# Project 1 Big Data Analytics

## EC2 info

AMI Detail:

```
Ubuntu, username = ubuntu
```

Public DNS:

```
ec2-54-167-36-132.compute-1.amazonaws.com
```

Public IP:

```
54.167.36.132
```

Login & Copy:

```
$ ssh -i 15619Project1.pem ubuntu@54.167.36.132

$ scp -i 15619Project1.pem -r Project1 ubuntu@54.167.36.132:
```

# Data Pre-processing

Pre-processing the initial data so that it is in the proper form is crucial before you can even begin analyzing it to gain any insights. **Never assume that the dataset is perfectly clean and well formed.** The phrase "garbage in, garbage out," points out that if your input data is malformed, you should not expect to gain insight. The phrase is also particularly applicable to data mining, machine learning or any data analysis project. Thus, the representation and quality of data must be ascertained and ensured before analyzing the data.

Your first task this week is to transform and filter out rows from the logs based on the following rules below.

1. **Handle dirty data**: Removing malformed data is always the very first step in data pre-processing. Some lines don't have the four columns as expected. You should filter out these lines, for example:

   ```
   en  506 0
   ```

   **In data processing, you need to make sure that your code must not crash when given malformed data.**

2. **URL normalization and percent-encoding**: According to RFC 3986, URLs should be normalized and special characters will be converted by percent-encoding.

For example, user talk created by `K6ka` with the title

```
User talk:K6ka
```

actually matches this URL:

```
https://en.wikipedia.org/wiki/User_talk%3AK6ka
```

Although `%` percent encoded titles are expected in URLs, decoded titles are expected in pageview logs. Due to arbitrary user behavior and third-party usage, various titles pointing to the same page still exist for some pages, such as `Special%3ASearch` and `Special:Search`. As a result, we need to map all these titles to the same original title `Special:Search`. We will accomplish this by making use of percent decoding.

However, the definition and solution of percent encoding for Wikipedia is slightly different from the widely used definition, and you may find that most percent decoders will break with our dataset. The reason for this is that there are many `%` symbols which are not followed by hex chars, such as `1%_rule_(Internet_culture)`. Most percent decoders can only be used in **perfectly encoded character sequences** where `%` has been encoded into `%25`.

**Once again, please do not assume that the dataset is in perfect format or encoding.** Developers of The Wikimedia Foundation are using their own decoder with the following two differences in contrast with the common URL decoder.

- They keep percent signs that are not followed by hexadecimal digits.
- They do not convert plus-signs to spaces.

We don't want you to reinvent the wheel, hence, we offer you code snippets in Java, Python 2 and Python 3. You are allowed to copy and adapt the percent-decoder code snippet into your code without citing it in the references file.

Use them wisely. If you wish to use Bash, it will help if you use piping with the Java/Python snippets.

Note that URL normalization usually requires other steps besides percent-encoding. In this project, we will ask you to only complete percent-encoding. Additionally, bear in mind that you can reorder the rules if you wish in your code, as long as handling dirty data and percent decoding are considered as the prerequisites for other rules. Note that the URL Encoded Characters include tabs and spaces which will change the number of columns in the records after percent-decoding. Think twice about which rule you should apply first in your data filter.

3. **Desktop/mobile Wikipedia sites**: We will focus on English Desktop/Mobile Wikipedia pages, keep the rows only if the first column is exactly `en` or `en.m` (**case sensitive**) and exclude all others.

4. **Wikipedia namespaces**: There are many special pages in Wikipedia that are not actually Wikipedia articles. Remove these pages. A Wikipedia namespace is a set of Wikipedia pages whose names begin with a particular reserved word recognized by the MediaWiki software (followed by a colon).

   For example, in the user namespace, all titles begin with "User:". In the case of the article (or main) namespace, in which encyclopedia articles appear, the reserved word and colon are **absent**. The details for namespaces are as follows for your reference, and in this project we will only focus on namespace 0 (Main/Article):

| Wikipedia namespaces | | | |
|---|---|---|---|
| Subject namespaces | | Talk namespaces | |
| 0 | (Main/Article) | Talk | 1 |
| 2 | User | User talk | 3 |
| 4 | Wikipedia | Wikipedia talk | 5 |
| 6 | File | File talk | 7 |
| 8 | MediaWiki | MediaWiki talk | 9 |
| 10 | Template | Template talk | 11 |
| 12 | Help | Help talk | 13 |
| 14 | Category | Category talk | 15 |
| 100 | Portal | Portal talk | 101 |
| 108 | Book | Book talk | 109 |
| 118 | Draft | Draft talk | 119 |
| 446 | Education Program | Education Program talk | 447 |
| 710 | TimedText | TimedText talk | 711 |
| 828 | Module | Module talk | 829 |
| 2300 | Gadget | Gadget talk | 2301 |
| 2302 | Gadget definition | Gadget definition talk | 2303 |
| Virtual namespaces | | | |
| -1 | Special | | |
| -2 | Media | | |

**Figure 2: List of Wikimedia namespaces (Source)**

Disambiguation pages, templates, navboxes, user pages, discussion pages, file pages, category pages, help pages, and Wikipedia policy pages are not articles. Keep pages only if they belong to namespace 0 (Main/Article) and those grouped into other namespaces should be excluded.

We offer you a JSON file listing all the namespaces which need to be excluded, as well as a readable version.

Instead of typing the namespaces by yourself, you should parse the JSON file with code to get all the namespaces (except for `namespace 0`). **Having a strong understanding of working with JSON will help you a lot in later projects!** You may notice that there are attributes about case sensitivity in the JSON file, please ignore them and follow our own rules.

- If you are using Java, you may use **Jackson** **or** **Google Gson**. We do not recommend downloading the lib jars manually to compile the source code with `javac -cp`, because this approach is error-prone and not maintainable. Besides, please note that we will exclude any file with its size larger than 5MB upon submissions. Maven is the tool widely used in our course to manage Java dependencies. If you are new to Maven, please read Maven primer first where a template is also available for you to start. If you get errors such as `Could not find or load main class` or `java.io.FileNotFoundException`, we expect you to debug the errors on your own.
- If you are using Python, you may use the **json** module in the standard library.
- You are also allowed to use other libraries of your choice.

Note:

- In order to improve the performance when we test your code, you should use a **separate** java/python/bash file to parse the JSON file and generate a blacklist to be used in your data filter code.
- You can then manually copy the blacklist from your JSON parser to your code which filters the data.
- Do not parse the namespace JSON repetitively each time the data filter program processes a new record. The filter will also be used in the next project with Hadoop streaming on a much larger dataset.
- Use attribute-value pairs with attribute `*` instead of `canonical`.
- When you submit your code, do not forget to submit the code to parse the JSON file as a proof of your work.

Page titles may contain spaces visually, but their URLs will replace spaces with underscores and this rule also applies to our pageview data.

Note that the standard namespace blacklist is **case insensitive**. You are expected to generate a prefix blacklist similar to this:

```
media:
special:
talk:
user:
user_talk:  # instead of "User talk:"
wikipedia:
wikipedia_talk:
... # some title prefixes omitted
gadget_definition_talk:
```

You are required to generate the blacklist in the format above to show your learning of JSON parsing. However, with the blacklist ready, you are free to transform and use the blacklist in your own approach.

As we mentioned, some titles use `%3A` or `%3a` instead of `:` due to percent encoding. Before you apply this blacklist, such titles should have already been percent decoded into `:`.

Blacklist filtering can take less effort if you make use of built-in functions in Python or helper utility libraries in Java. For example:

```
any(title.startswith(prefix) for prefix in prefixes) # python
StringUtils.startsWithAny(title, prefixes);          # java with Apache Commons Lang library
```

You may need to change the code above to make it case insensitive.

5. **Wikipedia article title limitation**: Wikipedia policy states that if any article starts with an English letter, the letter must be capitalized. For example, the English Wikipedia article for `iPad` actually matches this URL:

   ```
   https://en.wikipedia.org/wiki/IPad
   ```

   Filter out all page titles that start with lowercase English characters. You may notice that some page titles don't start with English letters but digits, symbols, lowercase characters in other languages, etc., **DO NOT** filter them.

   **Be cautious and read the documentation whenever you want to use any built-in utility in your chosen language**, for example, read the Javadoc carefully if you want to use `Character#isLowerCase(char ch)`.

6. **Miscellaneous filename extensions**: Despite having already filtered pages in `File:` or `Media:` namespace, you may still get files instead of articles.

   - Media file names are case-sensitive, for example, `picture.jpg` and `picture.JPG` are not identical files. Nevertheless, we should use **case insensitive** matching to filter all the media files, ending with any of these suffixes: `png, gif, jpg, jpeg, tiff, tif, xcf, mid, ogg, ogv, svg, djvu, oga, flac, opus, wav, webm`
   - A favicon (short for favorite icon), is a file containing one or more small icons. Browsers that provide favicon support typically display a page's favicon in the browser's address bar (sometimes in the history as well) and next to the page's name in a list of bookmarks. Originally, the favicon was a file called favicon.ico placed in the root directory (e.g., http://en.wikipedia.org/favicon.ico) of a web site. Therefore, we should exclude any files with `.ico` suffix.
   - The robots exclusion standard, also known as the robots exclusion protocol or simply `robots.txt`, is a standard used by websites to communicate with web crawlers and other web robots. The standard specifies how to inform the web robot about which areas of the website should not be processed or scanned. Robots are often used by search engines to categorize web sites. Exclude any files with the `.txt` suffix.

   Here is the filename extension blacklist (**case-insensitive**):

```
.png
.gif
.jpg
.jpeg
.tiff
.tif
.xcf
.mid
.ogg
.ogv
.svg
.djvu
.oga
.flac
.opus
.wav
.webm
.ico
.txt
```

7. **Wikipedia Disambiguation**: Disambiguation in Wikipedia is the process of resolving conflicts when one article title can have different meanings in different fields or scenarios.

   For example, the word "CPU " can refer to a computer's central processing unit, a human enzyme, software updates in Oracle products such as the Oracle Database and Java and others.

   Filter all disambiguation pages with the suffix `_(disambiguation)` (**case insensitive**), suffixes like `_%28disambiguation%29` should have already been decoded at this point. Don't filter any page with a specific topic such as `Numb_(Linkin_Park_song)`.

8. **Special pages**: Finally, there are some special pages to exclude as well. Page titles that are exactly (**case sensitive**) in the following list should be excluded:

   ```
   404.php
   Main_Page
   -
   ```

   - `Main_Page` is the main entrance of the Wikipedia site.
   - `404.php` is caused when a user attempts to follow a broken or dead link.
   - `-` is a single hyphen-minus character, why does this get many accesses every hour? If you read the Java source code of Pageview and you will find the clue: `-` is used whenever any unknown project or article is encountered.

**Output format**: Output the remaining articles in the following format:

```
[page_title]\t[count_views]
```

Where `\t` is a *tab*.

For example:

```
  Carnegie_Mellon_University  34
```

Name the output file exactly `output`, and follow these rules strictly:

- **If there are records from both desktop and mobile sites for the same page title, sum the accesses into one record.**
- Sort the output in **descending numerical order** of the number of accesses
- Break ties by ascending **lexicographical** order (based on the Unicode value of each character in the strings) of page titles. You can find the Order of Common Characters offered by the Wikipedia Manual. You can just use `String.compareTo(String anotherString)` in Java or `sorted()` in Python, while some tricks are needed if you want to use `sort` in Bash, more specifically, please figure out the usage of `LC_ALL=en_US.UTF-8` and `LC_ALL=C`.

To get a full understanding of the big picture, **please continue reading the writeup and finish the "Data Analysis" section before you start coding.**

Danger

# Be cautious about implicit reliance on your environment

Your code should work well and its behaviors should be consistent on different systems. You are allowed write and test the code on your own laptop first. **To help you learn best practices, we will test your code in a different environment.**

**If your code seems to run well locally but behaves differently upon submission, read this panel carefully before you create posts on Piazza.**

Failing to realize the potential difference among development, testing and production environments can lead to pitfalls. If your code behaves well in your development environment, it does not guarantee that your code will work perfectly in other environments.

You should make your code independent from unpredictable/uncontrollable external environments. Please pay attention to **encoding-aware I/O operations, newlines and locale** in your code and always **handle them explicitly instead of relying on the system default setting**. Besides, watch out for **versions and absolute/relative paths**. All the following topics can be pitfalls of this project.

## Locale

On POSIX platforms such as Linux, locale identifiers are defined in this format:

```
    [language[_territory][.codeset][@modifier]].
```

You can get the locale setting on your machine with command `locale`.

Locale on a Linux system can determine the default encoding in locale-aware programs. In addition, locale is also an important

topic in Internationalization/Localization, which will make a difference in date, time, number, currency and so on. In this project, we will explore how the default encoding may change the behavior of programs and the practice to make the programs independent from the system default encoding.

## Encoding-aware I/O

Let us start with File I/O.

```
BufferedWriter bw = new BufferedWriter(new FileWriter(OUTPUT));
```

This seems good, and it will work correctly on your own laptop in most cases.

If you run Findbugs upon this snippet, you will get this SEVERE warning:

```
Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and
will assume that the default platform encoding is suitable. This will cause the application behav
ior to vary between platforms. Use an alternative API and specify a charset name or Charset objec
t explicitly.
```

Similarly, the following snippet in python 2/3 relies on the value returned by `locale.getpreferredencoding(False)`:

```
with open(fname, "r") as f:
```

This bug can be easily ignored when you only run and test your code in your own development environment, in which the default encoding will be set to UTF-8. However, if the production environment has another default encoding, your code will produce unexpected output. If you see weird output like `????`, replacement characters or empty boxes in the future, check your encoding! Many traditional encodings will change unsupported code points into question marks, such as ISO-8859-1, a.k.a. Latin-1. Failing to deal with encoding when handling input can cause more severe failure, as the program may crash. For example, processing UTF-8 encoded input as ASCII in Python 3 may cause `UnicodeEncodeError: 'ascii' codec can't encode character: ordinal not in range(128)`.

You must set encoding explicitly when your program converts an input stream of bytes to strings, and when your program converts strings to an output stream of bytes. **You CANNOT pass all the test cases if you fail to make your program encoding-aware.**

Here are examples to handle encoding in an explicit fashion:

```
For example, in Java 8:

    BufferedReader br = new BufferedReader(
            new InputStreamReader(new FileInputStream(INPUT), StandardCharsets.UTF_8));
    PrintWriter printWriter = new PrintWriter(new File(OUTPUT), "UTF-8");
    Stream<String> stream = Files.lines(
            Paths.get(OUTPUT), StandardCharsets.UTF_8)

In Python 3:

    # for more information, read https://docs.python.org/3/howto/unicode.html
    with open(fname, "rt", encoding='utf-8') as f:

In Python 2.7:

    # for more information, read https://docs.python.org/2/howto/unicode.html
    with io.open(fname, "wt", encoding='utf-8') as f:
```

Okay, let's try another example! Will the following code produce the same output in multiple environments?

```
System.out.println(str);
```

No! Even `System.out`, as a `PrintStream`, is system dependent! To help you overcome this hurdle and pass all the tasks in this project, we offer you a best practice here:

```
Scanner in = new Scanner(
        new BufferedInputStream(System.in), "UTF-8");
PrintWriter out = new PrintWriter(
        new OutputStreamWriter(System.out, "UTF-8"), true);
```

If you cannot get a full score upon submission and you wonder if it is caused by encoding, you may test your program on your instance by running your program with different locales:

```
LC_ALL=en_US.UTF-8 ./your_program  # `locale charmap` will return `UTF-8`
LC_ALL=C ./your_program            # `locale charmap` will return `ANSI_X3.4-1968` (ascii)
```

Encoding-awareness covers not only I/O but also the source code. If there are UTF-8 characters in the source code, including the comments, the Java compiler can break if the system default encoding does not support UTF-8. You should set the source code encoding using `javac -encoding utf8 *.java` or use the Maven approach. Although the source code encoding in Python is independent from system default encoding and Python 3 supports UTF-8 by default, Python 2 will default to ASCII and you must follow this Python Enhancement Proposal to define a source code encoding in Python 2.

If you are passionate about mining insight from data, keep in mind that encoding-unaware data processing may fail to produce the expected output **silently**. If you are enthusiastic about API design, please make your library encoding-aware. Even in one of the most widely-used utility projects, Apache Commons, the contributors feel the pain caused by encoding reliance. Apache Commons IO, after 12 years since the initial release and more than 4 years since the encoding reliance was reported, the library finally deprecates all the encoding-unaware methods in the 2.5 release. Encodings can be more vital when working with

web applications and databases, bear this in mind when you work on the team project in the future.

> "It does not make sense to have a string without knowing what encoding it uses." -- Joel Spolsky

## Newline (EOL)

Believe it or not, `System.out.println(str)` has yet another problem. Try your code on both Linux and Windows platforms, and compare the md5sum of the standard output -- and they will be different.

A newline, also known as end of line (EOL), is a special character or sequence of characters signifying the end of a line of text and the start of a new line. The actual codes representing a newline vary across operating systems, which can be a problem when exchanging text files between systems with different newline representations.

Systems based on ASCII or a compatible character set use either LF (Line feed, '\n'), CR (Carriage return, '\r'), or CR followed by LF (CR+LF, '\r\n'). Unix and Unix-like systems, e.g. Linux, Mac OS X, etc., use '\n' while Windows use '\r\n'.

You may edit your code on your own laptop during your development and testing phases, however, keep in mind if you are a Windows user, the code may seem "visually" correct but will behave differently on a Unix or Unix-like System. **Make sure you set the EOL to UNIX format in your editor, especially when you write bash scripts locally, or your scripts might break when you upload the script or you "copy & paste" the code to the remote instance.** You may use `cat -e filename` to make sure there is no CR (Carriage return, '\r') in your code (there will be `^M` at the end of each line if CR exists).

It is also recommended to always handle newlines explicitly in your code.

```
For example, in Java 8:

    //OS-dependent code
    printWriter.println(entry.getKey() + "\t" + entry.getValue());
    printWriter.printf("%s\t%s%n", entry.getKey(), entry.getValue());
    printWriter.print(entry.getKey() + "\t" + entry.getValue() + System.lineSeparator());
    //System.lineSeparator() can be replaced with System.getProperty("line.separator");

    //OS-independent code with identical output in various operating systems
    printWriter.print(entry.getKey() + "\t" + entry.getValue() + "\n");

In Python 3:

    # OS-dependent code
    output = open('output', 'wt', encoding='utf-8')
    output.write(line + '\n')
    # Python automatically translates "\n" to the proper newline of the current OS.
    # it will be converted to "\r\n" in Windows

    # OS-independent code
    output = open('output', 'wt', encoding='utf-8', newline='\n')
    output.write(line + '\n')
```

## Versions & Compatibility

### Bash

If you want to use `awk` to do regex/string operations ignoring case, `IGNORECASE` seems to be a good idea. It can be set to a non-zero value.

```
awk 'BEGIN{IGNORECASE = 1;}...' FILENAME
```

The command will work well on our student AMI, but it will break on Mac OS X. Because `IGNORECASE` is implemented in GNU Awk (a.k.a. `gawk`) but not in Mac's `awk`. Alternatively, `tolower()` is supported by both versions.

Similarly, the BSD `grep` on Mac is different from the GNU `grep` on our student AMI. There is also no `wget` installed on Mac OS X and an "imperfect substitute" is `curl -O`. Consider using Homebrew if you need to install GNU tools on a Mac platform.

Whenever using tools among different systems, remember that there can always be various versions. For example, a lot of popular Unix-like tools have their GNU versions. Remember to check the versions first before you start using tools and choose more compatible solutions if possible.

### Python

If you are a Python user, specify your Python version explicitly.

Use either `python2` or `python3` to run python, do not use `python` and rely on the default one in the system environment. This also applies to the usage of `pip2` and `pip3` over `pip`. Alternatively, set the proper shebang line to specify the interpreter to use and run your script as an executable, such as `./script.py`. If you decide to use shebang, be extremely careful because some shebang lines can still break in different system environments, such as `#!/usr/bin/python3` because the python installation path may vary across platforms. `#!/usr/bin/env python3` is the best way to define the shebang for portability.

### Java

When you are trying to run a class compiled with Java 8 into a lower JRE, you will get "Unsupported major.minor version Error". We are using Java 8, GNU Awk and GNU grep in our course.

If you are in doubt about any versioning issue of tools/languages/libraries we support, please create a private post on Piazza.

## Absolute/Relative Paths

**DO NOT** write code like the example below because your code won't be able to execute anywhere other than the absolute path.

```
python3 /home/<andrewId>/Project1_1/script.py
```

Replace any absolute path with a relative one!

## Conclusion

To sum up, you should always pay attention to the potential causes of implicit reliance in your implementation, and it will help you a lot during **this project, this semester, and hopefully in your career**.

# Data Analysis

After filtering the data, you are expected to analyze your results and answer a set of questions, which are present in the file `/home/<andrewId>/Project1_1/runner.sh` on your instance. To complete this module, do the following:

1. Go to the project folder located at `/home/<andrewId>/Project1_1`
2. The project folder consists of the following files: the runner script `runner.sh`, `submitter` to submit your solutions and `reference` to cite the links and students you get help from, and other data files for you to work on. You have permissions to edit the `runner.sh` and `references` files.
3. Edit the script `runner.sh` to include the commands/code used to answer the questions. We recommend using bash scripting, but it is not required.

   Do not move any of the provided files.

   If you are using any external scripts, ensure that you are calling the correct scripts from `runner.sh`. Please ensure that you are placing all your code in the same folder and also assume that the dataset is present in the current folder.

   When you need to access the dataset in your code assume that it is present in the working directory (i.e., do not use any absolute or relative paths for accessing the dataset -- we will auto-grade it later in a single jailed environment).
4. Edit the text file `references` to include all the links that you referred to for completing this project. Also, include the Andrew IDs of all students who you might have discussed general ideas with when working on this project in the same file. This is extremely important.

   We analyze many aspects of your AWS and TheProject.Zone usage, as well as automated code similarity detection tools.

   **NOTE**: Citing resources and having big picture discussions with other students does not excuse cheating. **You are not allowed to look at or discuss code with another person. Similarly, you are not allowed to use any code snippet from anyone or anywhere on the Internet**.

   When in doubt, please post privately on Piazza.
5. You can run and check your answers by typing `./runner.sh` from the Project1_1 folder.

   Running this script should print out the answers to all the questions you have answered. Please ensure that the answers are printing correctly before you submit your answers.

   Do not print hardcoded answers in `runner.sh`. We will run your code on other log files for verification and grading.

   If you want to focus on one question and get a readable unescaped answer, we offer you `./runner.sh -r <question_id>` from the Project1_1 folder to run one single question. Type `./runner.sh -h` to get the usage example.

6. Once you have completed all the questions, you can submit the answers to the evaluation system using the `submitter` executable. Run the executable using the command `./submitter` from the auto-grader folder.

   Remember that your submission password can be found by clicking on the button on the top of this page. **Make sure you open port 80 for incoming HTTP traffic before you proceed.**

   Type `./submitter -h` to get the usage example.

7. After running the `./submitter` command, a website will be created with answers and values which are specific to your submission. Once your details are validated, these pages will be read by our load generators within a few minutes, at which point, you will see the feedback and scores in your submission table.

   The load generators also read your code and save them to our repository, where they will be strictly analyzed for similarities with submissions from other students in the past and present, as well as with code from the internet.

   There is no limit on the number of submissions allowed before the project deadline. However, submissions must be separated by at least 60 seconds.

   Make sure not to exceed the budget specified for this project module.

Information

### Progressively Solve Data Science Problems with Jupyter Notebook

We **strongly** recommend that you use `awk`, `grep` and Python Data Analysis Library to solve the data analysis questions in `runner.sh`. Please finish the Jupyter Notebook primer and the self-study interactive notebooks in the Azure Notebooks library 15-319/15-619: Cloud Computing Course, and you will be able to solve most questions quickly.

We have created the virtual environment at `/home/ubuntu/virtualenv/` on your instance with Pandas and Jupyter pre-installed, please use the following commands to run the server:

```
source /home/ubuntu/virtualenv/bin/activate
jupyter notebook --no-browser
```

You can now follow the guide in the "Remote Server Using SSH Tunneling" section in the Jupyter Notebook primer to build the connection.

When programming on Jupyter, please keep in mind that memory is very limited on a `t2.micro` machine, you should be aware of the memory usage, for example:

1. You should not read the whole dataset `pageviews-20161109-000000.gz` into memory.
2. You should not store duplicate filtered `output` in memory. Load it once and reuse it.
3. If you get any out-of-memory error, restarting the kernel will fix it.

### Notes

When submitting, please make sure the following source code is in your Project1_1 folder:

- Code to extract the namespace blacklist from the JSON file used in `json()` in `runner.sh`.
- Code to convert `pageviews-20161109-000000.gz` to `output` used in `filter()` in `runner.sh`.
- Code to use the `output` file to generate the answers in each data analysis question.
- Make sure that the source code is present (*.sh, *.java, *.py, etc.).
- Your code should demonstrate good style and discipline. Every time you write code, remember that someone else is going to read it. Make your code readable, self-explanatory and standardized, which will avoid many potential bugs. **Hard to read code of poor quality will lead to a loss of points during manual grading.**
- **Lack of comments, especially among complicate code, will lead to a loss of points during manual grading.**

Performance matters when processing big data, hence, the data filter program should not have a large overhead because the overhead will get amplified as the data size scales.

For example, although the Pandas library is the recommended tool in the data analysis task (since the filtered output is no longer that large), **DO NOT use Pandas in the data filter program.** The powerful features provided by Pandas is at the expensive cost of time and memory, which is not suitable for large dataset. We will explore the Spark Framework in Project 4 which enables you to process and analyze big data in a similar flavor as Pandas.

Danger

You are required to make your program efficient and get rid of any overhead, otherwise it will cause a "Grading timeout" with a 0 score.

1. Do not compile java code in `filter()`.
2. Do not use Pandas in the data filter program.

Besides, we block most internet connections during the grading.

1. Do not use any Maven remote repositories other than the *Maven central repository*.
2. Do not send any HTTP/HTTPS requests in your code.

If you get a "Grading timeout", do not simply resubmit without changing your code as the result will be the same.

Once again, if your submitted code does not produce the same results as the ones on your local computer, consider the topics mentioned in "Be cautious about implicit reliance on your environment".