PDS2

Thanks For The Memberies

Advanced filtering and editing of data sets and members

9 steps 45 minutes

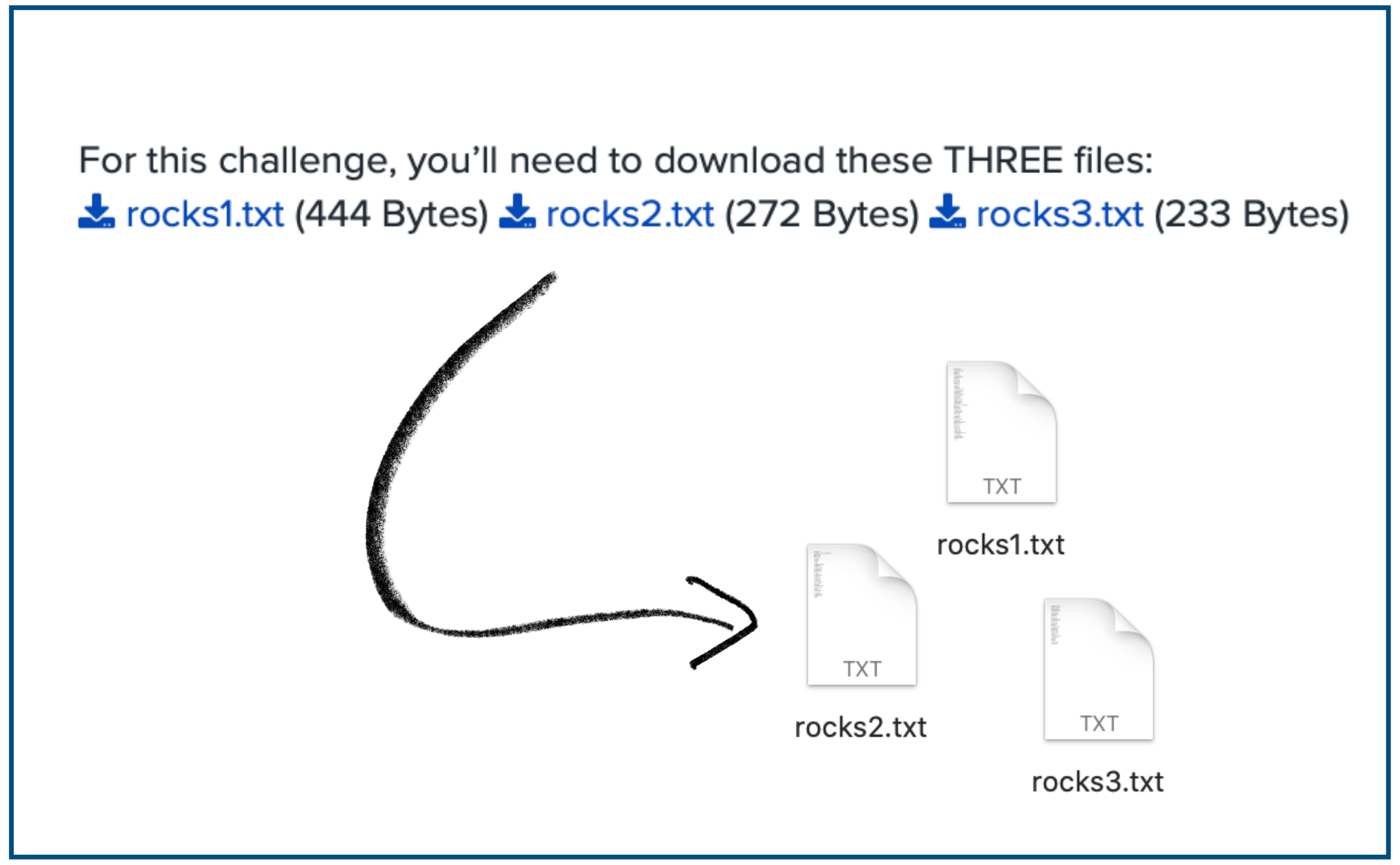
THE CHALLENGE

In z/OS, data is typically organized in structures called Data Sets. The concept is similar to the way you use files and folders on your personal computer, with a few very important differences, which you'll learn here.

In this challenge, you'll perform some basic operations around **Data Sets** and **Members**, and when you're done, you'll run a job to process some of those data set members.

BEFORE YOU BEGIN

Make sure your Visual Studio Code environment is all set up and connected to the z/OS system. Other than that, nothing else is required!

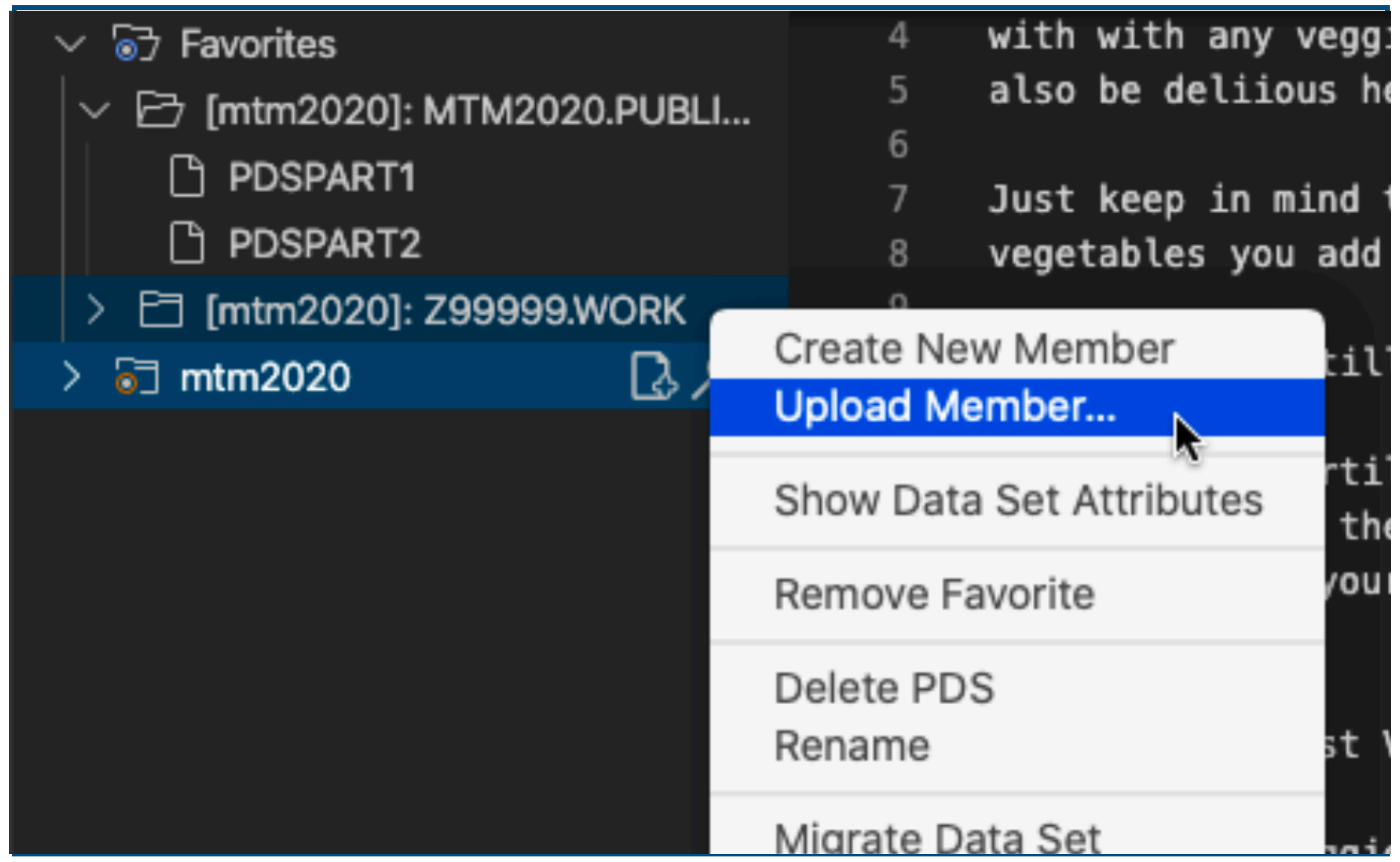


For this challenge, you'll need to download these THREE files:
rocks1.txt (444 Bytes) rocks2.txt (272 Bytes) rocks3.txt (233 Bytes)

1. THESE ROCK FILES ROCK

Download the PDS2 files associated with this assignment to your computer in a convenient location.

If you need help finding them, use this link: https://mtm.masterthemainframe.com/forum/t/thanks-for-the-memberies-pds2/31 and look for the files shown above.



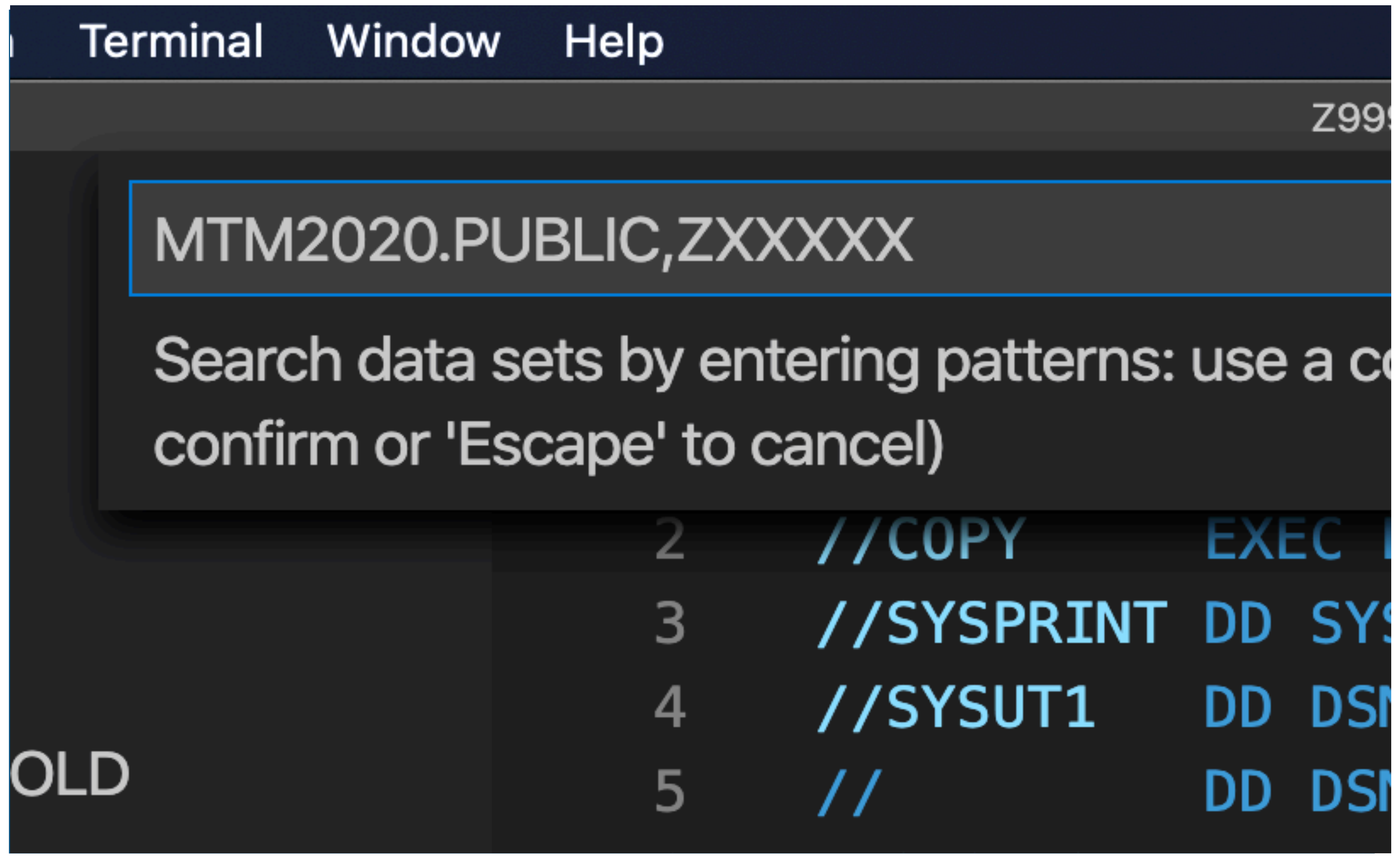
```
with with any vegg
also be deliious he

Just keep in mind
vegetables you add
```

2. UPLOAD TO 'WORK'

Set your HLQ to your WORK PDS. Right-click on the WORK data set and select "Upload member...". You should be able to select multiple files, so go ahead and select all of the PDS2 files you just downloaded and upload them to the WORK data set. Remember you may need to refresh the data set view before they show up.

In the screenshot above, the WORK PDS has been added to the list of favorites. If you made it a favorite in the previous challenge, you can find it there. If not, just change your HLQ filter to find it.



MTM2020.PUBLIC,ZXXXXX

Search data sets by entering patterns: use a c
confirm or 'Escape' to cancel)

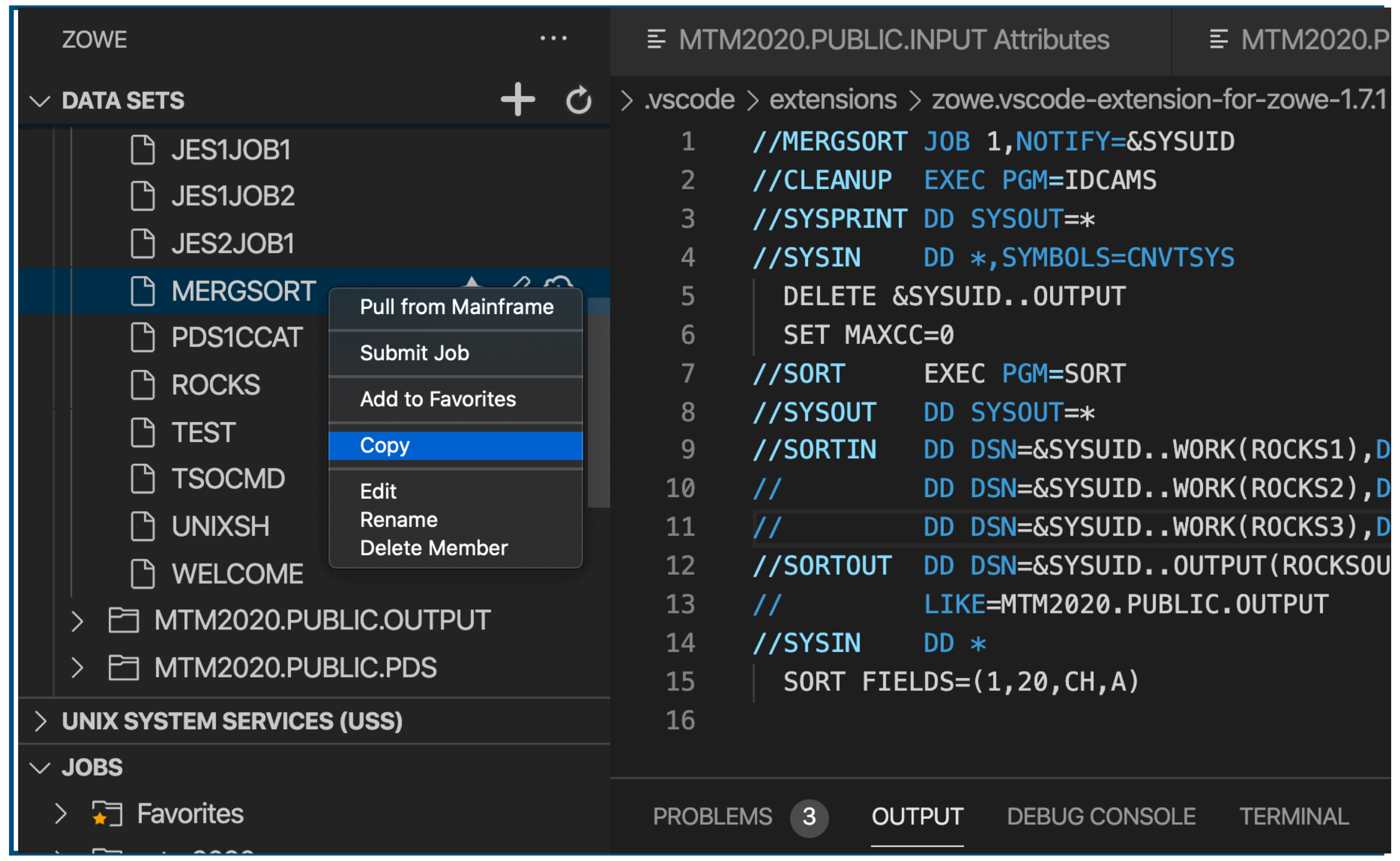
```
//COPY      EXEC
//SYSPRINT  DD SY
//SYSUT1    DD DSN
//          DD DSN
```

3. ADVANCED FILTERING

Click the HLQ filter button and notice the option for "Create a new filter". A filter can contain multiple patterns, including an asterisk (*) for wildcard matching.

Set your HLQ filter to **MTM2020.PUBLIC, ZXXXXX.** This is two entries, separated by a comma. Make sure you click on "Create a new filter" if the auto-complete is trying to spell out something longer that you don't want. Guess what will happen with this filter, then turn to the next step.

PDS2 Advanced filtering and editing of data sets and members

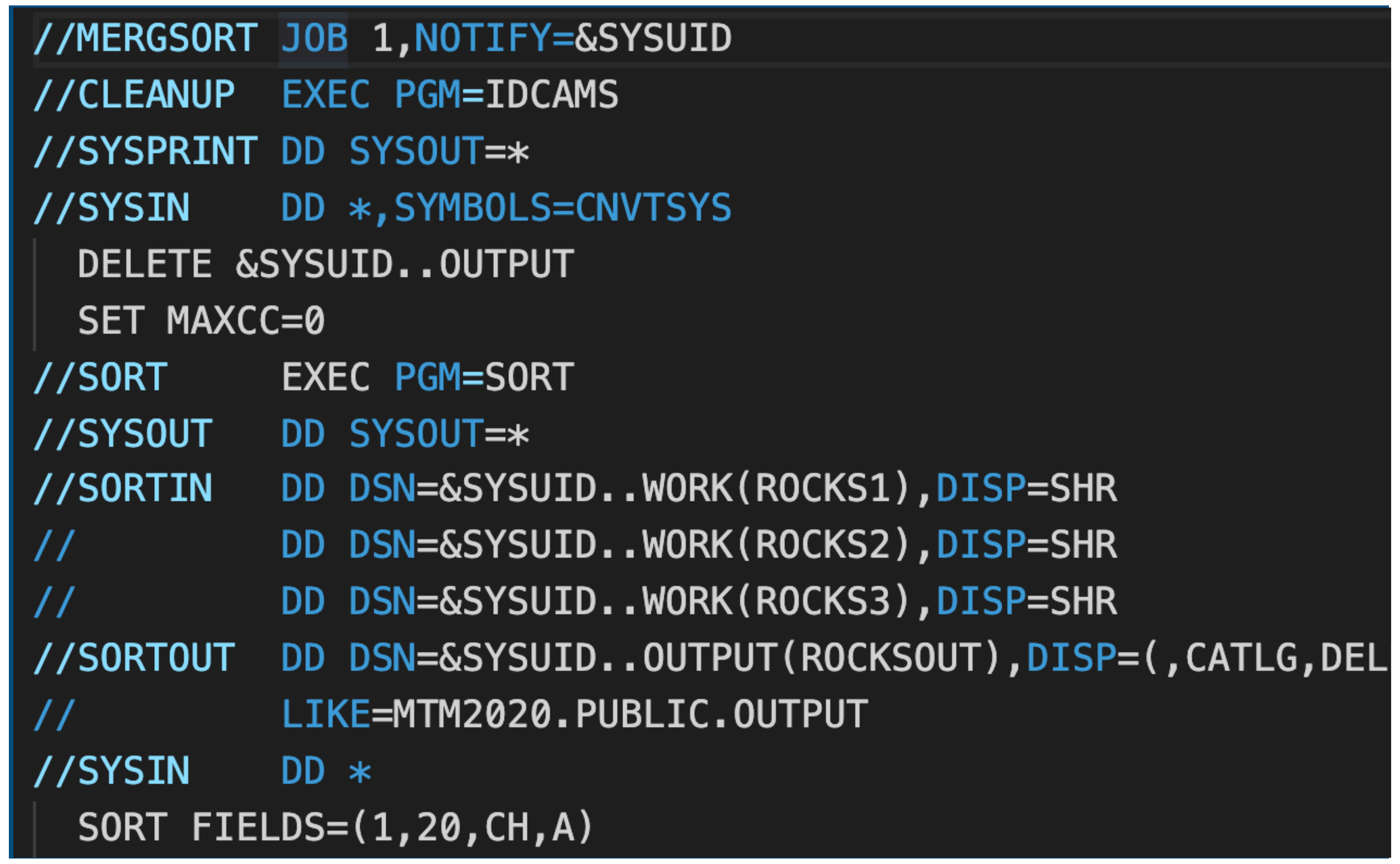


```
1   //MERGSORT JOB 1,NOTIFY=&SYSUID
2   //CLEANUP  EXEC PGM=IDCAMS
3   //SYSPRINT DD SYSOUT=*
4   //SYSIN    DD *,SYMBOLS=CNVTSYS
5     DELETE &SYSUID..OUTPUT
6     SET MAXCC=0
7   //SORT     EXEC PGM=SORT
8   //SYSOUT   DD SYSOUT=*
9   //SORTIN   DD DSN=&SYSUID..WORK(ROCKS1),D
10  //         DD DSN=&SYSUID..WORK(ROCKS2),D
11  //         DD DSN=&SYSUID..WORK(ROCKS3),D
12  //SORTOUT  DD DSN=&SYSUID..OUTPUT(ROCKSOU
13  //         LIKE=MTM2020.PUBLIC.OUTPUT
14  //SYSIN    DD *
15    SORT FIELDS=(1,20,CH,A)
16
```

4. COPY THE JOB

Look for '**MERGSORT**' in the **MTM2020.PUBLIC.JCL** data set and copy that into your WORK data set. Once it's copied over, open it up and take a look at what's inside.

NOTE: Each qualifier (the part between the dots) of a data set name can be 1-8 characters in length, so we sometimes have to get creative with making names fit into 8 characters. If you're having trouble copying MERGSORT, make notice of how it's spelled.

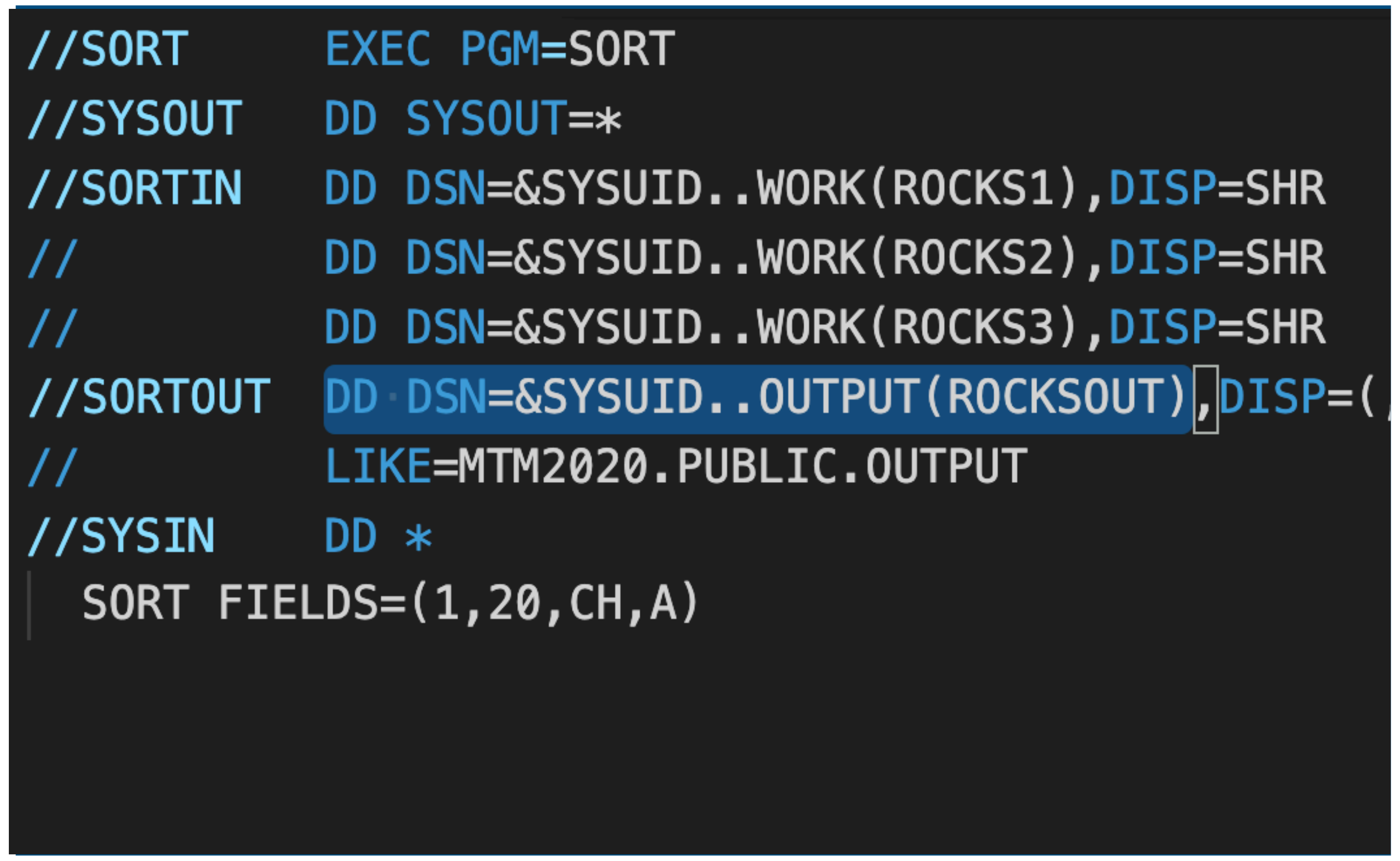


```
//MERGSORT JOB 1,NOTIFY=&SYSUID
//CLEANUP  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *,SYMBOLS=CNVTSYS
  DELETE &SYSUID..OUTPUT
  SET MAXCC=0
//SORT     EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=&SYSUID..WORK(ROCKS1),DISP=SHR
//         DD DSN=&SYSUID..WORK(ROCKS2),DISP=SHR
//         DD DSN=&SYSUID..WORK(ROCKS3),DISP=SHR
//SORTOUT  DD DSN=&SYSUID..OUTPUT(ROCKSOUT),DISP=(,CATLG,DEL
//         LIKE=MTM2020.PUBLIC.OUTPUT
//SYSIN    DD *
  SORT FIELDS=(1,20,CH,A)
```

5. INVESTIGATE THE CODE

Figure out what the code is doing. A big hint is that when the system sees &SYSUID, it will substitute in your userid, so **&SYSUID..ROCK1** would get read as **Z99999.ROCK1**, except with your own userid.

These are known as Symbolics, and they come in handy when writing scripts where certain values need to change from user to user or system to system.



```
//SORT     EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=&SYSUID..WORK(ROCKS1),DISP=SHR
//         DD DSN=&SYSUID..WORK(ROCKS2),DISP=SHR
//         DD DSN=&SYSUID..WORK(ROCKS3),DISP=SHR
//SORTOUT  DD DSN=&SYSUID..OUTPUT(ROCKSOUT),DISP=(
//         LIKE=MTM2020.PUBLIC.OUTPUT
//SYSIN    DD *
  SORT FIELDS=(1,20,CH,A)
```

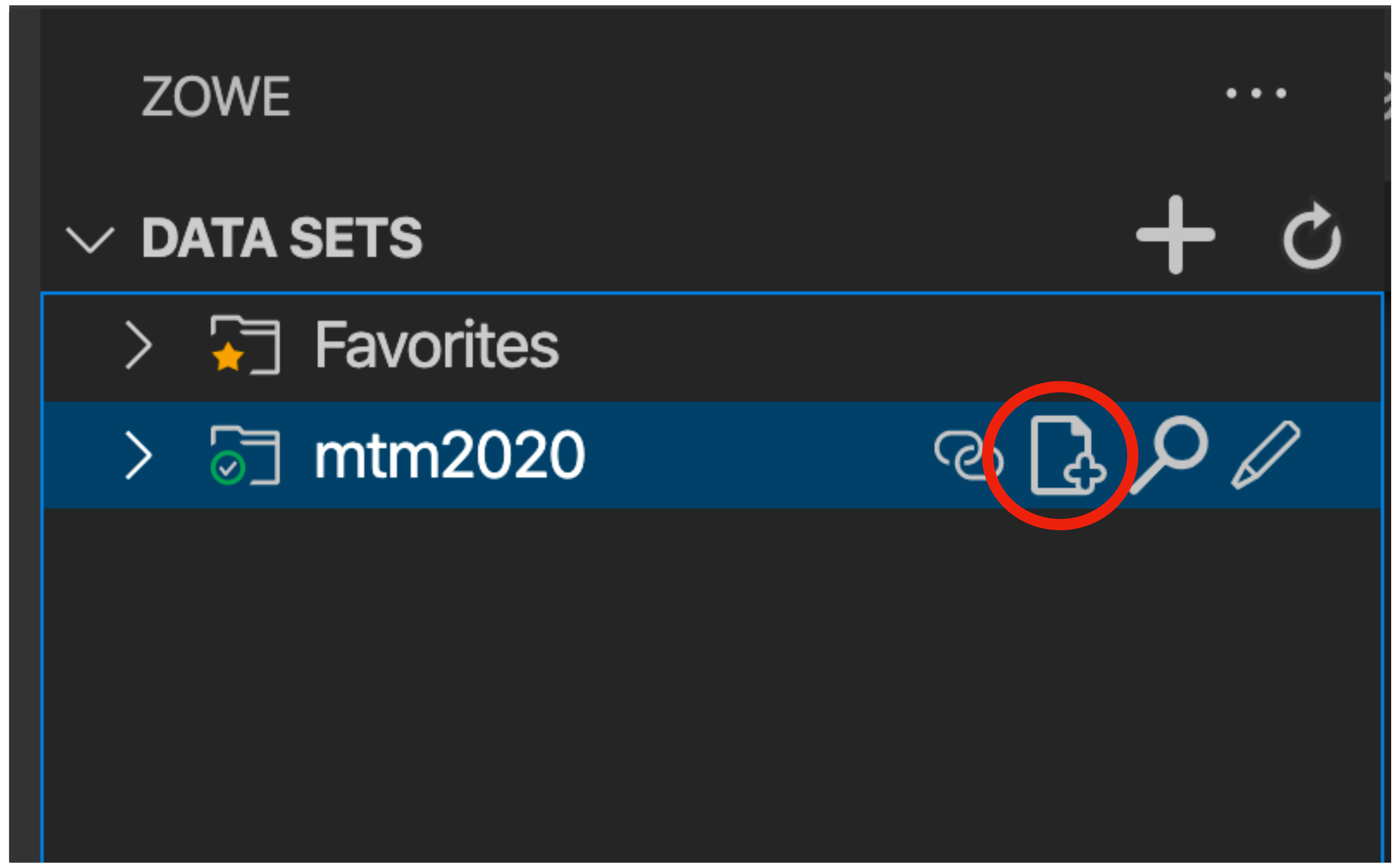
6. PREDICT THE OUTPUT

Look at line #8 of the program and notice where it's putting its output. Can you guess what the data set member name will be after the Symbolic substitution takes place?

JOBS, JCL, AND DATA SETS

While this is the data sets lesson, and you may not have gotten to the JCL challenges yet, you're getting some exposure to the concepts of Jobs and JCL. The way a lot of work happens in z/OS is by describing the task you want the computer to perform in a Job Control Language, or JCL. We typically use JCL to describe the programs we want run, the input and output to be used for them, and how we want everything to run. From there, the job is handed off to the Job Entry Subsystem, or JES, which handles running the JCL.

After the job has run, it also puts all of the relevant output into the output queues for you to look at. If something goes wrong, this will be your first step to figure out what didn't go as planned.



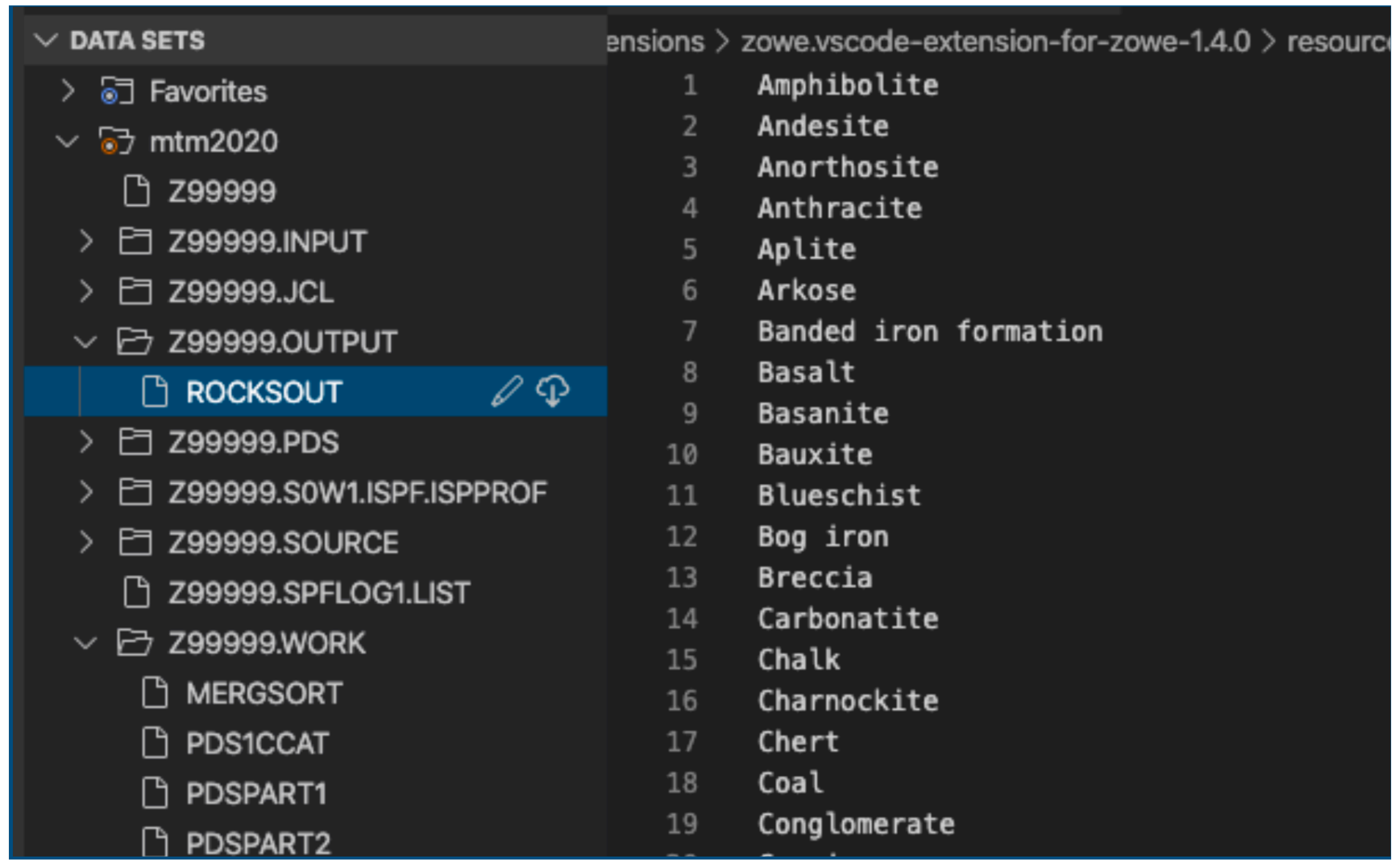
7. MAKE YOUR OWN DATA SETS

We have pre-allocated a number of data sets for you, like OUTPUT, INPUT, JCL and PDS. You can make your own data sets very easily right through VS Code.

Click on the "Create New Data Set" button next to your profile under Data Sets and select "Data Set Partitioned".

Give it a name that starts with your HLQ, followed by any second qualifier 8 characters or less. That isn't already used. For example, ZXXXXX.MYNEWPDS or ZXXXXX.PUMPKIN. *(substituting in your actual userid for the example)*

This step is not necessary to complete the challenge, but we figured by now, you're seeing so many data sets, you might want to see how easy it is to create them yourself.

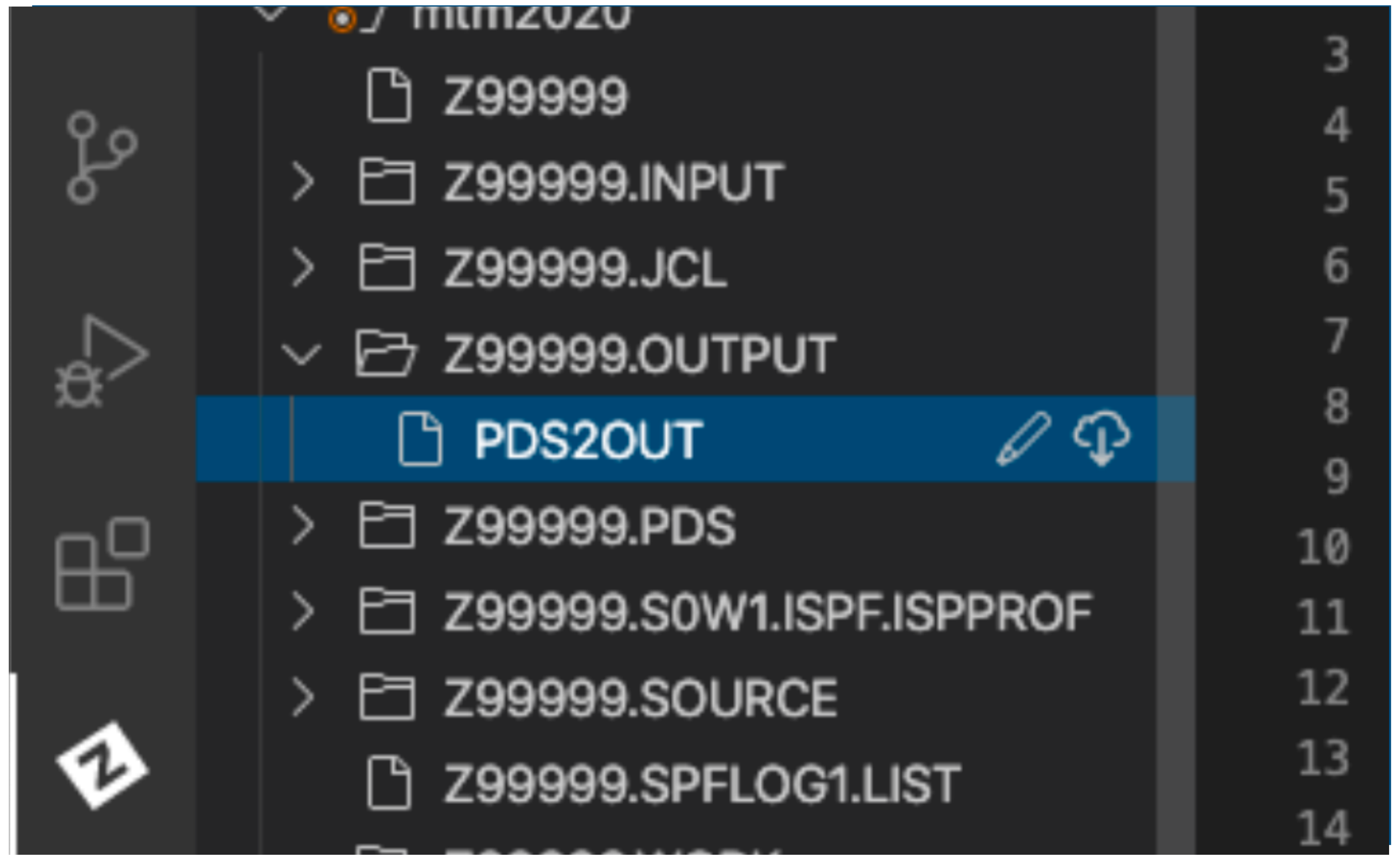


```
Amphibolite
Andesite
Anorthosite
Anthracite
Aplite
Arkose
Banded iron formation
Basalt
Basanite
Bauxite
Blueschist
Bog iron
Breccia
Carbonatite
Chalk
Charnockite
Chert
Coal
Conglomerate
```

8. RUN THE JOB

Submit the job the same way you ran the job in the earlier PDS challenge. Wait a moment, then check your newly-created OUTPUT data set. Remember that it may take a few refreshes to show up.

Click on the ROCKSOUT data set member and observe what the job did with the three input files.



9. GIVE IT A NEW NAME

Right-click on the newly-created member and select Rename. Give it the name of PDS2OUT.

All done? Think you got it?

Go into MTM2020.PUBLIC.JCL and look for a member named CHK. Then right-click on it and select "Submit Job" to mark it for validation.

ADVANCED DATA SET TIPS AND TRICKS

To keep things simple, Zowe hides a lot of the complexity behind the creation of data sets and members. In reality, there are many many different options being selected by default. You can change these by using the Zowe-Default-Datasets options in settings.json, with a full walkthrough of this and a few other nice tricks described in this article:

How to Automate z/OS Data Set Allocation Using Zowe & JSON

NEXT UP...

Make some tacos? Tend to your rock collection? The possibilities are endless, but now is a great time to jump into JCL, and learn just how programs run in z/OS.

Want to talk? Join our Slack ibm.biz/mtm_slack

Tweet about it! #MasterTheMainframe