Task #3: Data profiling

1. brief description of how you checked the data. If needed screenshot(s), code/query snippets, excel formulas can be added and explained

Reviewed CSV file in Excel and used Textjoin to compile tags, then adding RIGHT to find possible tag with category\_id. Then used VLOOKUP to associate category\_id with category titles and combine the tables (GB\_videos and categories).

Afterwards, used Python and Pandas library to review data by looking at data type, null value and correlation between numeric columns. Benefits from using Pandas is that it will create a dataframe from the CSV file and we can use it to perform data analysis and data manipulation – more in this in the following points. Used the following python commands in Jupyter Notebook for the analysis



2. brief description of non-trivial data quality issues you noticed if any

Before concluding on critical data issues, it needs to be understood what is important to the company in terms of data-points and let that guide the analysis. If data is not important and will be excluded from the analysis, then it might not be required to clean or remove these fields.

Looking at the data it’s noticed that columns in (publish\_time, views, likes, dislikes, comment\_count, thumbnail\_link, comment\_disabled, ratings\_disabled and video\_error\_or\_removed) have exactly 2080 blank values. These are obvious candidates for deletion or manipulation unless they contain other data points that is vital for further analysis.

3. brief description of non-trivial improvements in data processing you would suggest if any

Empty cells can potentially cause incorrect results hence disrupt the analysis. We can potentially remove these rows if we view that the dataset is large enough to continue the analysis. In Pandas combining the pd.read\_csv and dropna will remove the rows and create a new dataframe to analyse further. Another approach is to use the fillna() method which will keep the row and add a defined value, either chosen by us or we can replace using Mean, Median or Mode (value that appears most often).

If a value is in a wrong format, then we have two options; either delete or attempt to correct the format so we can analyse the dataset. To achieve this we can combine the pd.read\_csv with pd.to\_*dataformat*(). A good approach could be to use a combination of converting the data format (example date; trending\_date) and if not possible, we can use the remove option - dropna. If the dataset was smaller, we could consider using replace wrong values, where we would target specific columns. This is meaningful with obvious entries. For larger datasets, we can use a replace if condition is meet, looping through the rows. An example we want to delete rows with video\_id with value \n (new line in programming). Here we can use Pandas to remove these entries as we view them as invalid.

for x in df.index:

if df.loc[x, "video\_id"] == 'NaN':

df.drop(x, inplace =True)

print(df.to\_string())

We can also identify rows with duplicate values by combining pd.read\_csv and duplicated() can print the occurrence in boolean expression.

4. your final conclusion whether this data set is good enough for further analysis

The data set contains useable data, but will need some cleaning based on the suggested approaches described previously. We will need to identify what is the used columns so we can review the quality more in depth.

For 2080 blank values in columns publish\_time, views, likes, dislikes, comment\_count, thumbnail\_link, comment\_disabled, ratings\_disabled and video\_error\_or\_removed, I believe that these rows can be dropped. On the other hand, if we have great category data and this is of importance, then we can consider using Mean as a value if this otherwise will cause issues with calculations.

Country and Category are more often than not blank and would be candidates for review based on the value they provide for further analysis and conclusions.

# Not Null usage

When creating our tables, we can ensure to add not null constraint for columns which is critical for reading data correctly and naturally for primary and foreign keys. However, while ensuring that data can be queried in a meaningful way, we want to ensure that the database isn't too rigid. Say if an employee haven't got a Latvian personal-code yet or filed a travel request with an open-ended return date (endDate), consider whether data is useful to enter or consider whether it’s an incomplete entry.

## Triggers

Adding triggers will allow for an automated check when data is created or updated. Triggers will guarantee that inserted and updated data is valid, that it follows business rules, are unique records and can generate log/audit changes in the database. Triggers can be used before or after the command (trigger\_time) and it will then function as either a warning or an it returns error.

While triggers comes with benefits mentioned, they can cause issues when developers are debugging as they occur 'behind the scene'. Triggers can initiate other triggers and can cause confusion for employees who are not aware of them – deletion one place can insert a new row another place.

Maintaining data quality

It’s not easy to completely prevent low quality data to enter a database and perhaps sometime it’s allowed to intake data with varied quality to get enough data into a database. Therefore, the ability to maintain data quality and checks are key and will allow for a flexible intake of data combined with a way to identify compromised records. The following examples will go through some of the consideration for the EMPLOYEE and related tables.

# Assertion queries

To verify our data, we can run queries daily or on a regular basis to ensure that data follows certain business rules. Checking for null values, unique entries, using WHERE clause to query unwanted entries such as

* column\_name is NULL
* len(personalCode) > 12
* NOT process IN (“approved”, “rejected”)
* etc.

# Former employees

As the workforce in Nordigen is dynamic and talented people will join and leave the company, consideration must be taken to how to handle such cases from a database perspective.

## Cascade delete

When considering how to handle employees leaving the company, an option is to use cascade delete. This will enable to delete an employee and the records related to this employee - when the id is used as a foreign key in other tables.

While allowing to delete rows in multiple tables gives certain flexibilities, consideration must be taken to think about issue that might come with it. TRAVELS table might be compromised if an employee is removed that is added as byEmployee as it will remove entries for other employees.

## Add active/inactive column

Adding a new status column to the EMPLOYEE table will allow keeping a reference to employees who has left the company. Previous employees id used as a foreign key in other tables will be kept while enabling queries to distinguish between current and previous workforce.