Emailmanager User's Guide

by Ferdi Ploeger

Version 1.0

July 8, 2020

Contents

1 Introduction

EmailManager organizes emails that are transferred from the server of the internet service provider (ISP) to local media as local hard disks or CDRoms. Thus the emails can be kept for years rather than that they are deleted either by the ISP because the mailbox is full, the ISP goes bust or by the user wanting to delete emails considering they are of no longer use.

By deleting emails you are deleting your history. Emails replaced letters as they were exchanged in earlier times. Collected letters of friends, lovers or relations preserve your history that you (or someone else) might want to recall later.

But there might be more pressing needs to keep your emails: Amazon that claims you to be a prime member at a monthly cost although you could prove you stopped the membership, British Gas that claims you didn't answer their claims you had to pay some outstanding bills or the Home Office that wants to have 'proof' of having lived in the UK some 15 years ago.

EmailManager keeps the emails as sent to you or sent by you. This is including the images and the attachments. When you display the preserved emails from the near past they look exactly as they looked in your mail-browser. You even can 'reply' or 'forward' them as if they were still on the server of your ISP.

Unfortunately (by now) the images will disappear over the years as they are referred to images stored on the server of the ISP and will be deleted over time. But the text, addresses and attachments all remain.

2 Installation

For EmailManager to work it needs to have three files and a subfolder together in the same folder. This folder can be on a local disk , an external disk or in a cloud. The executable is EmailManager.exe[1].2.1:

```
EmailManager.exe
EmailManager.ui
EmailManager.ico
```

The subfolder is EmailManagerHelp-folder with the files:

```
UserGuide.pdf
SQL-examples.pdf
SQLite Tutorial.pdf
About.pdf
```

EmailManager will run in Windows by a double-click on the .exe-file or a shortcut of the .exe-file (i.e. a shortcut placed on the desktop or the status bar of Windows). EmailManager doesn't need any more 'installation' rather then just put these files in a folder and arrange a shortcut to invoke EmailManager.exe.

Nevertheless it comes handy as there is a default program installed that opens emails (.eml-files). Also handy, but strictly not needed, is a program that can handle .zip-files and a program that can handle a SQLite-database such as SQLiteStudio or DB Browser for SQLite (all free).

When EmailManager runs for the first time it will create a folder 'C:/EmailManager'. In this folder it will write and maintain the two files:

```
PythonEmailManager.log
EmailManager.json
```

Every time EmailManager starts it will read the file EmailManager.json to find out what the 'default-folder' is. EmailManager keeps an ongoing log of its activities in PythonEmailManager.log.

These two files should be left untouched by means other than EmailManager itself. Deleting the files or changing its names results in new files being made with the above names. The content of both the .log and .json files can be seen and even changed

[1] EmailManager.ui defines the layout of EmailManager and EmailManager.ico is the icon used by the program

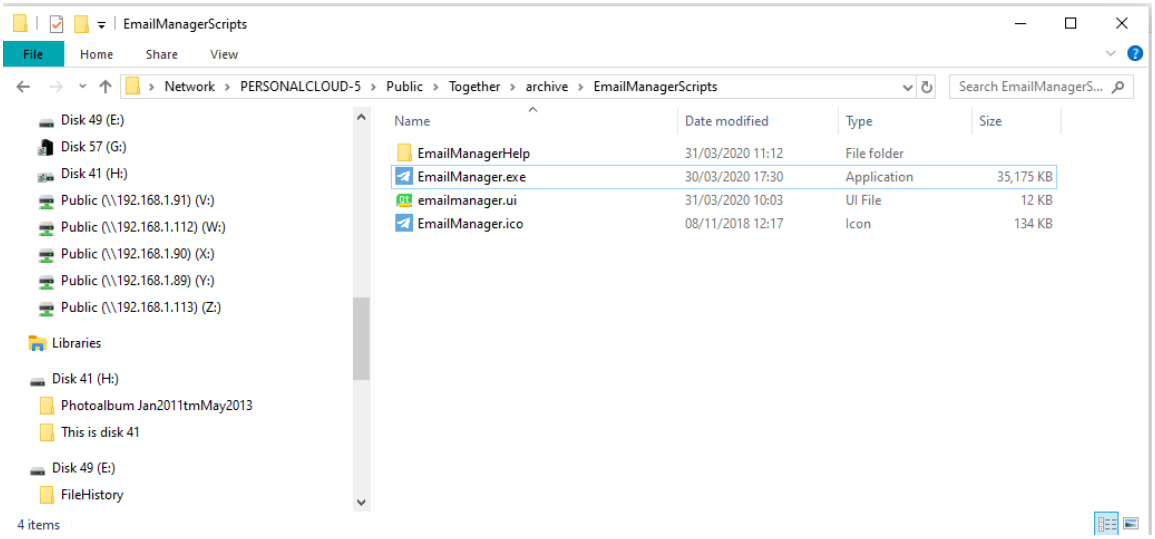


Figure 2.1: Installed EmailManager - files

(don't do this unless you know very well what you are doing) in a simple application as Windows Notepad, its working will be destroyed when viewed in Windows WordPad or another text editor like LibreOffice and accidentally saved.

2.1 Default Folder

EmailManager.log keeps track of the activities of EmailManager. EmailManager.json remembers where the 'default-folder' is you put and keep your emails, your Emails.zip and Emails.db. The default folder is the folder (anywhere) where you save your 'raw' emails (.eml-files). These .eml-files are then processed[2] by EmailManager. When you perform a query on the database EmailManager will use the database EmailManager.db in the default-folder. When you want to display an email EmailManager will take the email from EmailManager.zip in the default folder. As EmailManager deals with certain files only it leaves other files untouched so you can keep other files in this default folder as well.

The files EmailManager deals with in the default folder are:

[2]Renaming goes in two steps: first all the .eml-files are renamed to TEMPFILE-xx-yy.eml, where xx is the number of restarts and yy is the number, starting from the last archived FILEyy (in Emails.zip and Emails.db). When all the .eml-files are thus renamed the second rename-action renames all the TEMPFILE-xx-yy.eml's to FILEyy.eml's. The value of yy (the last archived filenr) is kept in the file LocalEmailManager.json and looks like: {"LastFileNr": 16037}. It is extremely important to leave this file untouched because if this file is not found a new file LocalEmailManager.json will be created with content {"LastFileNr": 0}. EmailManager will then rename new afdded files as FILE1.eml, FILE2.eml and subsequent overwrite files with the same name in EmailManager.zip and EmailMananger.db

```
.eml-files
Emails.db
Emails.zip
localEmailManager.json
```

The .eml-files are the 'raw' unprocessed emails as saved from your emailprogram. Emails.db is the database which keeps the details of the emails including the new name given to the email as it is stored in Emails.zip. localEmailManager.json keeps the number of the last processed email[3]

2.2 Different Default Folders

It is possible to maintain more than one 'default folder'. In case you want to work with a different 'default folder' you have to let EmailManager know which 'default folder' you want EmailManager to deal with. This can be done with the menu-option 'Options/Set default folder'. Until you change this 'default folder' again EmailManager will use this new folder as the default folder, even after you restart the program. EmailManager will create new files as given in section 2.1.

.

[3]Before they are zipped Emails are given the name FILExxx.eml, where xxx is an increasing number. After being zipped the original emails are deleted.

3 Organizing emails

When using EmailManager one should choose a 'default-folder' (from the menu: 'Options | Set default folder'. This folder should contain at least one .eml-file to start with.

After choosing the above option from the menu the user will be confronted with an OpenFileDialog that will see .eml-files only. Initially when EmailManager is used for the first time, this will be the 'C:/'-folder. The user then should navigate to a folder where he/she intends to keep its emails. This folder might be on an internal or external disk with a complicated path. By clicking on any .eml-file in this folder this folder will be set as the default folder. When confronted with OpenFileDialogs from other menu-options EmailManager will then always open in this folder. (This folder-path will be preserved in 'C:/EmailManager/EmailManager.json).

The aim of EmailManager is that all .eml-files to archive should be dumped into this default folder. Once you have written one or more emails to this folder from you email-program your email program will suggest this folder in subsequent 'save-as' commands.

Archiving of emails happens in EmailManager.zip created and maintained in the default folder. For each email the columns are (see fig 3.1):

```
date
time
timezone
sender
recipient
subject
cc's
bcc's
spam-indication
size (in Mb)
FileName
```

will be written in the Emails.db database. In the same database a list of all the email-addresses that occurred in the sender, recipients, cc's and bcc'c will be preserved (see fig 3.2) with one relevant column address. (the column 'count' is a dummy so is irrelevant).

Initially the emails will be saved by your email program as <subject>.eml. EmailManager will rename these filenames to FILE1.eml, FILE2.eml, ...

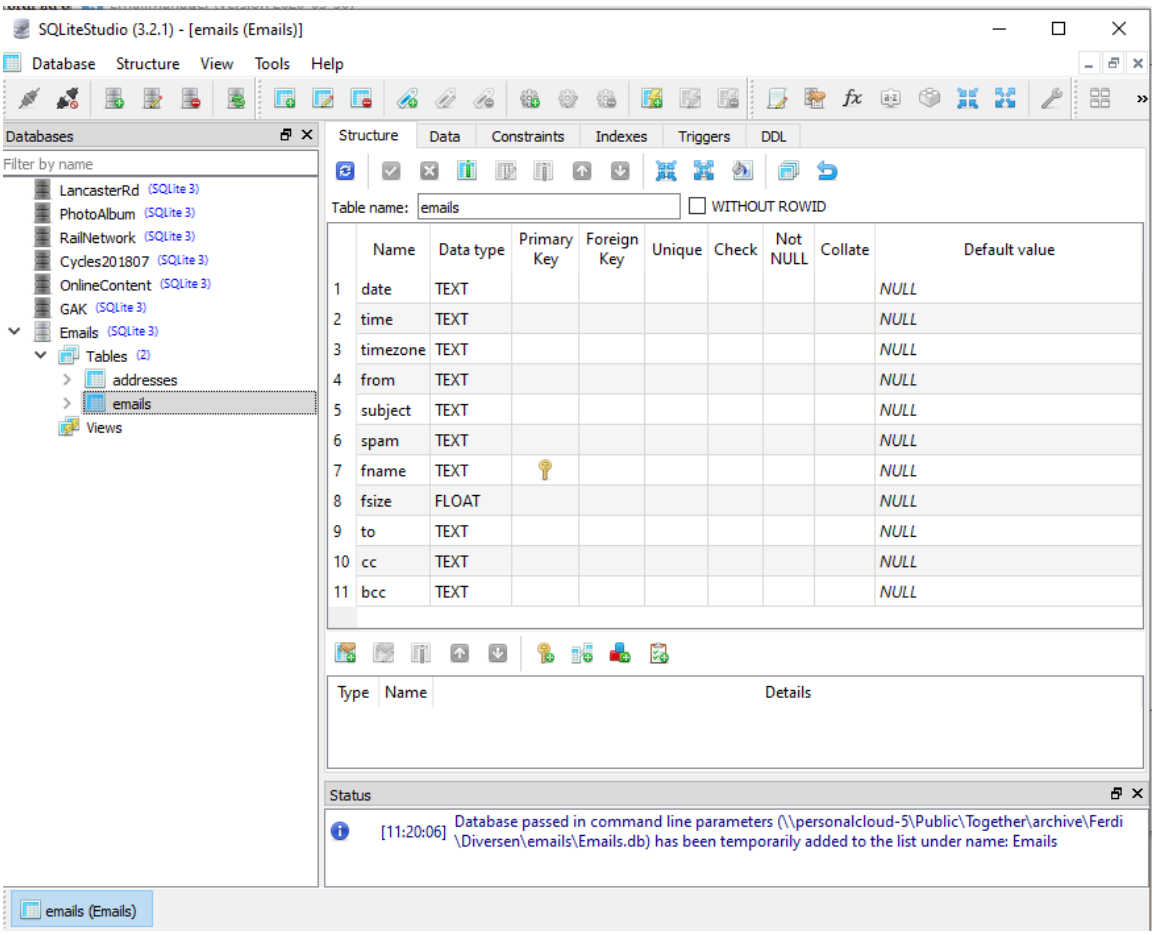


Figure 3.1: Columns in SQLite Studio

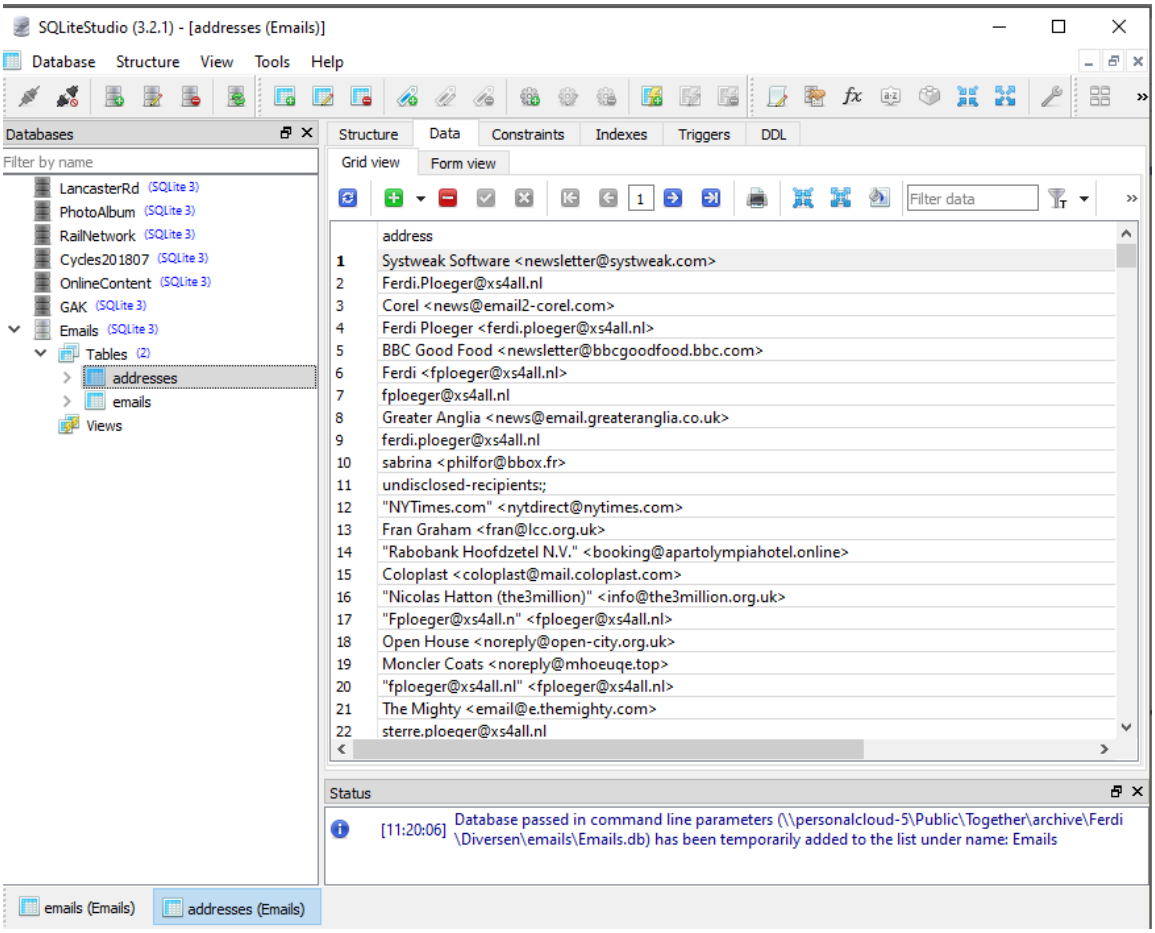


Figure 3.2: table addresses kept in Emails.db

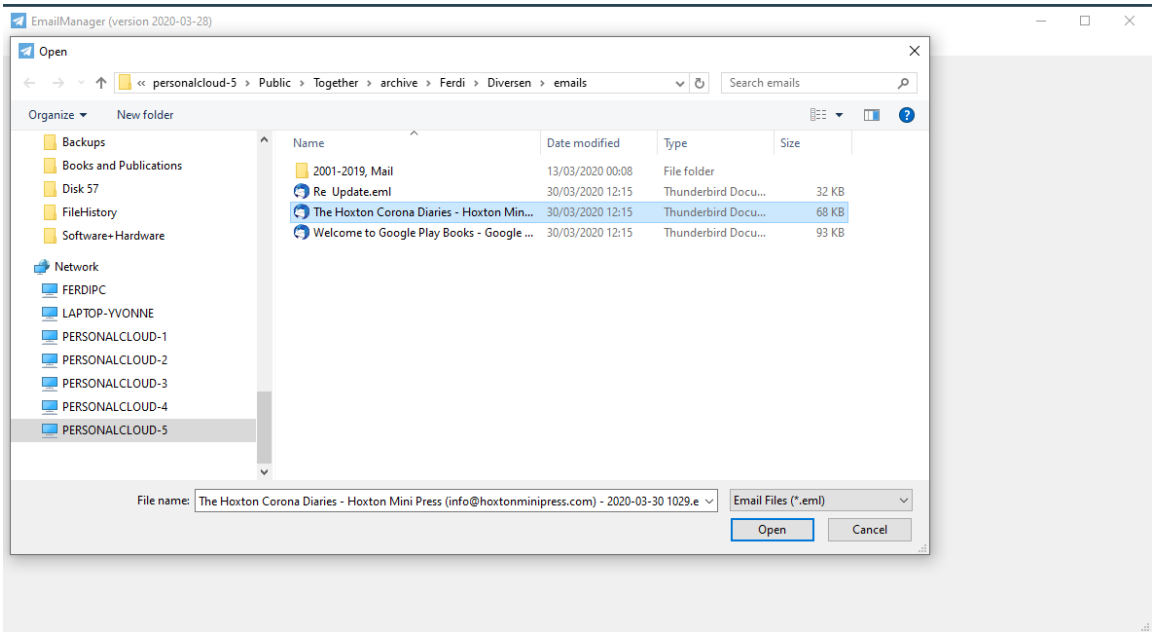


Figure 3.3: start archiving

EmailManager keeps the name of the last renamed and archived email in the file:

`localEmailManager.json`

This file should be left untouched because otherwise EmailManager would overwrite existing emails.

To properly archive the emails as saved from your email program the user should use the menu-option 'Actions | Archive Emails'. EmailManager then will open in the default folder and show the emails written there. Any one of those emails should be clicked on followed by clicking 'Open'. Clicking on 'Cancel' cancels the activity (see fig 3.3).

EmailManager then will display what it is doing (see fig 3.4) and keeps track of its activities in its Log File (Options | Open LogFile).

3.1 Warning and reassurances

In the 'default-folder' other files can be kept as .eml-files only. This enables the user i.e. to keep scans of letters sent by ordinary mail or other considered relevant files. EmailManager will archive .eml-files only and leave non-eml-files untouched.

Be aware that EmailManager will also find, archive and delete .eml-files in sub folders of the default folder. This makes it possible to merge emails some of which might have the same name. As long as batches of emails kept in different subfolders of the default-folder the renaming of these emails goes without problems. After renaming the

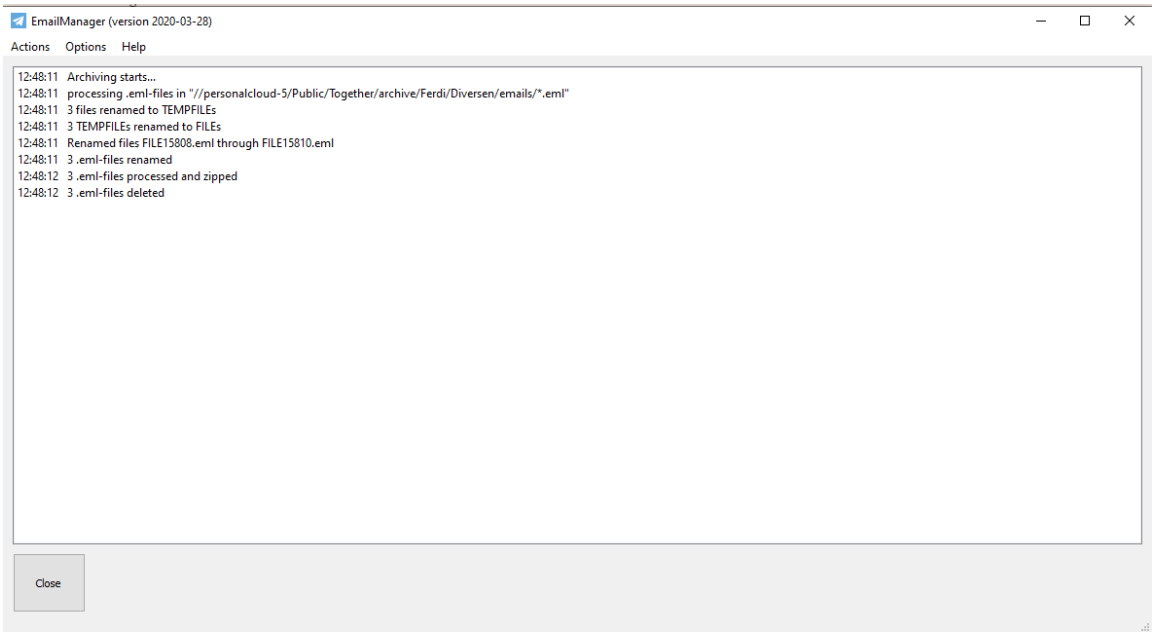


Figure 3.4: finished archiving

subfolders are still there but are empty and the emails that were in these subfolders are now amongst the renamed emails (in Emails.zip and Emails.db)

The EmailManager program and its associated files can be stored anywhere on remote or local harddisks. EmailManager will work fine and fast (enough) on a very remote defaultfolder for say up to batches of 500 emails. EmailManager can manage hundreds of thousands emails but, when offered a large amount of emails it works a factor 10 faster when it works on a defaultfolder that is somewhere on the local harddisk. After EmailManager processed all the thousands of emails al that remains is an Email.db, Emails.zip en LocalEmailManager.json-file (together with other files - that were not handled by EmailManager - stored by the user). Then just move this folder to a more convenient and remote place (i.e. a place that will be backupped in a regular scheme) and change the defaultfolder in EmailManager (Optiond | Change defaultfolder) as such.

4 Retrieving emails

The emails will be renamed to a simple filename (FILEn.eml, n=1, 2,..) and compressed in the file Emails.zip in the default folder. This Email.zip-file can be handles with zip-utilities as WinZip (not free) or 7Zip (free). The files in Emails.zip can be clicked on to open them in the default mail-program. From there these mails can be answered on or forwarded. However the filename of the emails has been made meaningless so unless you know which email you are looking for just clicking on an email in Emails.zip is a random action.

EmailManager offers several methods to find the email(s) you are looking for. You can look under the menu-option 'Get Emails' (see fig 4.1)

There are several pre-organised queries. You can do the query including or excluding spam-emails[1]. The condition used for a query without spam-emails is:

```
(where spam <> 'YES' or spam IS NULL)
```

4.1 Get emails from today

The menu-option 'Get Emails | from today will display a list of emails that are added to the EmailManager system today. The list is sorted on the timestamp of the emails and standard displays as shown in fig 4.2.

In case the way the table is shown doesn't satisfy, the columns can be adapted as shown in fig 4.3 just by dragging the column boundaries with the mouse.

Where a column-name 'fname' exists the email can be retrieved by clicking on a filename: this will retrieve the email from the Emails.zip-file, store it in the folder 'C:/EmailManagerResults' and display the email in the default email program (see fig 4.4).

The folder 'C:/EmailManagerResults' will be cleared before the results are displayed in the table on screen and all the files picked from the table on screen will be stored in this folder with the name it has been given by EmailManager before it was compressed

[1] 'spam' is a qualification assigned to the email by the provider, in this case xs4all. When the file is too bulky xs4all gives the spam the qualification 'Skipped, message too large', so this email will be included in the 'no spam' selection. Sometimes emails don't have a qualification 'spam', i.e. emails sent by you before they are seen by xs4all. In that case they appear in the result of the query as spam 'None'.

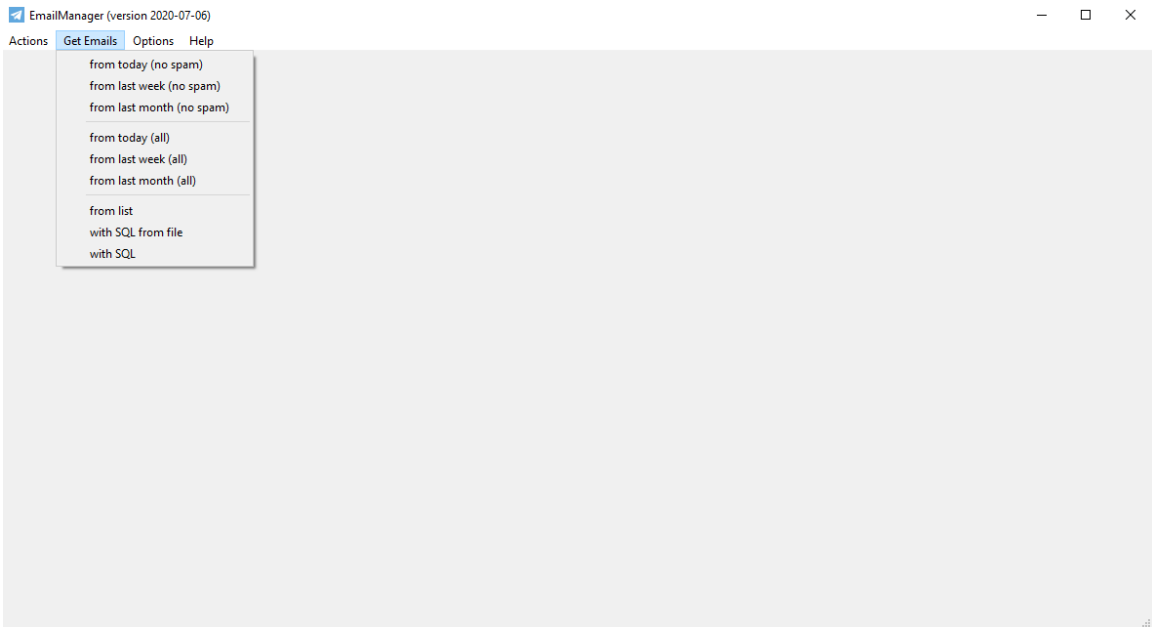


Figure 4.1: retrieving emails

in Emails.zip. These emails can be saved somewhere else at will or just be left to be deleted once another selection is made.

The retrieved emails stay in the file 'Emails.zip' and don't mingle with the saved data so the integrity of the EmailManager data is preserved.

The folder EmailManagerResults is used for the results of searches for 'Emails of today', Emails of last week' and 'Emails of last month'. The options 'Emails from list' and 'Emails with SQL' follow a different procedure: see there.

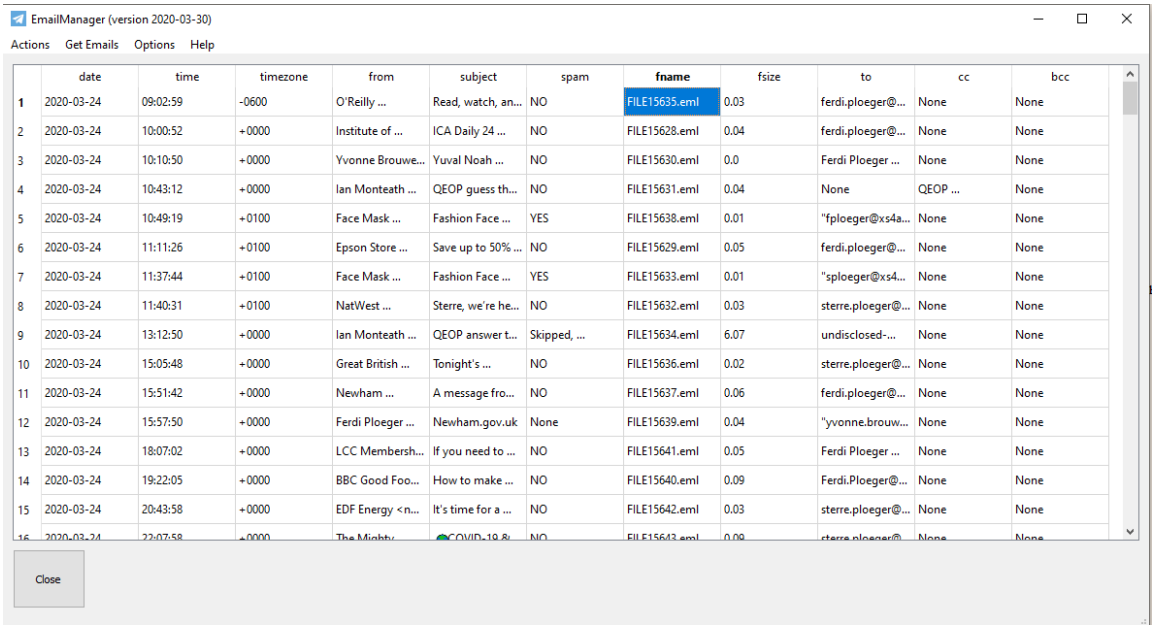
The email as displayed in fig 4.4 allows to reply on or to forward to other recipients.

4.2 Get emails from last week

The menu-option 'Get Emails | from last week' will display a list of emails added tot the EmailManager system during last week.The list is sorted on the date and timestamp of the emails and standard displays as shown in fig 4.2.

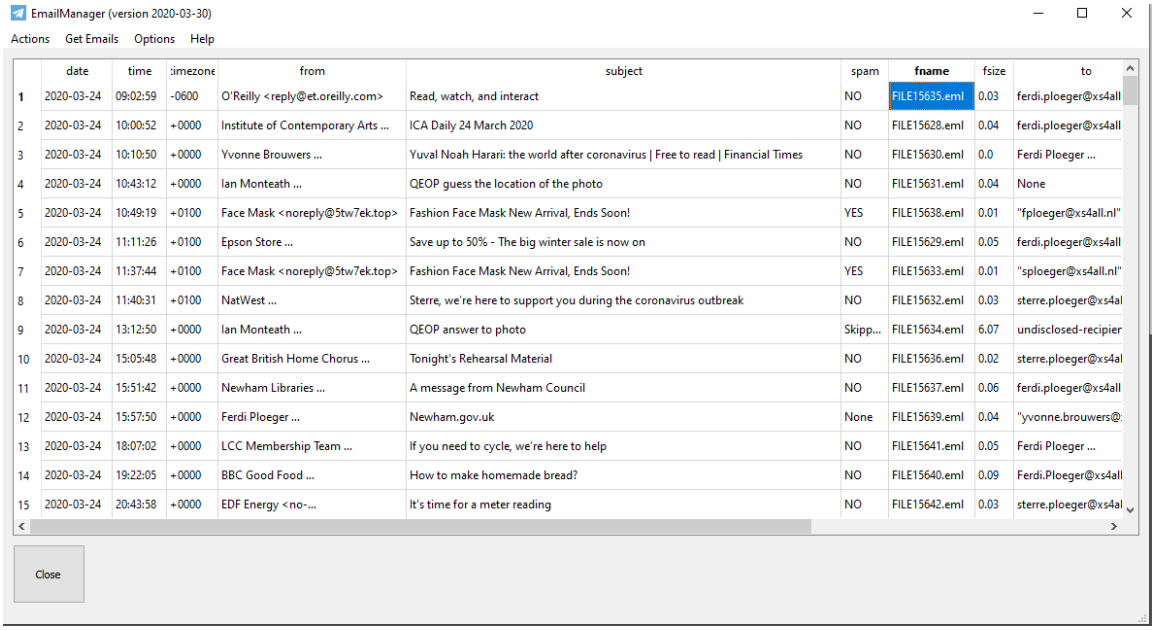
4.3 Get emails from last month

The menu-option 'Get Emails | from last month' will display a list of emails added to the EmailManager system during the previous 31 days. The list is sorted on the date and timestamp of the emails and standard displays as shown in fig 4.2.



	date	time	timezone	from	subject	spam	fname	fsize	to	cc	bcc
1	2020-03-24	09:02:59	-0600	O'Reilly ...	Read, watch, an...	NO	FILE15635.eml	0.03	ferdi.ploeger@...	None	None
2	2020-03-24	10:00:52	+0000	Institute of ...	ICA Daily 24 ...	NO	FILE15628.eml	0.04	ferdi.ploeger@...	None	None
3	2020-03-24	10:10:50	+0000	Yvonne Brouwe...	Yuval Noah ...	NO	FILE15630.eml	0.0	Ferdi Ploeger ...	None	None
4	2020-03-24	10:43:12	+0000	Ian Monteath ...	QEOP guess th...	NO	FILE15631.eml	0.04	None	QEOP ...	None
5	2020-03-24	10:49:19	+0100	Face Mask ...	Fashion Face ...	YES	FILE15638.eml	0.01	"fploeger@xs4a...	None	None
6	2020-03-24	11:11:26	+0100	Epson Store ...	Save up to 50% ...	NO	FILE15629.eml	0.05	ferdi.ploeger@...	None	None
7	2020-03-24	11:37:44	+0100	Face Mask ...	Fashion Face ...	YES	FILE15633.eml	0.01	"sploeger@xs4...	None	None
8	2020-03-24	11:40:31	+0100	NatWest ...	Sterre, we're he...	NO	FILE15632.eml	0.03	sterre.ploeger@...	None	None
9	2020-03-24	13:12:50	+0000	Ian Monteath ...	QEOP answer t...	Skipped, ...	FILE15634.eml	6.07	undisclosed-...	None	None
10	2020-03-24	15:05:48	+0000	Great British ...	Tonight's ...	NO	FILE15636.eml	0.02	sterre.ploeger@...	None	None
11	2020-03-24	15:51:42	+0000	Newham ...	A message fro...	NO	FILE15637.eml	0.06	ferdi.ploeger@...	None	None
12	2020-03-24	15:57:50	+0000	Ferdi Ploeger ...	Newham.gov.uk	None	FILE15639.eml	0.04	"yvonne.brouw...	None	None
13	2020-03-24	18:07:02	+0000	LCC Membersh...	If you need to ...	NO	FILE15641.eml	0.05	Ferdi Ploeger ...	None	None
14	2020-03-24	19:22:05	+0000	BBC Good Foo...	How to make ...	NO	FILE15640.eml	0.09	Ferdi.Ploeger@...	None	None
15	2020-03-24	20:43:58	+0000	EDF Energy <n...	It's time for a ...	NO	FILE15642.eml	0.03	sterre.ploeger@...	None	None

Figure 4.2: retrieved emails



	date	time	timezone	from	subject	spam	fname	fsize	to
1	2020-03-24	09:02:59	-0600	O'Reilly <reply@et.oreilly.com>	Read, watch, and interact	NO	FILE15635.eml	0.03	ferdi.ploeger@xs4all
2	2020-03-24	10:00:52	+0000	Institute of Contemporary Arts ...	ICA Daily 24 March 2020	NO	FILE15628.eml	0.04	ferdi.ploeger@xs4all
3	2020-03-24	10:10:50	+0000	Yvonne Brouwers ...	Yuval Noah Harari: the world after coronavirus | Free to read | Financial Times	NO	FILE15630.eml	0.0	Ferdi Ploeger ...
4	2020-03-24	10:43:12	+0000	Ian Monteath ...	QEOP guess the location of the photo	NO	FILE15631.eml	0.04	None
5	2020-03-24	10:49:19	+0100	Face Mask <noreply@5tw7ek.top>	Fashion Face Mask New Arrival, Ends Soon!	YES	FILE15638.eml	0.01	"fploeger@xs4all.nl"
6	2020-03-24	11:11:26	+0100	Epson Store ...	Save up to 50% - The big winter sale is now on	NO	FILE15629.eml	0.05	ferdi.ploeger@xs4all
7	2020-03-24	11:37:44	+0100	Face Mask <noreply@5tw7ek.top>	Fashion Face Mask New Arrival, Ends Soon!	YES	FILE15633.eml	0.01	"sploeger@xs4all.nl"
8	2020-03-24	11:40:31	+0100	NatWest ...	Sterre, we're here to support you during the coronavirus outbreak	NO	FILE15632.eml	0.03	sterre.ploeger@xs4al
9	2020-03-24	13:12:50	+0000	Ian Monteath ...	QEOP answer to photo	Skipp...	FILE15634.eml	6.07	undisclosed-recipier
10	2020-03-24	15:05:48	+0000	Great British Home Chorus ...	Tonight's Rehearsal Material	NO	FILE15636.eml	0.02	sterre.ploeger@xs4al
11	2020-03-24	15:51:42	+0000	Newham Libraries ...	A message from Newham Council	NO	FILE15637.eml	0.06	ferdi.ploeger@xs4all
12	2020-03-24	15:57:50	+0000	Ferdi Ploeger ...	Newham.gov.uk	None	FILE15639.eml	0.04	"yvonne.brouwers@
13	2020-03-24	18:07:02	+0000	LCC Membership Team ...	If you need to cycle, we're here to help	NO	FILE15641.eml	0.05	Ferdi Ploeger ...
14	2020-03-24	19:22:05	+0000	BBC Good Food ...	How to make homemade bread?	NO	FILE15640.eml	0.09	Ferdi.Ploeger@xs4all
15	2020-03-24	20:43:58	+0000	EDF Energy <no-...	It's time for a meter reading	NO	FILE15642.eml	0.03	sterre.ploeger@xs4al

Figure 4.3: emails shown in adapted layout

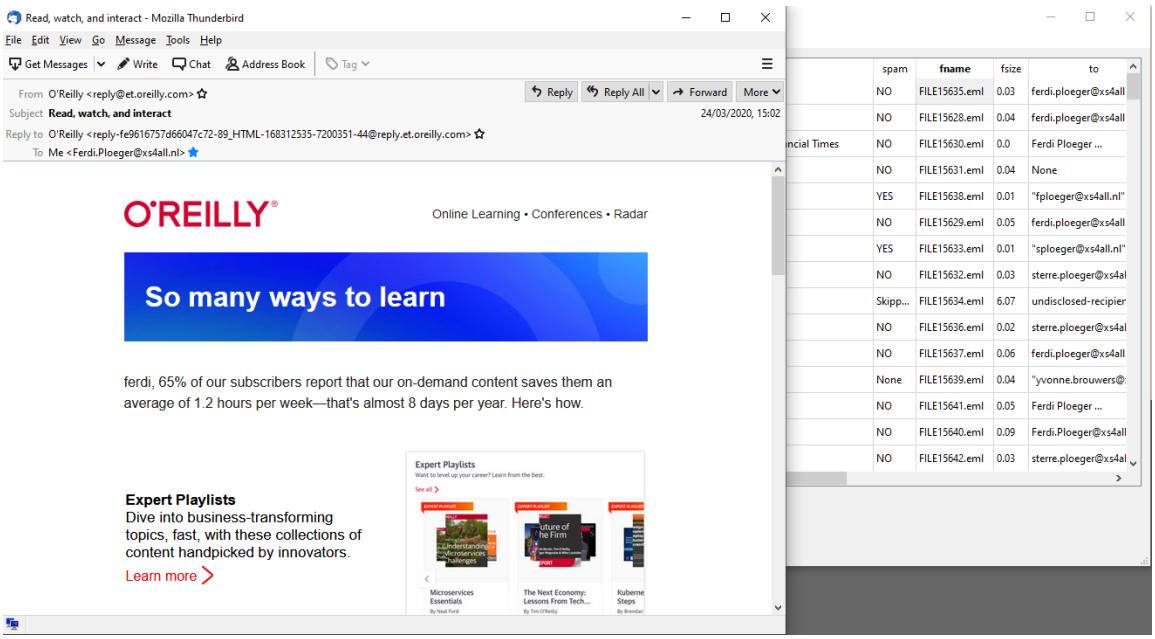


Figure 4.4: display an email

4.4 Get emails from a list

Retrieving emails from a list is asking EmailManager to read a textfile with the filenames of the wanted emails. This list can be assembled either by manually picking the filenames from displays as given in tables of 'today', 'last week' or 'last month' writing them in a .txt-file with i.e. Windows NotePad or retrieving this list from a copy and paste action from another application.

The list given in the Example.txt file (see fig 4.5) can be copied i.e. from a SQL-search in SQLiteStudio (see fig 4.6). Example.txt gives the outcome of a search of all emails stored with 'Amsterdam' somewhere in its subject. The search found 116 emails.

Although the list in Example.txt gives the filenames of the search with which these emails are stored in Emails.zip the order (by date and time) of this list doesn't show the filenames but from the place in the list of these filenames.To keep this order these files should be renamed as FILE00001.eml, FILE00002.eml, ..., FILE00116.eml and that is exactly what the program does (see fig 4.7).

EmailManager explains onscreen (and in its Log File) what it has done (see fig 4.8). Firstly the user choose d to write Example.txt in its Downloads folder: 'C:/users/ferdi/downloads/E EmailManager uses the filename 'Example' of 'Example.txt' to create a new folder 'Example'. If such a folder already exists EmailManager will delete it with its contents. Next it will extract aal the files with given filenames from Emails.zip in the current 'defaultfolder': here '//personalcloud-5/Public/Together/Ferdi/Diversen/emails/Emails.zip'. These files will be renamed FILE00001.eml, ..., FILE00116.eml following the order

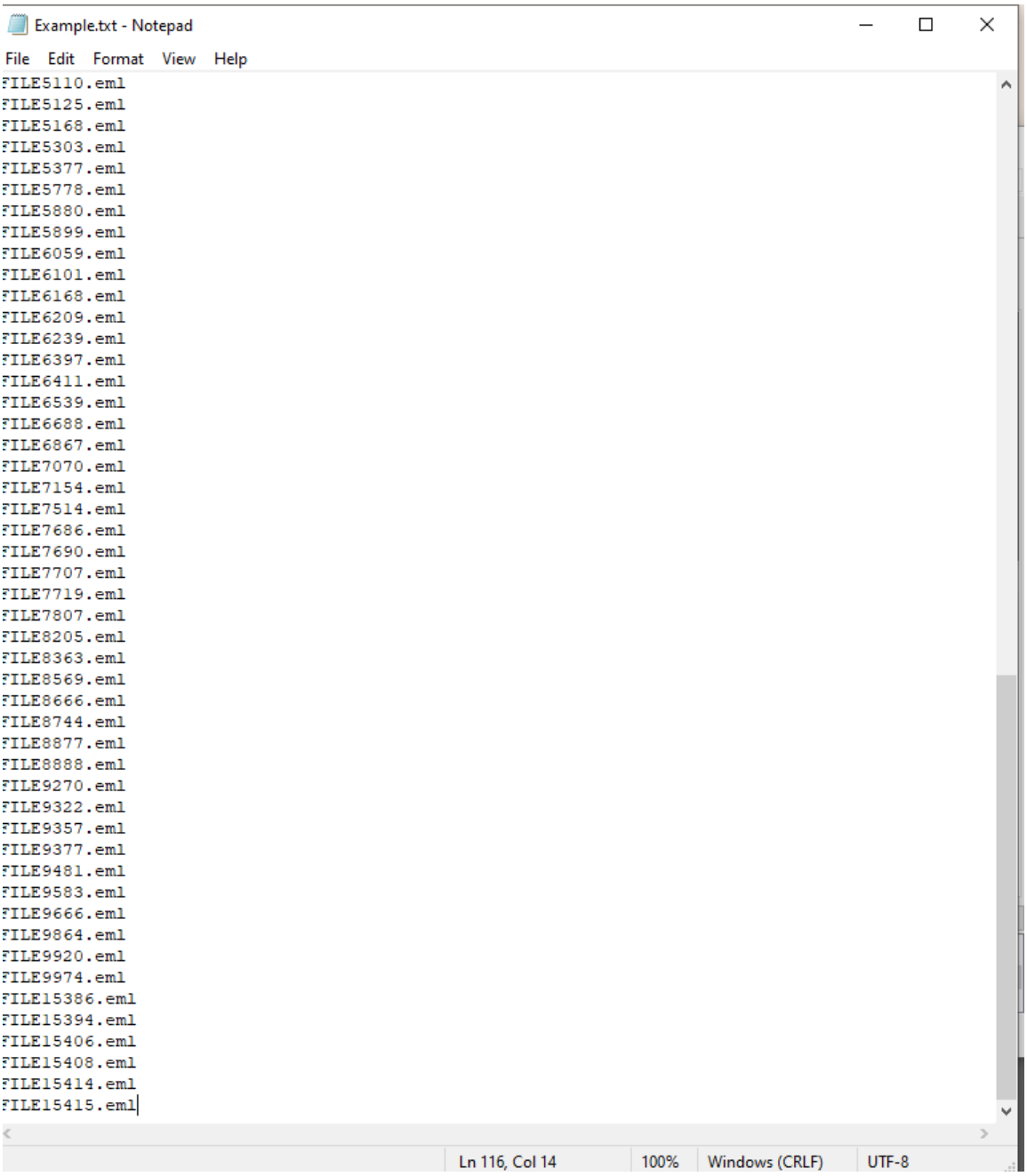


Figure 4.5: list of filenames in Example.txt

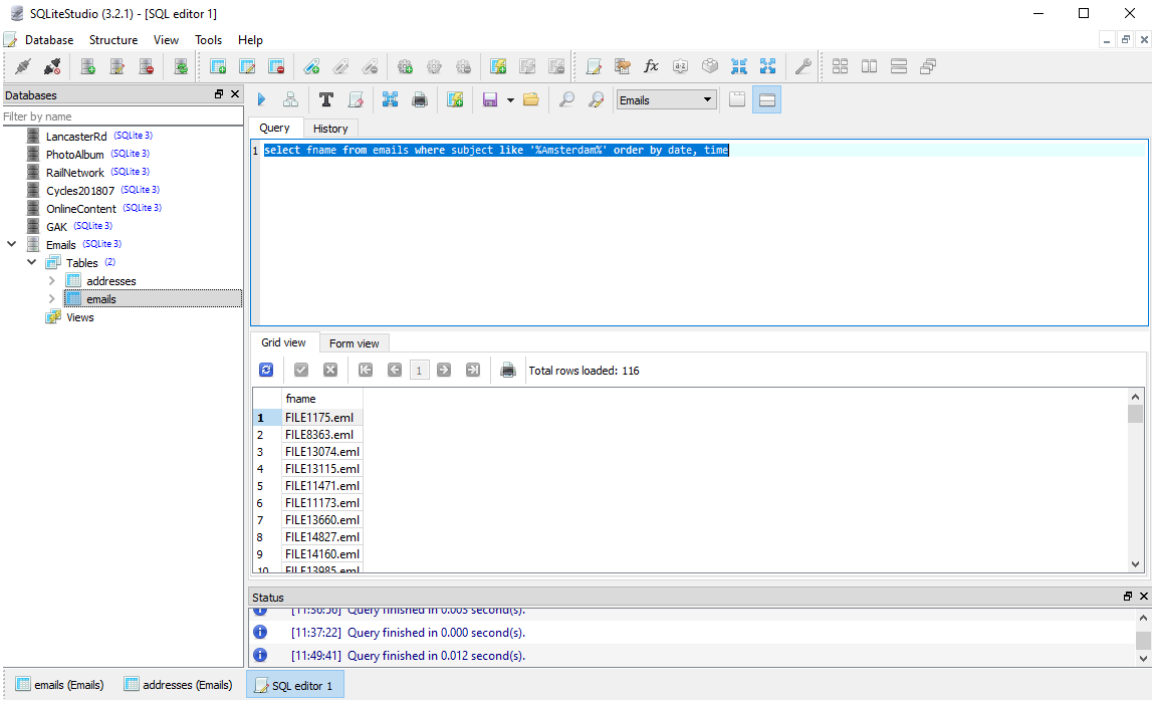


Figure 4.6: search in SQLietStudio

given in 'Example.txt'.

Unneccasry to state that thsi folder and/or these files can be manipulated and searched further.

4.5 get emails with SQL

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems.

SQL (Structured Query Language) is a database management language for relational databases. SQL itself is not a programming language, but its standard allows creating procedural extensions for it, which extend it to functionality of a mature programming language.

So, SQL is not a programming language, it's a query language. The primary objective where SQL was created was to give the possibility to common people get interested data from database. ... So once you learn SQL it should be similar to work across any relational databases.

EmailManager uses intern SQL-statements to retrieve the emails 4.1, 4.2 and 4.3. For these options the SQL-statements are 'hidden' for the user.

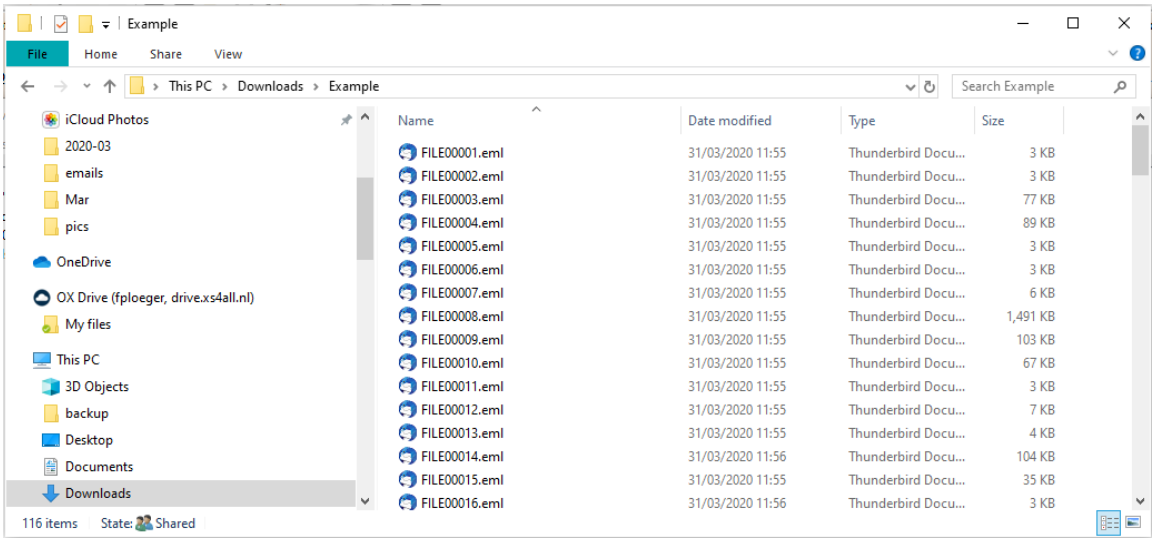


Figure 4.7: new list of emails

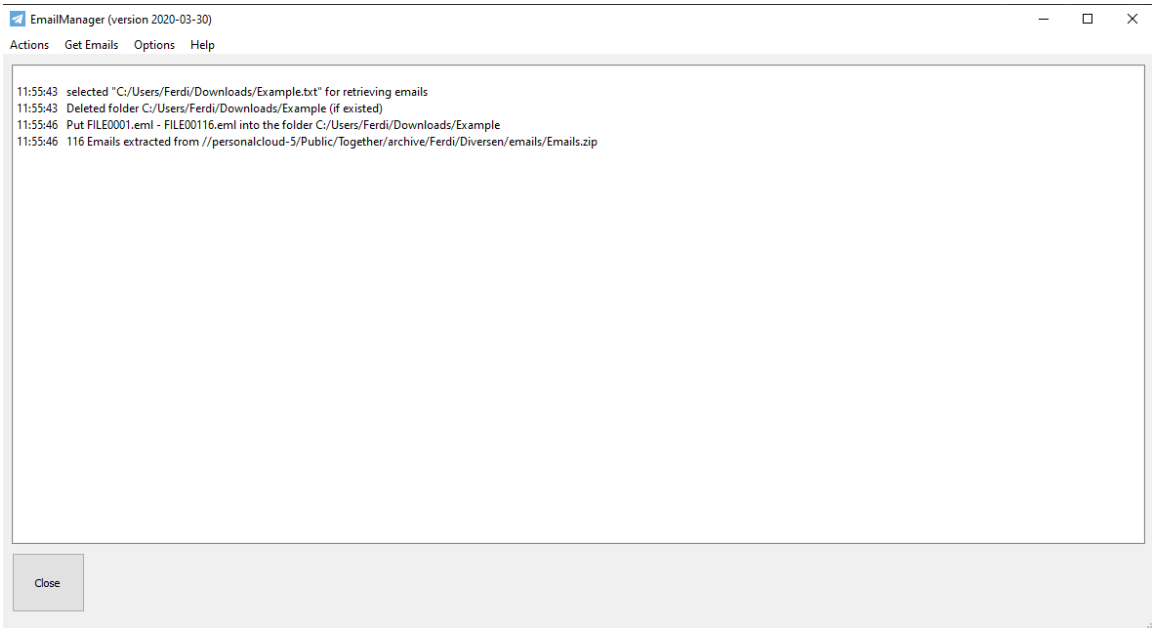


Figure 4.8: making a new folder with emails

Two other options are offered: to execute SQL-statements 4.5.1 and to 4.5.2 within the program.

4.5.1 Read SQL from a file

A SQL-file is a text-file that can be composed in simple text-editors like MSWindows NotePad. The file should be saved with the extension .SQL (rather then .txt), i.e. 'FILE.SQL', anywhere on your disks.

The file should contain a valid SQL-statement. 'Valid' means it can execute the statement using the EmailManager.db in the 3.[2]

EmailManager will display the emails found as a result of the SQL-statement in the FILE.SQL. The result should contain at least the column 'fname'.

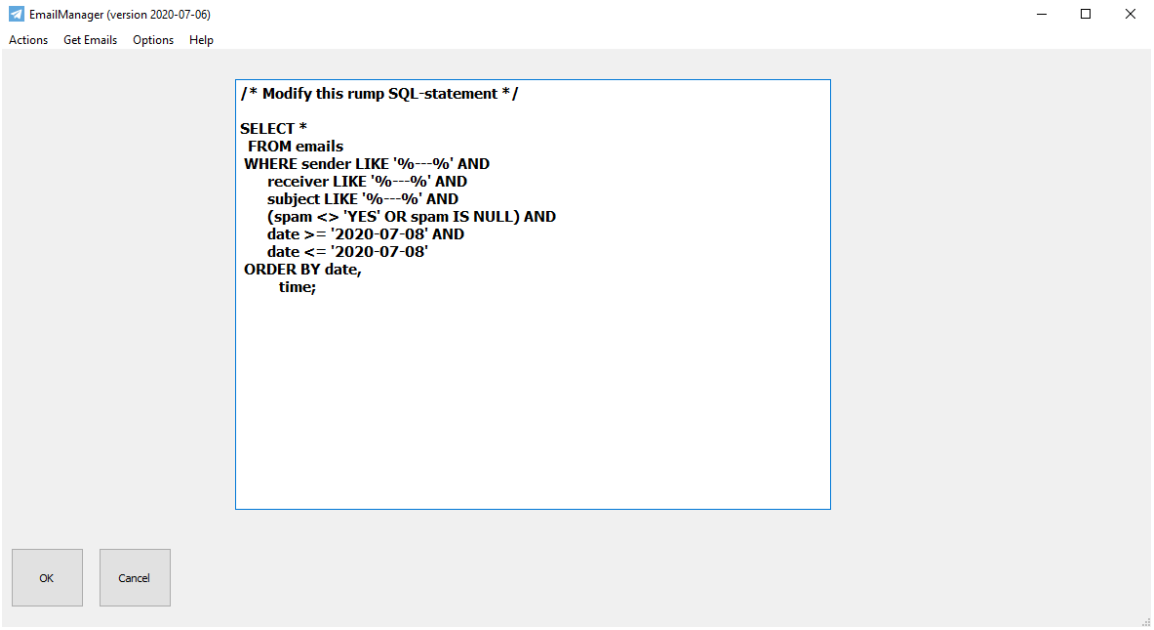
When clicked on one of the results under fname the email will be displayed and the email (email.eml) will be written in a folder that was created bij EmailManager in the same folder where FILE.SQL was found. This folder will have the same name as the SQL-file. So, when the SQL-file has the name 'EXAMPLE1.SQL' EmailManager will create a folder 'EXAMPLE1'. Before EmailManager starts writing these emails it tries to delete a possible folder of the same name.

4.5.2 Adapt basic SQL-statement

EmailManager offers the option to adapt a basic SQL-statement. This option is under Get Emails | with SQL.

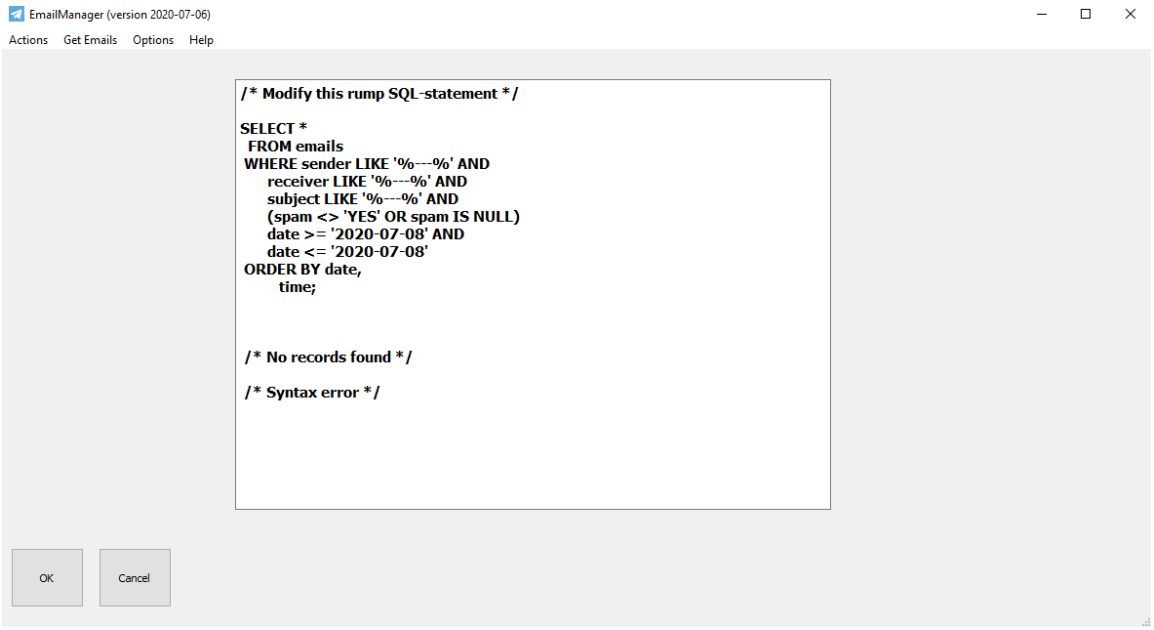
The user will then be confronted with an editor and a basic extendable SQL-statement:

[2]It is possible to have more than one folder in which emails are stored

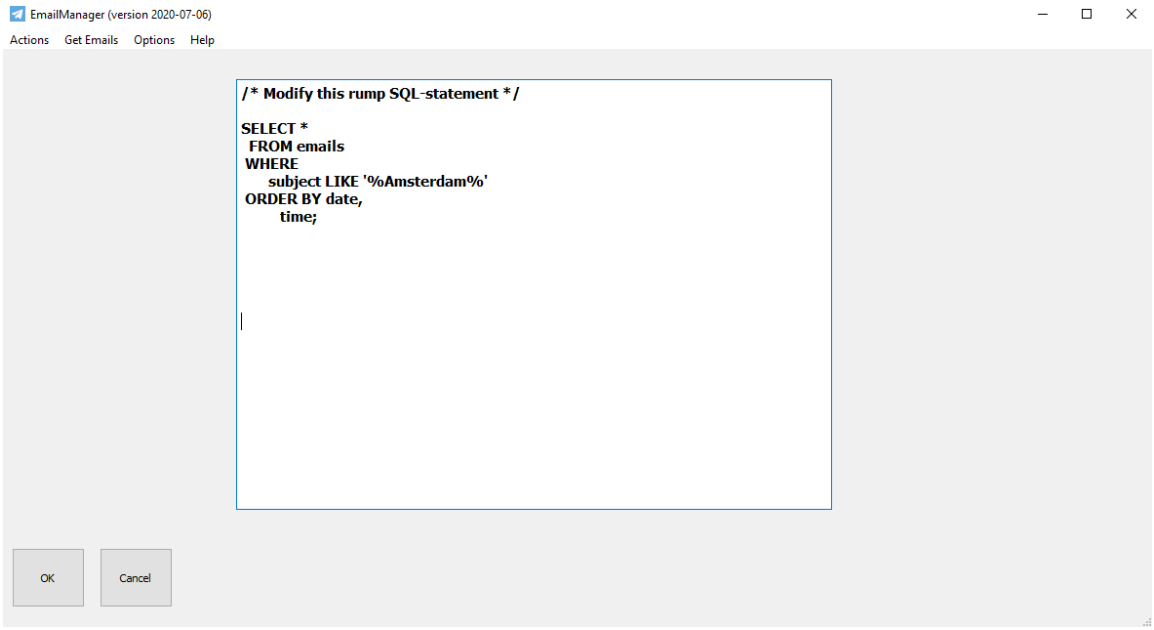


This statement can then be adapted as one wishes. The OK-button makes EmailManager try to execute the statement, the Cancel button cancelles the request.

EmailManager gives feedback in case of 'No records found' or in case of a syntax error:



If the SQL-statement leads to results, any email that is asked to be displayed will be written in the folder C:/EMAILManagerResults as an .eml-file under the name it was stored in the EmailManager.zip-file, so something like FILE17568.eml.



4.5.3 Notes on search criteria

4.5.3.1 Using quotes

Search criteria, other the numbers, should be placed between parenthesises, so

```
WHERE date = '2020-06-03'
```

gives emails with matching dates '2020-06-03'
When the 'LIKE' clause is used the extra '%' should be put between the quotes:

```
WHERE date LIKE '%2020-06%'
```

gives all the emails from June 2020

4.5.3.2 Use of space

The use of space has no meaning in SQL, so

```
SELECT * FROM emails
```

is equivalent with

```
SELECT
*
FROM emails
```

4.5.3.3 Upper case or Lower case

SQL doesn't distinguish between uppercase or lowercase. So the SQL:

```
SELECT * FROM emails
WHERE
subject = 'Amsterdam'
```

is equivalent with:

```
SeLeCt * FROM Emails
WHeRE
subjeCt = 'AMSTErDAM'
```

4.5.3.4 Selecting specific columns of the emails

With

```
SELECT * FROM emails
```

you ask for all columns in the emails.

You can change the * with specific colums, like:

```
SELECT date, time, sender, subject, FName FROM emails
```

which will give just the specified columns as output.

However when FName is missing the SQL-editor will display a message that 'FName is missing'. It is only clicking on one of the FName's that will display the email as it is stored.

4.5.3.5 Number of 'WHERE'-clauses

There is no limit on the 'WHERE'-clauses, so you can add or delete them from the rump-SQL at will. However be aware that these clauses should be connected with 'AND's and 'OR's

4.5.3.6 Difference between '=' and LIKE

A search criterium with '=' expects the criterium exactly being met (apart from uppercase and lowercase). So the statement:

```
WHERE subject = 'Amsterdam'
```

will return records wich subject is exactly 'Amsterdam' and ignores subjects like 'Amsterdam ' (i.e. extra space).

A search criterium 'LIKE' is being met when part of the subject matches the criterium, so:

```
WHERE subject LIKE '%Amsterd%'
```

finds subjects like 'Amsterdam', 'Amsterdam Centraal', 'from Amsterdam', and so on.

It is of course possible to give the search criterium in more complex statements: (take care of the right use of parenthesises)

```
WHERE subject LIKE '%Amst%' AND
(subject LIKE '%Centraal%' OR subject LIKE '%CS%')
```

This will give emails with 'Amsterdam CS' and 'Amsterdam Centraal'

4.5.3.7 Occurences of 'None' or 'Invalid'

Although 99% of the emails have all columns present, some emails are being sent without all of them them. An email without date (time, timezone, sender, ...) or with an invalid date can be found with:

```
WHERE date = 'None' OR date = 'Invalid'
```

5 Why is SQL the most important skill to learn?

SQL (Structured Query Language) is the primary language responsible for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

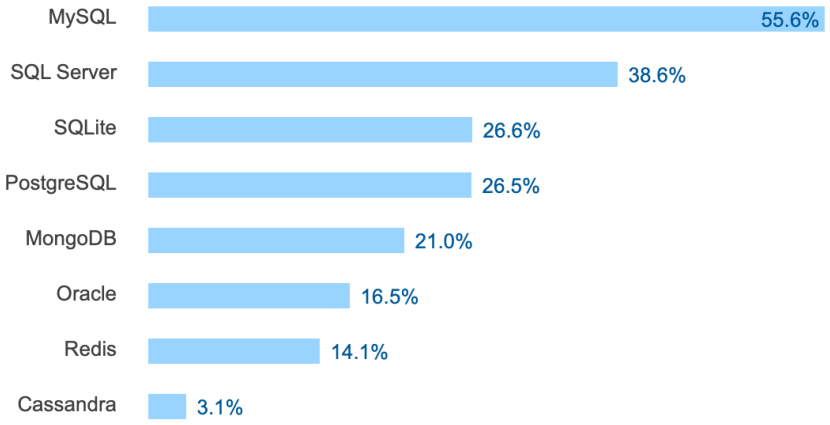
Simply put, SQL is the language you use to interact with a database.

5.1 Why it's important to learn SQL?

5.1.1 SQL is the most universal and common used database language.

We live in an era where data is the most valuable asset and it's being put at heart of every decision making process. Despite of the explosion of NoSQL in the recent years, SQL is still making its way back to become the universal interface for data analysis once again.

It powers the most commonly used database engines like MySQL, SQL Server, SQLite, and PostgreSQL:



The most popular databases. Source: StackOverflow Developer Survey Results 2017

It's not something surprising as SQL is particularly effective at data manipulation. You can be able to see the exact data and the way it works then data testing and manipulating will be done faster. Evenmore, the data stored in a relational database is dynamic, which means it can be queried, modified, and manipulated easily with some basic SQL queries.

If you want to access databases then yes, you need to know SQL.

5.1.2 It is not really difficult to learn SQL.

SQL is not a programming language, it's a query language. The primary objective where SQL was created was to give the possibility to common people get interested data from database. It is also an English like language so anyone who can use English at a basic level can write SQL query easily.

The good news is that most DB engines are compatible with all SQL code. So once you learn SQL it should be similar to work across any relational databases.

5.1.3 SQL is one of the most sought-after skills by hiring employers.

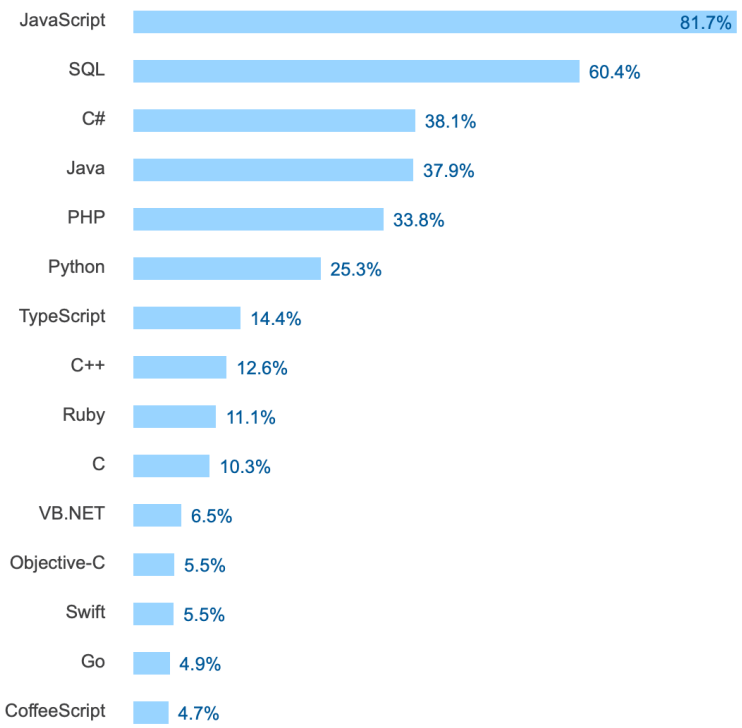
Let's take a look at some actual numbers from the job market:

On Indeed.com:

- There are 149,124 developer jobs in total (at the moment I did this quick research, Aug 8).
- Out of that number, there are 105,146 job listings that are either hiring SQL Developers or having SQL as a required skill.
- There are 35,306 SQL Developer jobs.

On Stackoverflow:

SQL is among the most popular languages across web developers, desktop developers, sysadmins/DevOps, and data scientists:



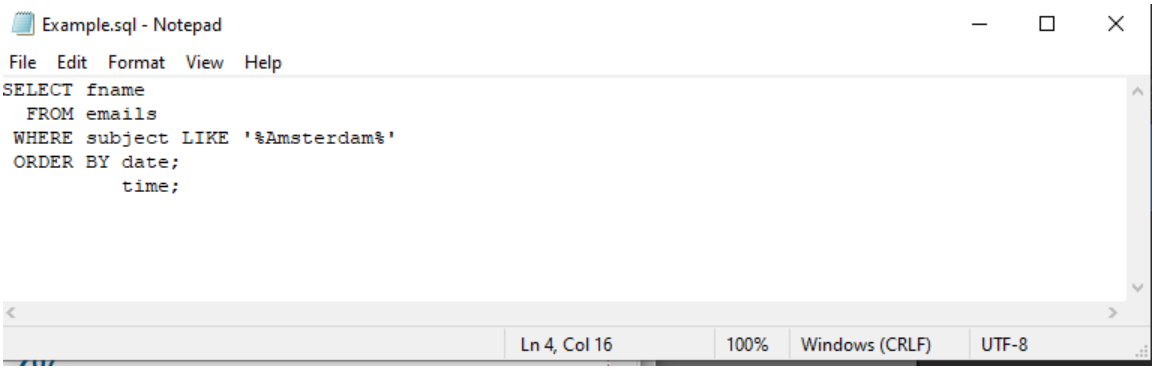
It's easy to understand as all companies no matter which industry they are in, they all rely on data and need to organize and understand the information in a relevant way. Chances are, they are going to encounter SQL databases, a lot. So they are always going to need a database professional.

5.2 Examples of EmailManager and SQL

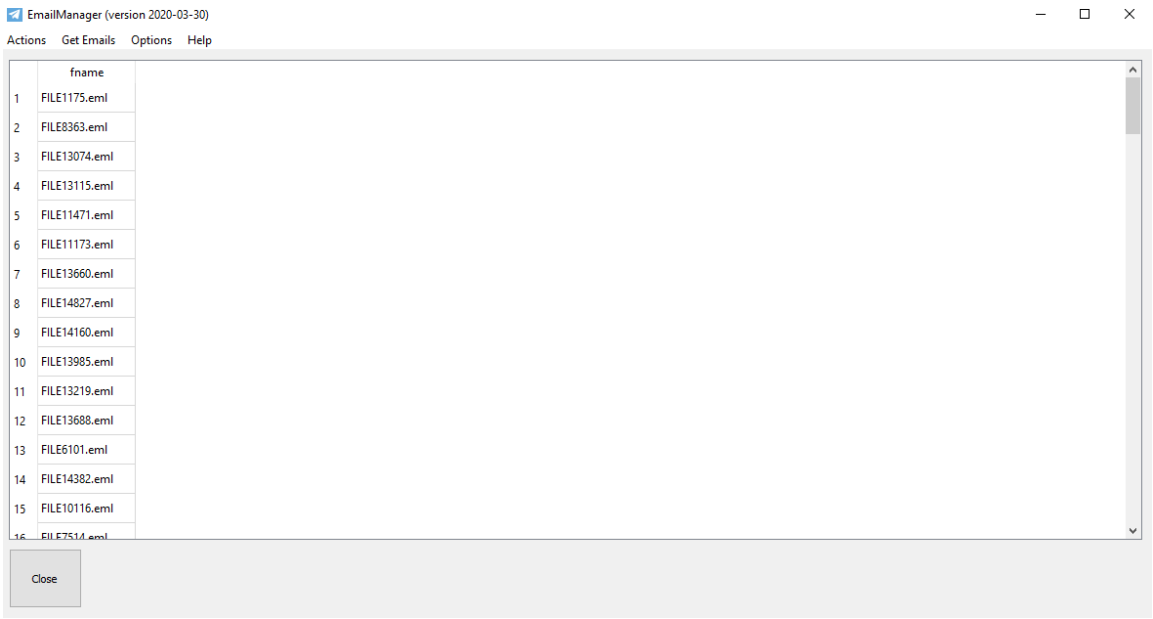
When offered a textfile with SQL-statements given as 'Example.sql', so with an extension .sql, EmailManager will try to apply this SQL-statement to 'Emails.db' in the default folder.

The file Example.sql can be put anywhere Emailmanager will always use the right Emails.db bfrom the defaultfolder.

It is possible that the file Example.sql hase errors: in that case EmailManager cannot solve the SQL-search and will display its discontent on screen, although it will not show what the error was.



Because SQL is simple it is not difficult to spot the error: the semicolon after 'date' should be a comma. After replacing the semicolon with a comma and try it again, EmailManager will display:



being the same list as was used in 'Emails from a list'. Of course if the select statement was extended like:

```
SELECT * FROM emails
```

All the columns of the table Emails would be shown.

Clicking on a filename (fname) will display the email and simultaneously write this email in the folder Example, being a subfolder of the folder where Example.sql was written in.

5.2.0.1 Examples of sql

There are only 2 tables in Emails.db: 'emails' and 'addresses'.

The table addresses has two relevant[1] 'columns': address and spam. This table collect all email-addresses EmailManager one found in emails and adds the column 'spam' about this address was encountered in a spam-message.

A simple query could be:

```
SELECT * FROM addresses WHERE address LIKE '%.ru%'
```

thus finding all the email-addresses once used that obviously came from Russia (.ru).

The other table - the more relevant one - is emails.

The table emails has the relevant columns: date, time, timezone, sender, subject, spam, fname, fsize, to, cc and bcc.

In analogy of the above search you can do a search on emails sent from Russia:

```
SELECT * FROM emails WHERE sender LIKE '%.ru%'
```

You can add more attributes like:

```
SELECT * FROM emails WHERE sender LIKE '%.ru%'
and date > '2020'
and spam = 'YES'
order by date, time
```

[1] the column 'count' is not relevant and has always value 1