The goal of this challenge is to build a script which identifies invoices that have been paid twice. A double payment refers to the act of paying for a service or a product two times, effectively doubling its cost. They are frequently caused by human errors, and friction within the invoice handling process.

There are many possible ways on how a duplicate payment can occur and even more ways to analyze the data in order to find one. In this challenge, you should focus on identifying duplicate invoices with the following characteristics on the vendor attribute (**NAME1** column):

1. Duplicate vendor master data, when there are two entries available in the ERP system for the same vendor – for example ACS S.r.l. and ACS Srl. Or PERSONAL and PERSONAL GmbH
2. The company has many different vendors with multiple subsidiaries such as Telekom vs. Telekom Interstate or IT-Services vs. IT-Services Belgium, so the accountant confused companies with its subsidiaries, such as entering an invoice with vendor E&Y GmbH instead of E&Y Real Estate GmbH
3. Typing errors e.g., Schmidt vs Schmitt, O vs. 0, S vs. 5, or omitting certain characters

There might be hundreds of invoices that have been booked on these vendors in the dataset. In order to distinguish which invoices are considered to be a duplicate, the duplicates must be equal in the following attributes (AND condition):

- Invoice amount (column: WRBTR)
- Company code (column: BUKRS)
- Original invoice date (column: BLDAT)
- Original invoice reference (column: XBLNR)

Duplicate invoice pairs with above mentioned characteristics would typically look as the examples below (they don't necessarily have to be a group of 2 but it is most common):

1. Duplicate vendor master data

BELNR	BUKRS	BLDAT	XBLNR	NAME1	WRBTR
01234	0450	2021-01-15	0485/INV	PERSONAL	12345.1
01235	0450	2021-01-15	0485/INV	PERSONAL GMBH	12345.1

2. Multiple subsidiaries

BELNR	BUKRS	BLDAT	XBLNR	NAME1	WRBTR
01300	0300	2020-10-10	GEBUEHREN	E&Y GMBH	13000.5
02000	0300	2020-10-10	GEBUEHREN	E&Y REAL ESTATE	13000.5

3. Typing errors

BELNR	BUKRS	BLDAT	XBLNR	NAME1	WRBTR
03580	0300	2020-12-03	014500	SACHKOSTEN.	4000
03610	0300	2020-12-03	014500	SACHKOSTEN	4000

We provided you with a dataset from financial SAP data. The columns are standard SAP columns. You can focus on the main columns listed below:

- Invoice number in the system (column: BELNR)
- Vendor name (column: NAME1)
- Invoice amount (column: WRBTR)
- Company code (column: BUKRS)
- Original invoice date (column: BLDAT)
- Original invoice number (column: XBLNR)
- Accountant's comment (column SGTXT)

We have attached a .csv file along with this document containing 12 thousand rows of invoice data, which you will be using for this task. The hints given above may make this challenge quite trivial. However, a perfect duplicate detector is useless if it requires a runtime of an entire month to process millions of invoice data. Naïvely comparing all rows of data with each other can yield a performance of O(n^2) in the worst-case scenario – even with just 10k rows, that is 100 million comparisons to make! As such, an important benchmark for this test is to minimize the time it takes to analyze the 12k rows of data and finding a duplicate.

The task for you is to create a python script that finds duplicate invoices with above mentioned criteria and to minimize the number of possible row comparison – design your solution as performant (i.e. as quick) as possible.

You will be scored in the following criteria:

1. Number of true duplicates found
2. Number of false positives (i.e. invoices that are not actually duplicates) found
3. Time it takes to run your code from scratch and produce the results

You are free to use any library you find, and implement your own methods, so long as they are in Python 3. We do recommend, however, using **Pandas** as your main data managing library, since it offers powerful functions and is the main tool that we use in Duplid.

As submission, please provide us with a link to a github repository, or a zip file including the Python code you have written. Keep relative directories for the joined_dataset.csv file in mind! The github repository or the zip file should additionally contain the following files:

1. A .csv file that lists the duplicates you have found using the code, which will be compared with the actual output of the software that you send us
2. A Readme that describes how the code should be run
3. A requirements.txt file that lists the libraries used, so that we can install the necessary libraries for running the code
4. The source csv file, joined_dataset.csv, in the correct directory, so as to minimize the setup required for running your code

The submission deadline for this task is 29.03.2021, 23:59. Late submissions will not be considered.

Submission can be done by sending an email to product@duplid.de containing the deliverables.