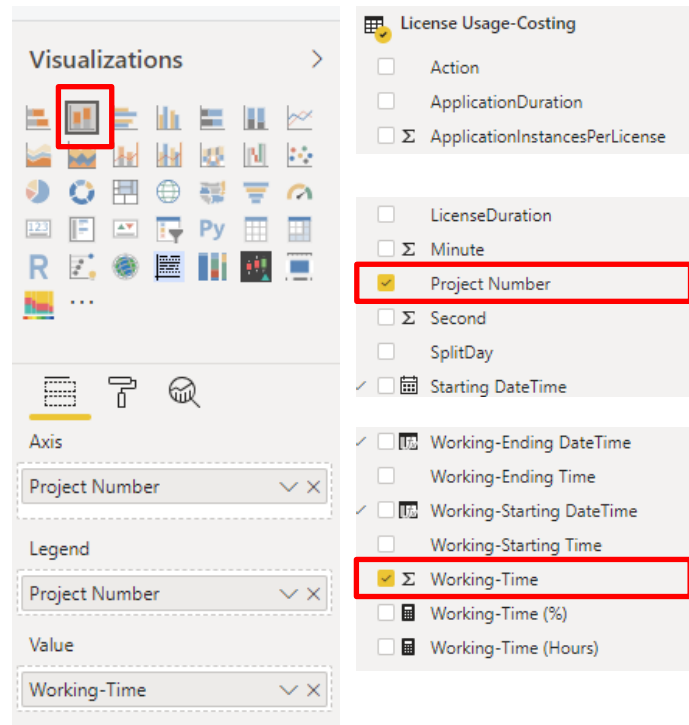
- License usage per project

Objective:

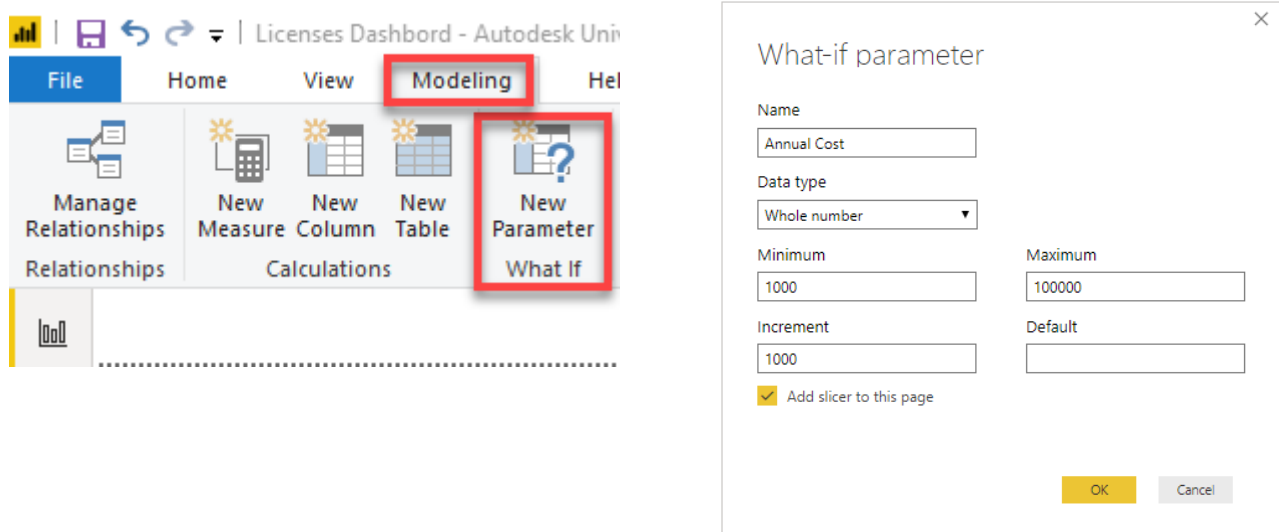
Identify and quantify the time of licenses used in the projects.

Visualization:

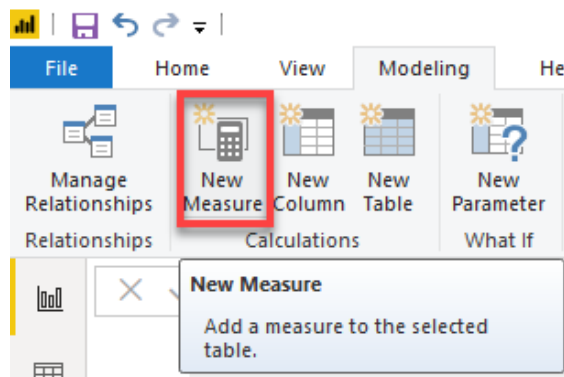
Stacked Column Chart



In order to calculate the License Cost per Project, we need to manually input the License Budget per License Package for a defined time period. Use the “New-Parameter” to create a user-defined value in Power BI report interface. Call this parameter “Annual Cost” and define it's Data type, Minimum, Maximum and Increment values to 1000, 100.000 and 1000 accordingly.



Select the “License Usage-Costing” table and create the following three DAX Measures “Working-Time (Hours)”, “Working-Time (%)” and “License Cost” using the “New Measure” command. The “License Cost” measure will allow creating interactive graphs defining license budgets based on different timeframes on the historic project license usage. You can also use this measure to check a future license project cost based on historical project usage and license usage data.



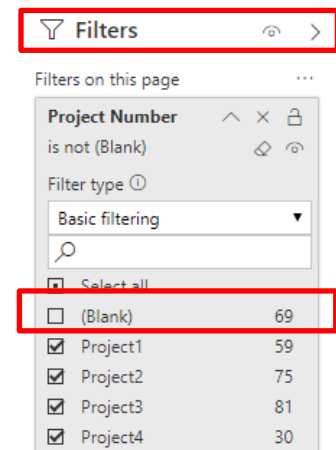
```
Working-Time (Hours) = FORMAT((SUM('License Usage-Costing'[Working-Time])/3600)/24, "dd:HH:mm:ss")
```

```
Working-Time (%) = SUM('License Usage-Costing'[Working-Time])/CALCULATE(SUM('License Usage-Costing'[Working-Time]),ALLSELECTED())
```

```
License Cost = [Working-Time (%)] * 'Annual Cost'[Annual Cost Value]
```

If users don't synchronize, the system will not assign a specific project. Therefore, a “Blank” Project Number value will be created.

We exclude those users from the next graphs by clicking at the page background and by creating a page filter unselecting all values which report a “Blank” Project.



Graphs

- Time Period

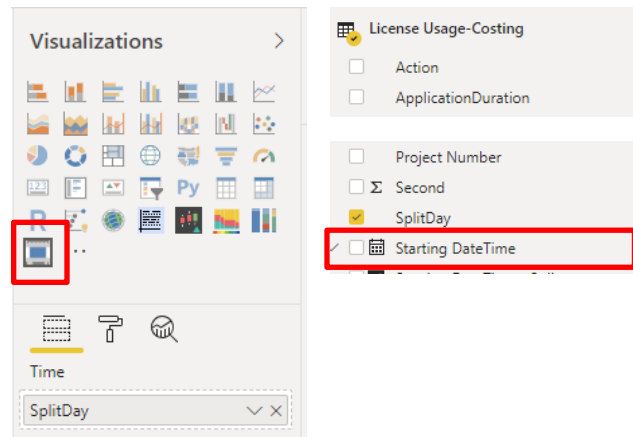
Objective:

To let the PowerBI user to define the Time Period for the License Cost

Visualization:

Timeline 2.1.1

(download from Power BI marketplace)



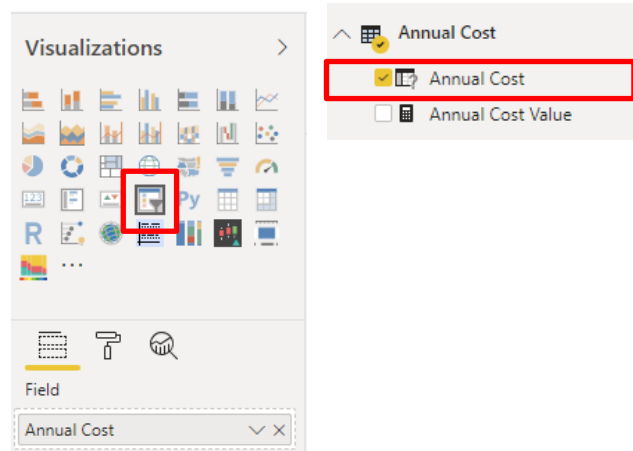
- Budget Slider User Input

Objective:

To let the user define the licensing budget spend for the specified time period

Visualization:

Slicer



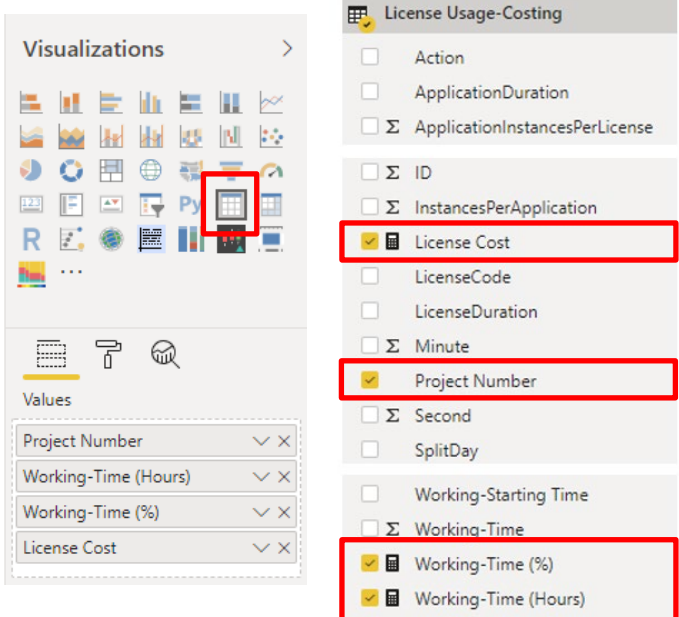
- Calculating Cost Per Project

Objective:

Calculate the license cost spent on every project according to the total user's working time – license usage

Visualization:

Table



5.6 Identify when extra licenses are required

Collecting data of license usage for several months, we will use Power BI forecasting capabilities to predict future license usage. For every week from the historical data, we will base our analysis on the weekly users to predict the license demand for the following week.

Power BI can use time sequence values to create a forecast on a line chart using the analytics tab under the visualization panel. However, Power BI built-in forecast method is based on a single input parameter prediction.

To create a prediction that improves accuracy and relevance to the current and the following week between the users and the maximum used licenses, we will use a more advanced prediction methodology which allows using multiple parameters as inputs for our predictions.

For this purpose, we will create a neural network prediction model using the scripting capabilities of the R programming language together with the “Neuralnet” library. This model will be trained across historical data based on the unique number of users and the licenses weekly maximum data.

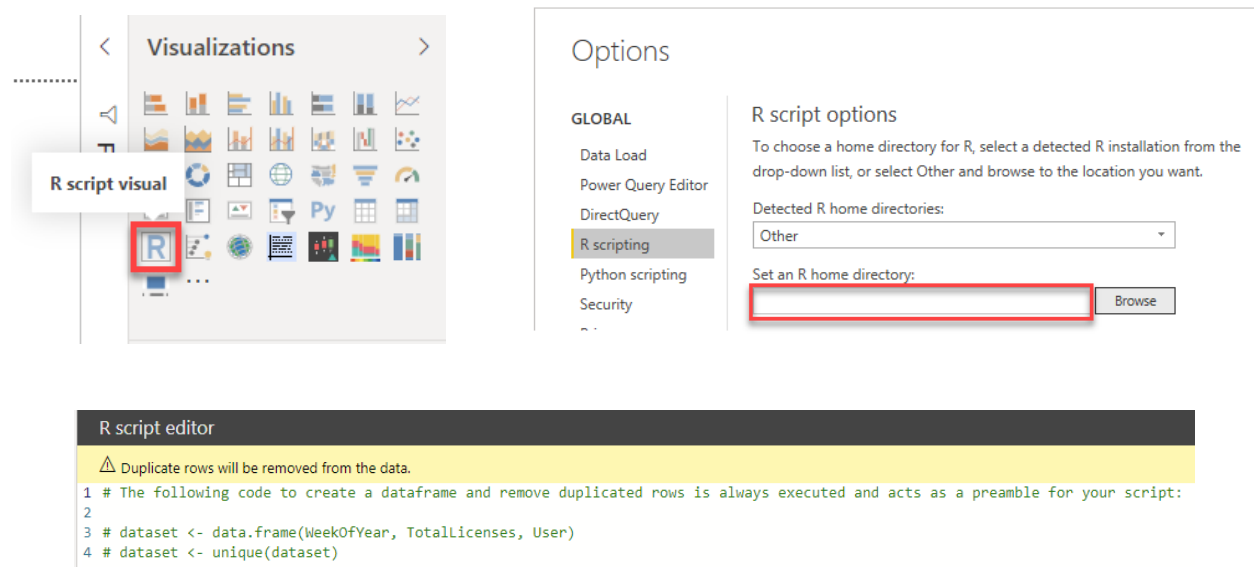
The Model will be evaluated for its accuracy and then it will be applied in the current week data to predict licenses for the next week.

R Scripting Preparation

We follow the instructions and install the R programming language for Power BI.

<https://docs.microsoft.com/en-us/power-bi/desktop-r-visuals>

Then we use the “R Script visual” under the visualization palette to create a new R script. At the bottom of the page copy-paste the given script.



Visualizations

R script visual

Options

GLOBAL

- Data Load
- Power Query Editor
- DirectQuery
- R scripting**
- Python scripting
- Security

R script options

To choose a home directory for R, select a detected R installation from the drop-down list, or select Other and browse to the location you want.

Detected R home directories:

Other

Set an R home directory:

Browse

R script editor

⚠ Duplicate rows will be removed from the data.

```

1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:
2
3 # dataset <- data.frame(WeekOfYear, TotalLicenses, User)
4 # dataset <- unique(dataset)
5

```

To run the script, we need to install two R library packages using R command line. We navigate at the R Home directory we previously setup (i.e. C:\Program Files\Microsoft\R Open\R-3.5.3) and run the “R.exe” command line which exists at the “bin” subfolder. We install neuralnet and ggplot2 libraries typing the following commands at the R command line.

```
install.packages("neuralnet")
install.packages("ggplot2")
```

Prediction Timeframe

In Power Query interface, at the “License Usage-Historic” table, we create an additional custom column to aggregate the user and license usage.

Custom Column

Add a column that is computed from the other columns.

New column name

WeekOfYear

Custom column formula ⓘ

= Date.WeekOfYear(DateTime.Date([Ending DateTime]))

Due to lack of data provided in the sample dataset, for the purpose of this exercise instead of the WeekOfYear we can use the DayOnly Value to make estimations for the next day. The principle of the following script and the details provided in this section could work in any time frame.

We create the R script using the historical user and license weekly values to be added as input and outputs data for our prediction model. For the purpose of this exercise, we will use the historical user and license daily values.

Graphs

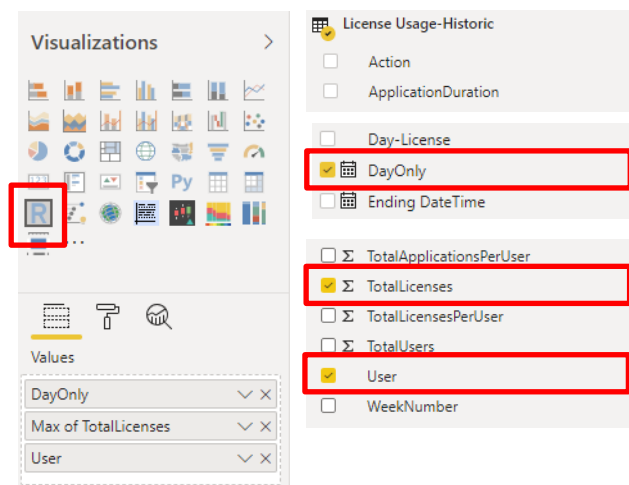
- Predict maximum licenses demand of next day

Objective:

Running the predictive model to predict the required future licenses.

Visualization:

R Script Visual



Neural Network Training / Evaluating and Running Process

Create the “R Neural Network Script” by copying and pasting it from the “Predictive Modeling” subfolder to Power BI R visual. The predictive modeling procedure has several steps.

Initially, script prepares the historical data to input and output data and then it splits them into two groups, the training and the testing sets of data. The model is trained using the training set.

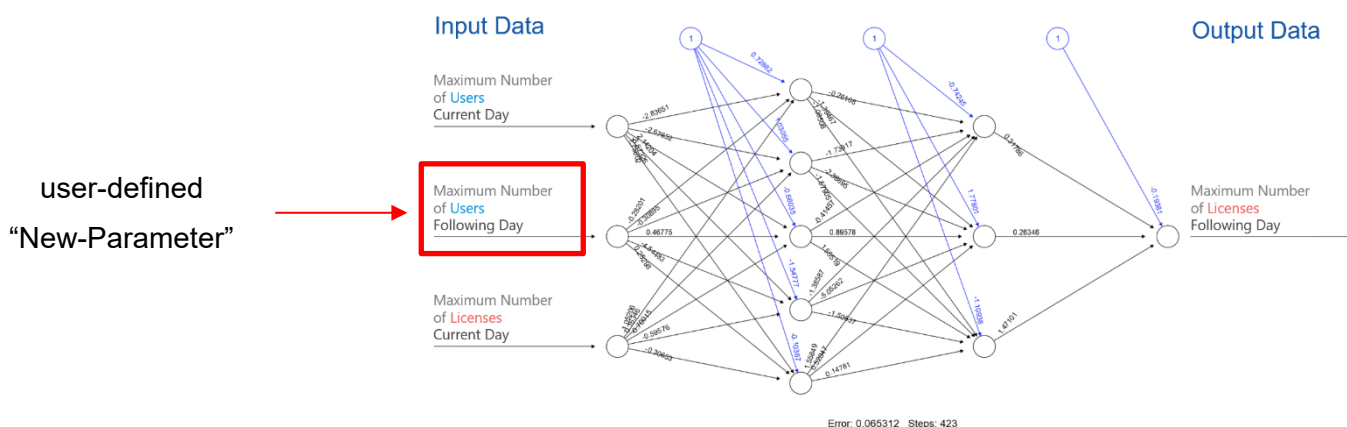
When the neural network training finishes, the model uses the input values both from the training and testing set of data to make a prediction and compares the results (output) to the historical values so it can be evaluated.

The Root Mean Square Error factor is calculated for both training and testing sets. This factor is very important as it reveals the training capability of the network and provides evidence which measures the confidence of our prediction.

If RMSE is high, this could be improved by providing more data to our model, increasing the training time by decreasing the threshold parameter or increasing the neurons of the network by increasing the hidden parameter.

If the RMSE between training and testing sets are different, this could be due to model overfitting. This means that our model is not able to generalize from a trend. Our training needs more data or training procedure needs to stop earlier on.

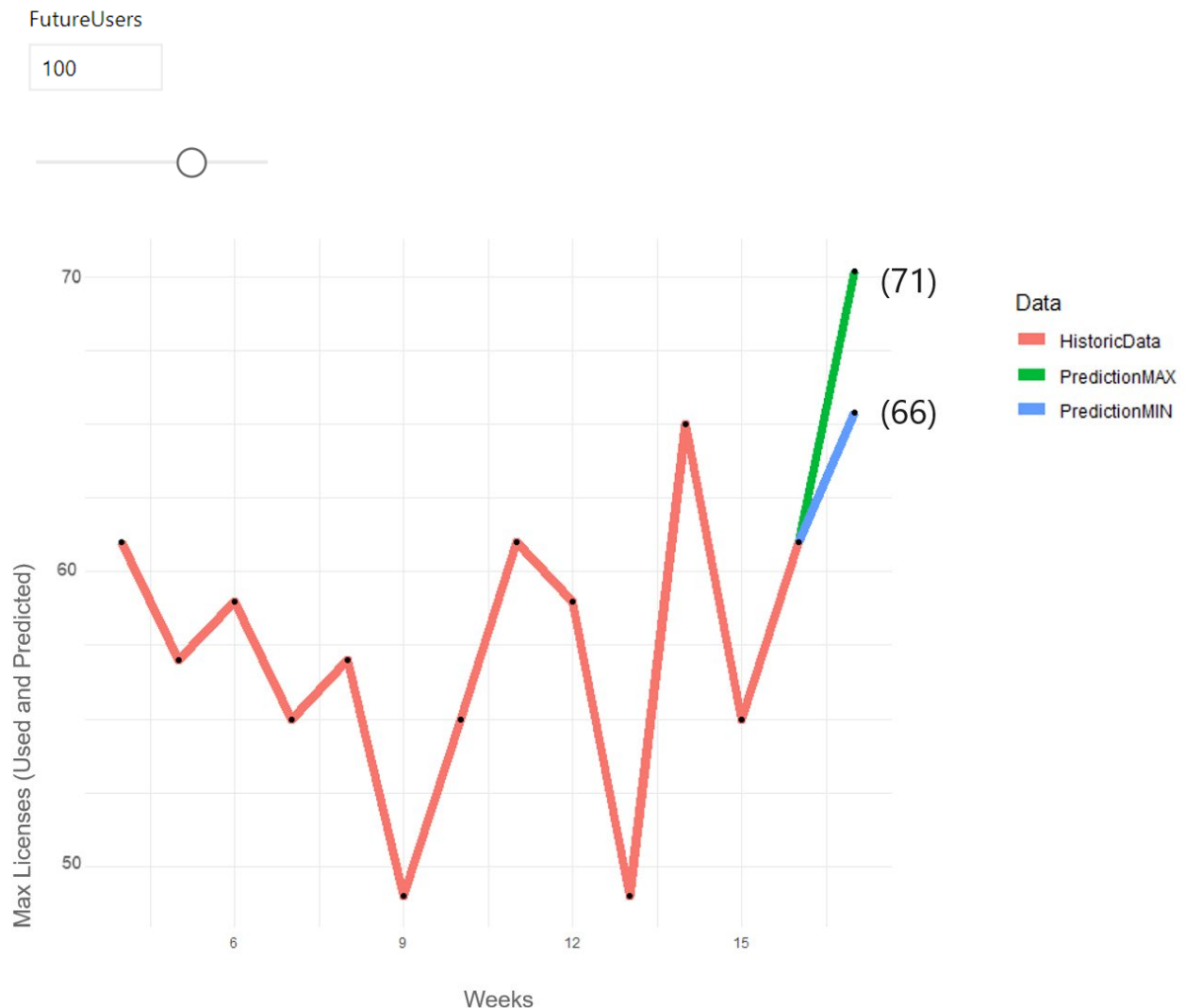
To improve accuracy, our methodology includes a Power BI “New-Parameter” to be used as an input value. This is a user-defined value that provides the number of users of the following week (or day) as an additional input value.



In the last step of the script, model predicts the future usage based on the today's users and current maximum license consumption. We also run 10 different neural networks and calculate the minimum and maximum prediction value.

Visualizing a predictive model

The following graph shows the number of licenses during a period of few weeks in red. It also illustrates that, on the current week (16th week), the model predicts that given the 100 users as a figure for next week, we will require to provide 66-71 Max licenses (green and blue lines) based on our historical demand. The range between Prediction Max and Prediction Min provides us the confidence of the prediction. The closer those two lines are to each other, neural network prediction is more probable to be accurate.



Conclusion

This session demonstrated the importance of license insight tools for business management and the multiple benefits of license analysis.

Our workflow depends on the process of decoding the Flex License Data to be able to structure data that are subsequently processed in the MySQL database.

During the presentation, you learned how to install and prepare the open source database MySQL to receive and process data.

We also showed the process of importing the data to create analysis and dashboards in the PowerBI.

The following key points summarise the learning achievements :

- License insight tools are very important for businesses.
- FlexLM Log file and the Revit Synchronization are providing enough data to create the analysis.
- Clear benefits to collect data using MySQL which is a robust solution.
- License usage and overview dashboards to identify the demand.
- Proactive monitoring of licenses requirements.

Dataset:

The most update code and dataset are available on GitHub:
<https://github.com/WilkinsonEyre/License-Management>

Disclaimer:

This presentation and datasets are subject to change as the technology and license management evolves.

The user of this presentation and datasets accepts this risk of using the presentation content. No liability is implied for all content of this presentation and how it is subsequently utilized.



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.