How To Execute

Approach 1:
If you use IDE such as Intellij, you can run the integration test from **PatternResultGenIntegrationTest.java.**
Approach 2:
You can also run the main function in **util.TestExample.java**.
Approach 3:
I've added a gradle task to run TestExample from terminal. So you can go to the /title-pattern directory, and run command **./gradlew testExample**

All of these approaches will write the output dataset to files in /result directly. For question c), there will be 2 subdirectory in it, one for exact path matching, the other is for loosely matching. (Note the loosely matching example I used is for `OurPlanetTitle` -> `HomePage` path since this way we can clearly validate the results)

Design

The main purpose of this task is to find patterns among the input session visits dataset.

Table Schema

In order to properly serve the requested queries, I have defined 2 tables to store the intermediate computation results and other metadata. A brief description of the tables and their schemas are listed as follows. I'll go through the decision making process after the table description.

Visit Table

This table contains all the metadata of users' visits. It's created directly from parsing the input dataset, and used for Question a, b, d, e.

Table schema (Non-bold fields are from input dataset while bold fields are computed)

user_id	
session_id	
nagivation_page	
url	
date	

hour	
timestamp	
page_id	Id generated from page name and url, mainly used for path tracking
page_type	{LOGIN, HOME, ORIGINALS, TITLE}
prev_page_name	Name of the page that the user has previous visited
prev_page_type	Type of the page that the user has previous visited
step_no	The number of pages that user has already visited within the session
total_steps	The number of total steps in this session

Path Table

Description:
This table contains information of order of user's visit the website in the single session. Keyed by both user id and session id, each entry contains all the ids of the pages that user visited in this session, ordered by timestamp of the visit.

This table is specifically used to check whether users visited pages in a certain order, or followed the exact path.

user_id	
session_id	
path	Represents the path of a user's visits with in one session. This is a string field, which consists of all ids of the pages that user visited in this session, ordered by timestamp of the visit.

Thinking Process

1) Regarding visit table, I've been choosing between having visit table as described above or having the page information as a separate table. The reason I didn't model page as a separate table is because for most of the queries, we would need to join visit and page table. It's ok for our current dataset, but when the page and visit datasets get larger, the performance will be impacted. Plus the metadata of pages are relatively small.
2) Prev_page_name and page_type in visit table are not needed for the given 5 use cases. I keep them there so we can flexibly deal with other queries as well.

3) Regarding path table, currently it's mainly for question c), we can use it for question e) as well. I was choosing between having the path table separately or combining the information into visit table or calculating it on the fly. The reason I decided to model it separately is that the path itself could be very long, putting this information into visit table could make each row of visit table huge, thus impact the performance for other purposes. Plus path pattern finding itself sounds a very interesting area to explore.
4) In order to make the path shorter, I introduced page_id, which is used to represent one page in the path instead of the name of the page. Currently I used a in-memory map to store the id and name mapping, but have a comment there that we could push this logic into database and add index and cache on top of it in the future.
5) Another interesting thought for question c) came across to me but I didn't use it in the end is that we could generate a small dataset for the path on the fly, use it to broadcast join with visit table, and then calculate the diff between step_no. For example, if the path is home -> originals -> onePlanetTitle, the small dataset could be

page_name	path_no
home	1
originals	2
onePlanetTitle	3

Then use this dataset to joining with visit table on page_name, and sort the result by path_no within one session. Then the diff between step_no in the original visit table can be used to check if the session is followed the pattern exactly or loosely or not at all. But this approach is only valid when the path has no repeated pages. Our question c) is not valid for this.