Question-1:

Pull any image from the docker hub, create its container, and execute it showing the output.

docker pull <image_name>: It pulls images from registry or downloads docker images from internet.

Let us pull MySQL image from Docker.

```
C:\Users\vaishnavi>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
197c1adcd755: Pull complete
45f2e353f7d2: Pull complete
68ec6ece42ef: Pull complete
cfa4d9a7b88e: Pull complete
64cab5858b1d: Pull complete
92fcd248d982: Pull complete
88635e83312d: Pull complete
43f0427259d9: Pull complete
79828698a290: Pull complete
a8854781893e: Pull complete
6c8bdf3091d9: Pull complete
Digest: sha256:8653a170e0b0df19ea95055267def2615fc53c62df529e3750817c1a886485f0
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

Now create a container for MySQL using "**docker create mysql**".

```
C:\Users\vaishnavi>docker create mysql
218dcf74ed43a91b27ed6139e50d239b6a3433d0718c7aa63b9f7aae8817f5fb
```

Docker run <image_name>: This command searches for the specified image, if found then it executes otherwise it downloads and then executes the image using container.

```
C:\Users\vaishnavi>docker run mysql
2023-02-19 09:04:02+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
2023-02-19 09:04:03+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2023-02-19 09:04:03+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
2023-02-19 09:04:03+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specified
    You need to specify one of the following as an environment variable:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_PASSWORD
```

Question-2:

Create the basic java application, generate its image with necessary files, and execute it with docker.

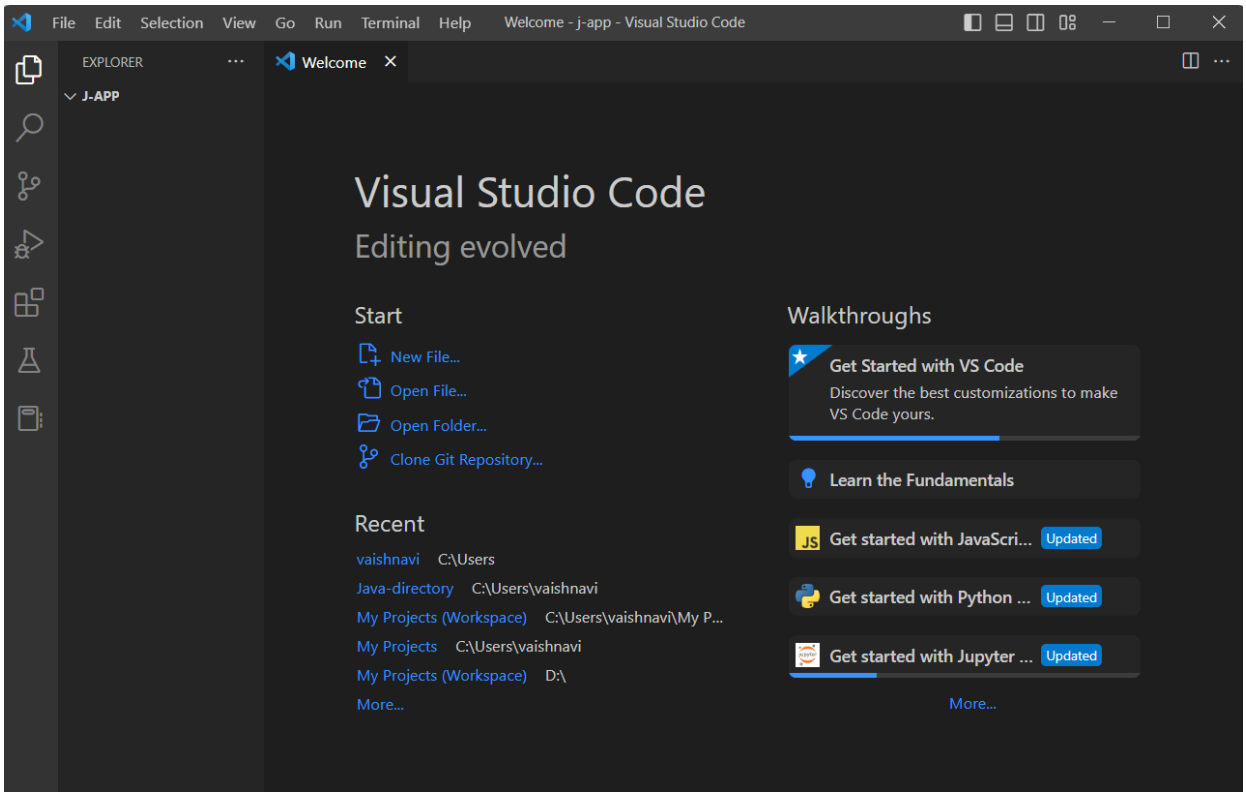
Creating a java application and executing it with docker can be done as follows.

1.First create a new directory using "**mkdir <directory_name>".** Then switch to the directory and open Visual studio code using "**code .**" .

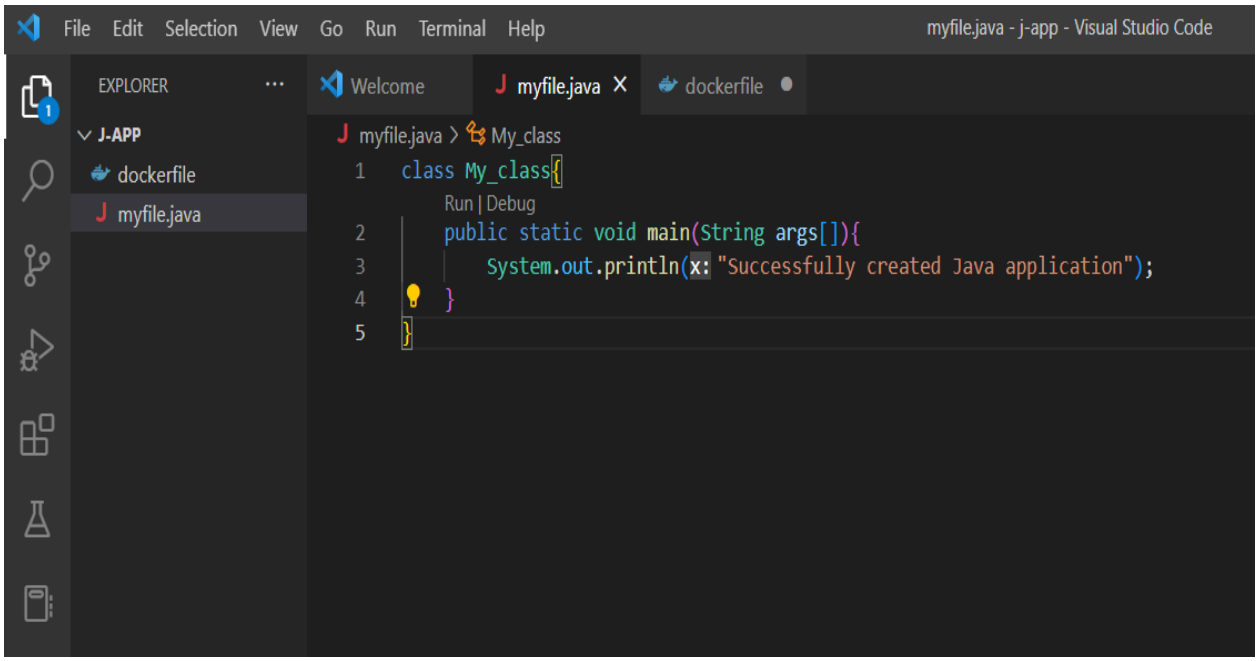
I have created a directory called j-app.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ mkdir j-app

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ cd j-app

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/j-app (master)
$ code .
```

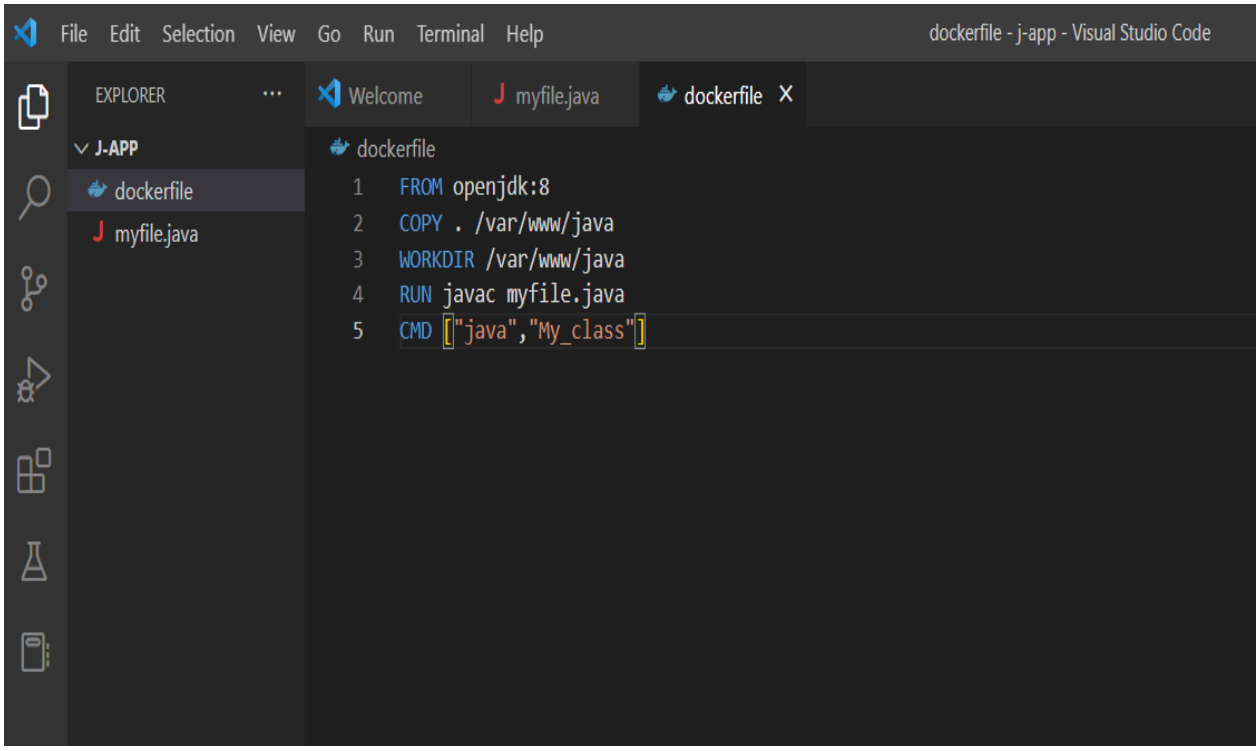
Visual Studio Code will be opened as follows.



2.Now create a java file “myfile.java” inside the directory.



3.Create a Docker file “dockerfile” with required instructions for the docker



4.Now build the image in the “j-app” directory .Use “**docker build -t <image_name> .**”

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/j-app (master)
$ docker build -t j-image .
#1 [internal] load build definition from Dockerfile
#1 sha256:f549e9fa6464c07ee1f2eb4ff9f22408ad8c293b7d61356f86645b9e532f4346
#1 transferring dockerfile: 31B 0.0s done
#1 DONE 0.1s

#2 [internal] load .dockerignore
#2 sha256:af31aa2e7e27c99c9ad2761308b13cec7e884825ff8ba9bc60a61f4a203a914e
#2 transferring context: 2B done
#2 DONE 0.1s

#3 [internal] load metadata for docker.io/library/openjdk:8
#3 sha256:c1006613b124ab347fbb29ae49e2ab62add271baf34bdfe7a7a4c383ac159b76
#3 DONE 1.8s

#4 [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625f
e8f150577f9eb58bd937f5452cb8
#4 sha256:1e7d9e224eeb34ef733a8ab6274c72dbf6d09407a6df09d7e6001657f4d7ee92
#4 DONE 0.0s

#5 [internal] load build context
#5 sha256:9088a7dff5600091ba924e9f1fe6bfcadba8752c9394084bd10e9463bd97523b
#5 transferring context: 213B done
#5 DONE 0.0s

#4 [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625f
e8f150577f9eb58bd937f5452cb8
#4 sha256:1e7d9e224eeb34ef733a8ab6274c72dbf6d09407a6df09d7e6001657f4d7ee92
#4 CACHED

#6 [2/4] COPY . /var/www/java
#6 sha256:8963fbf9535fb25b3ac9e19eefda9efb05b289870a2e83d479daff53546dd678
#6 DONE 0.1s

#7 [3/4] WORKDIR /var/www/java
#7 sha256:a4f6ab8d38c8e14fa5312a08caa1c633ffaa4493887f5822ac754758111f44be
#7 DONE 0.1s

#8 [4/4] RUN javac myfile.java
#8 sha256:d323a7f48dc48273c7830a6d65f4dc21f8a830afa45e342a75805fa36959dd14
#8 DONE 1.0s

#9 exporting to image
#9 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#9 exporting layers
#9 exporting layers 0.2s done
#9 writing image sha256:047f18a7d00dfb7697259ae76789d6fb22a6ce7d67f57683fc9c420394f77
3fe 0.0s done
#9 naming to docker.io/library/j-image done
#9 DONE 0.2s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
how to fix them

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/j-app (master)
```

5.Now run the image using “**docker run <image_name>**”.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/j-app (master)
$ docker run j-image
Successfully created Java application

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/j-app (master)
$
```