Module 6 Quiz **(Mayur Brijwani)**

Q1. Refactoring

Before we can start covering our code with tests, we need to refactor it. We'll start by getting rid of all the global variables.

- Let's create a function `main` with two parameters: `year` and `month`.
- Move all the code (except `read_data`) inside `main`
- Make `categorical` a parameter for `read_data` and pass it inside `main`

Now we need to create the "main" block from which we'll invoke the main function. How does the `if` statement that we use for this looks like?

ANS:

```
if __name__ == "__main__":
    main()
```

- **<u>Creating virtual environment</u>**

```
mlops-training(base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ pipenv install
Creating a virtualenv for this project...
Pipfile: /home/ubuntu/mlops-training/module-6/homework/Pipfile
Using default python from /home/ubuntu/anaconda3/bin/python (3.11.7) to create virtualenv...
⠋ Creating virtual environment...created virtual environment CPython3.11.7.final.0-64 in 666ms
  creator CPython3Posix(dest=/home/ubuntu/.local/share/virtualenvs/homework-O5Fku772, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/ubuntu/.local/share/virtualenv)
    added seed packages: pip==24.1, setuptools==70.1.0, wheel==0.43.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

✓ Successfully created virtual environment!
Virtualenv location: /home/ubuntu/.local/share/virtualenvs/homework-O5Fku772
Creating a Pipfile for this project...
Pipfile.lock (eaed16) out of date: run `pipfile lock` to update to (51d3e4)...
Running $ pipenv lock then $ pipenv sync.
Locking [packages] dependencies...
Locking [dev-packages] dependencies...
Updated Pipfile.lock (ed6d5d614626ae28e274e453164affb26694755170ccab3aa5866f093d51d3e4)!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (51d3e4)...
All dependencies are now up-to-date!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (51d3e4)...
```

- **Running Code**

batch.py | Dockerfile module-6/homework | Pipfile.lock | Pipfile .../homework | variables.tf .../ecr | Pipfile .../prac

module-6 > homework > batch.py

```python
def main():
    df_result = pd.DataFrame()
    df_result['ride_id'] = df['ride_id']
    df_result['predicted_duration'] = y_pred


    df_result.to_parquet(output_file, engine='pyarrow', index=False)


if __name__ == "__main__":
    main()
```

PROBLEMS OUTPUT TERMINAL PORTS 5 python - homework

```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ python batch.py 2023 03
predicted mean duration: 14.203865642696083
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$
```

- **Python Script**

```python
import sys
import pickle
import pandas as pd

def read_data(filename,categorical):

    df = pd.read_parquet(filename)
    df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
    df['duration'] = df.duration.dt.total_seconds() / 60
    df = df[(df.duration >= 1) & (df.duration <= 60)].copy()
    df[categorical] = df[categorical].fillna(-1).astype('int').astype('str')
    return df


def main():
    year = int(sys.argv[1])
    month = int(sys.argv[2])
    input_file = f'https://d37ci6vzurychx.cloudfront.net/trip-
data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    output_file = f'taxi_type=yellow_year={year:04d}_month={month:02d}.parquet'
```

```python
    with open('model.bin', 'rb') as f_in:
        dv, lr = pickle.load(f_in)

    categorical = ['PULocationID', 'DOLocationID']
    df = read_data(input_file,categorical)
    df['ride_id'] = f'{year:04d}/{month:02d}_' + df.index.astype('str')
    dicts = df[categorical].to_dict(orient='records')
    X_val = dv.transform(dicts)
    y_pred = lr.predict(X_val)
    print('predicted mean duration:', y_pred.mean())
    df_result = pd.DataFrame()
    df_result['ride_id'] = df['ride_id']
    df_result['predicted_duration'] = y_pred
    df_result.to_parquet(output_file, engine='pyarrow', index=False)

if __name__ == "__main__":
    main()
```

Q2. Installing pytest

Now we need to install `pytest`:

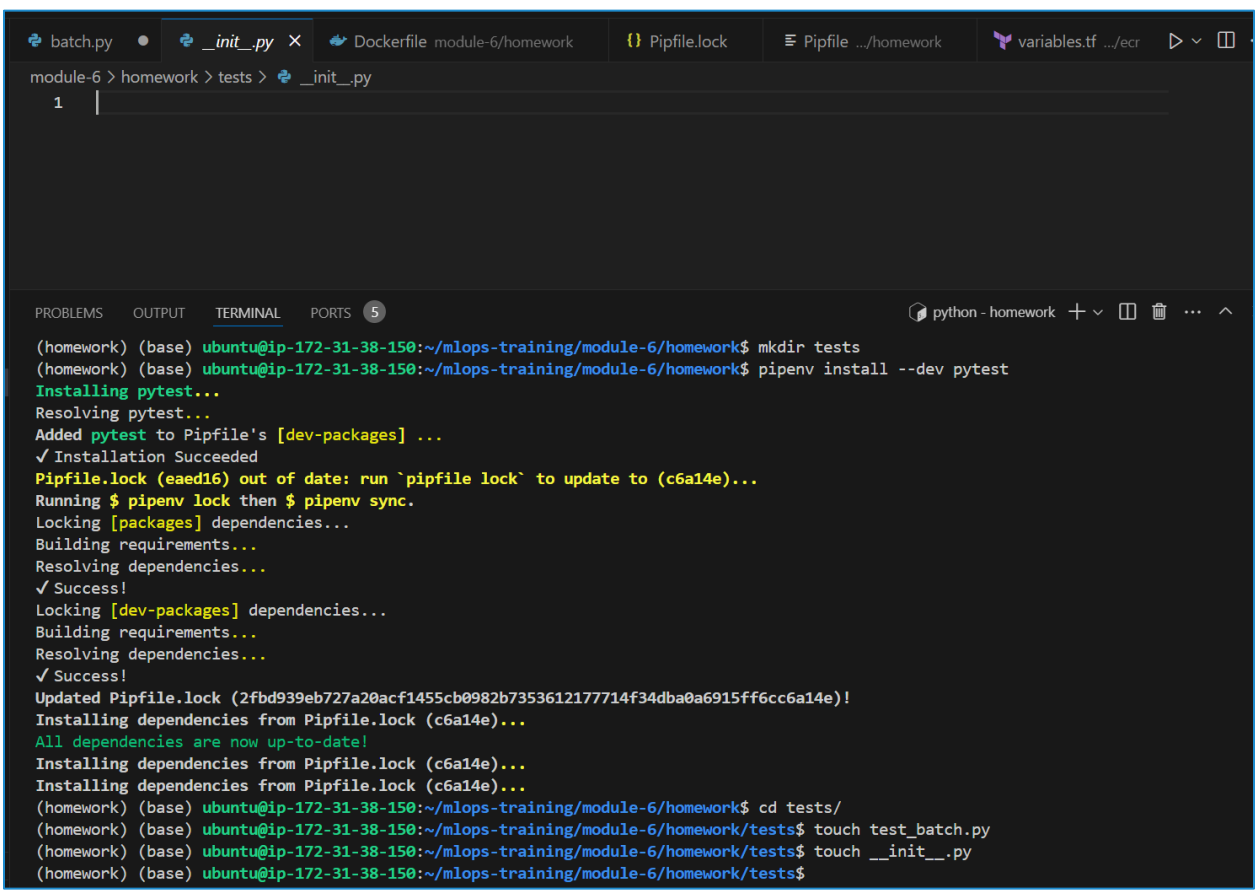
```
pipenv install --dev pytest
```

Next, create a folder `tests` and create two files. One will be the file with tests. We can name it `test_batch.py`.

What should be the other file?

Hint: to be able to test `batch.py`, we need to be able to import it. Without this other file, we won't be able to do it.

ANS: __init__.py



Q3. Writing first unit test

Now let's cover our code with unit tests.
We'll start with the pre-processing logic inside `read_data`.
It's difficult to test right now because first reads the file and then performs some transformations. We need to split this code into two parts: reading (I/O) and transformation.
So let's create a function `prepare_data` that takes in a dataframe (and some other parameters too) and applies some transformation to it.

How many rows should be there in the expected dataframe?

- 1
- 2
- 3
- 4

ANS : 2

- **Output of Test**

```
    return df

def test_data():
    actual_input = create_actual_input()
    categorical = ['PULocationID', 'DOLocationID']
    actual_output = prepare_data(actual_input, categorical)
    expected_output=create_expected_output()
    assert actual_output.to_dict() == expected_output.to_dict()
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS 5                                   python - homework
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ pytest tests/
========================================================== test session starts ==========================================================
platform linux -- Python 3.11.7, pytest-8.2.2, pluggy-1.5.0
rootdir: /home/ubuntu/mlops-training/module-6/homework
collected 1 item

tests/test_batch.py .                                                                                                          [100%]

========================================================== 1 passed in 0.41s ===========================================================
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$
```

- **Python Code**

```python
import pandas as pd
from datetime import datetime
from batch import prepare_data

def dt(hour, minute, second=0):
    return datetime(2023, 1, 1, hour, minute, second)

def create_actual_input():
    data = [
    (None, None, dt(1, 1), dt(1, 10)),
    (1, 1, dt(1, 2), dt(1, 10)),
    (1, None, dt(1, 2, 0), dt(1, 2, 59)),
    (3, 4, dt(1, 2, 0), dt(2, 2, 1)),
    ]
    columns = ['PULocationID', 'DOLocationID', 'tpep_pickup_datetime',
'tpep_dropoff_datetime']
    df = pd.DataFrame(data, columns=columns)
    return df

def create_expected_output():
    data = [
        ('-1', '-1', dt(1, 1), dt(1, 10), 9.0),
        ('1', '1', dt(1, 2), dt(1, 10), 8.0),
    ]
    columns = ['PULocationID', 'DOLocationID', 'tpep_pickup_datetime',
'tpep_dropoff_datetime', 'duration']
    df = pd.DataFrame(data, columns=columns)
    return df
```

```python
def test_data():
    actual_input = create_actual_input()
    categorical = ['PULocationID', 'DOLocationID']
    actual_output = prepare_data(actual_input, categorical)
    expected_output=create_expected_output()
    assert actual_output.to_dict() == expected_output.to_dict()
```

Q4. Mocking S3 with Localstack

Now let's prepare for an integration test. In our script, we write data to S3. So we'll use Localstack to mimic S3.

First, let's run Localstack with Docker compose. Let's create a `docker-compose.yaml` file with just one service: localstack. Inside localstack, we're only interested in running S3. Start the service and test it by creating a bucket where we'll keep the output. Let's call it "nyc-duration".

With AWS CLI, this is how we create a bucket:

```
aws s3 mb s3://nyc-duration
```

Then we need to check that the bucket was successfully created. With AWS, this is how we typically do it:

```
aws s3 ls
```

In both cases we should adjust commands for localstack. What option do we need to use for such purposes?

- `--backend-store-uri`
- `--profile`
- `--endpoint-url`
- `--version`

ANS: `--endpoint-url`

../ecr | batch.py | test_batch.py | docker-compose.yaml .../homework

module-6 > homework > docker-compose.yaml

```yaml
services:
  S3:
    image: localstack/localstack
    ports:
      - "4566:4566"
    environment:
      - SERVICES=s3
```

PROBLEMS OUTPUT TERMINAL PORTS 6

```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ aws --version
aws-cli/2.17.12 Python/3.11.9 Linux/6.8.0-1010-aws exe/x86_64.ubuntu.24
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ aws s3 mb s3://nyc-duration --en
dpoint-url=http://localhost:4566
make_bucket: nyc-duration
```

```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ aws s3 ls  --endpoint-url=http:/
/localhost:4566
2024-07-12 13:18:17 nyc-duration
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$
```

Modify our `read_data` function:

- check if `S3_ENDPOINT_URL` is set, and if it is, use it for reading
- otherwise use the usual way

```
import sys
import pickle
import pandas as pd
import os


def get_input_path(year, month):
    default_input_pattern = 'https://d37ci6vzurychx.cloudfront.net/trip-
data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    input_pattern = os.getenv('INPUT_FILE_PATTERN', default_input_pattern)
    return input_pattern.format(year=year, month=month)


def get_output_path(year, month):
    default_output_pattern = 's3://nyc-duration-prediction-
alexey/taxi_type=fhv/year={year:04d}/month={month:02d}/predictions.parquet'
    output_pattern = os.getenv('OUTPUT_FILE_PATTERN', default_output_pattern)
    return output_pattern.format(year=year, month=month)



def read_data(filename):
```

```
    S3_ENDPOINT_URL = os.getenv('S3_ENDPOINT_URL')
    if S3_ENDPOINT_URL is not None:
        options = {
            'client_kwargs': {
                'endpoint_url': S3_ENDPOINT_URL
            }
        }
        df = pd.read_parquet(filename, storage_options=options)
    else:
        df = pd.read_parquet(filename)
    return df


def save_data(df, output_filename):
    S3_ENDPOINT_URL = os.getenv('S3_ENDPOINT_URL')
    if S3_ENDPOINT_URL is not None:
        options = {
            'client_kwargs': {
                'endpoint_url': S3_ENDPOINT_URL
            }
        }
        df.to_parquet(output_filename, engine='pyarrow', index=False,
storage_options = options)
    else:
        df.to_parquet(output_filename, engine='pyarrow', index=False)



def prepare_data(df,categorical):

    df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
    df['duration'] = df.duration.dt.total_seconds() / 60
    df = df[(df.duration >= 1) & (df.duration <= 60)].copy()
    df[categorical] = df[categorical].fillna(-1).astype('int').astype('str')
    return df

def main():
    year = int(sys.argv[1])
    month = int(sys.argv[2])
    #input_file = f'https://d37ci6vzurychx.cloudfront.net/trip-
data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
```

```
    #output_file =
f'taxi_type=yellow_year={year:04d}_month={month:02d}.parquet'
    input_file = get_input_path(year, month)
    output_file = get_output_path(year, month)
    with open('model.bin', 'rb') as f_in:
        dv, lr = pickle.load(f_in)

    categorical = ['PULocationID', 'DOLocationID']
    df = read_data(input_file)
    df = prepare_data(df,categorical)
    df['ride_id'] = f'{year:04d}/{month:02d}_' + df.index.astype('str')
    dicts = df[categorical].to_dict(orient='records')
    X_val = dv.transform(dicts)
    y_pred = lr.predict(X_val)
    print('predicted mean duration:', y_pred.mean())
    df_result = pd.DataFrame()
    df_result['ride_id'] = df['ride_id']
    df_result['predicted_duration'] = y_pred
    save_data(df_result,output_file)


if __name__ == "__main__":
    main()
```

Q5. Creating test data

Now let's create `integration_test.py`

We'll use the dataframe we created in Q3 (the dataframe for the unit test) and save it to S3. You don't need to do anything else: just create a dataframe and save it.

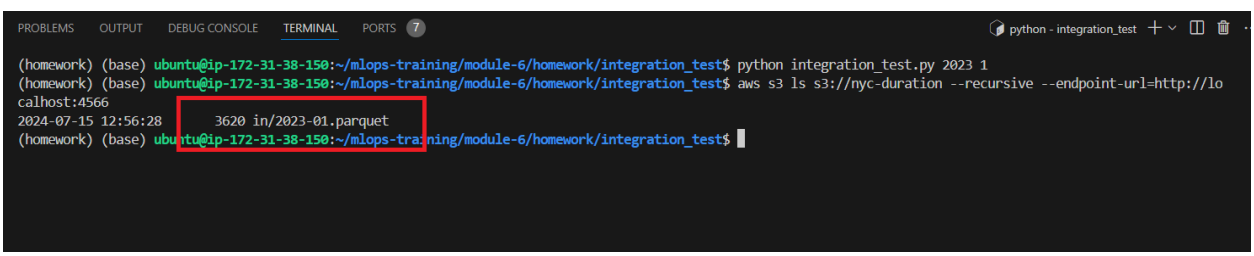
We will pretend that this is data for January 2023.

Run the `integration_test.py` script. After that, use AWS CLI to verify that the file was created.

What's the size of the file?

- 3620
- 23620
- 43620
- 63620

ANS: 3620




```
import os
import sys
from datetime import datetime
import pandas as pd
```

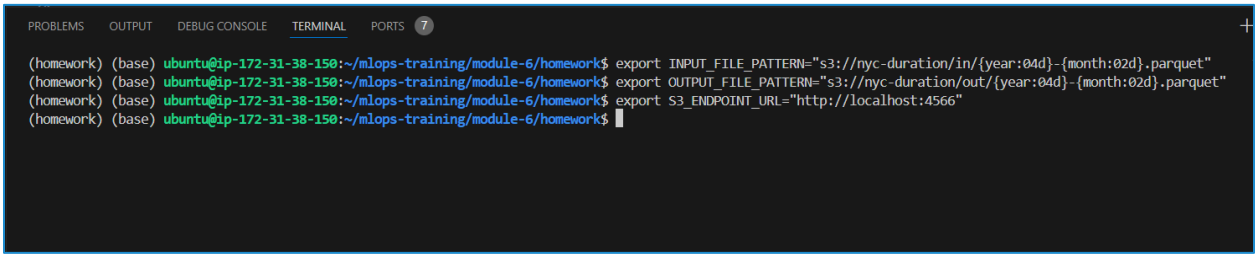
```python
def dt(hour, minute, second=0):
    return datetime(2023, 1, 1, hour, minute, second)



options = {
    'client_kwargs': {
        'endpoint_url': "http://localhost:4566"
    }
}
def create_data():
    year = int(sys.argv[1])
    month = int(sys.argv[2])
    data = [
        (None, None, dt(1, 1), dt(1, 10)),
        (1, 1, dt(1, 2), dt(1, 10)),
        (1, None, dt(1, 2, 0), dt(1, 2, 59)),
        (3, 4, dt(1, 2, 0), dt(2, 2, 1)),
     ]

    columns = ['PULocationID', 'DOLocationID', 'tpep_pickup_datetime',
'tpep_dropoff_datetime']
    df_input = pd.DataFrame(data, columns=columns)

    df_input.to_parquet(
        f"s3://nyc-duration/in/{year:04d}-{month:02d}.parquet",
        engine="pyarrow",
        compression=None,
        index=False,
        storage_options=options,
    )





if __name__ == "__main__":
    create_data()
```

Q6. Finish the integration test

We can read from our localstack s3, but we also need to write to it.
Create a function `save_data` which works similarly to `read_data`, but we use it for saving a dataframe.
Let's run the `batch.py` script for January 2023 (the fake data we created in Q5).
We can do that from our integration test in Python: we can use `os.system` for doing that (there are other options too).
Now it saves the result to localstack.
The only thing we need to do now is to read this data and verify the result is correct.
What's the sum of predicted durations for the test dataframe?

- 13.08
- 36.28
- 69.28
- 81.08

ANS: 36.28



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS  7

(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ python batch_q6.py
predicted mean duration: 18.138625226015364
Sum of prediction is  36.27725045203073
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$
```

```python
import os
import sys
from datetime import datetime
import pandas as pd

def get_input_path(year, month):
    default_input_pattern = 'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    input_pattern = os.getenv('INPUT_FILE_PATTERN', default_input_pattern)
    return input_pattern.format(year=year, month=month)


def get_output_path(year, month):
    default_output_pattern = 's3://nyc-duration-prediction-alexey/taxi_type=fhv/year={year:04d}/month={month:02d}/predictions.parquet'
    output_pattern = os.getenv('OUTPUT_FILE_PATTERN', default_output_pattern)
    return output_pattern.format(year=year, month=month)



def dt(hour, minute, second=0):
    return datetime(2023, 1, 1, hour, minute, second)


options = {
    'client_kwargs': {
        'endpoint_url': "http://localhost:4566"
    }
}
```

```python
def prediction():

    input_file = get_input_path(2023, 1)
    output_file = get_output_path(2023, 1)
    os.system('python batch.py 2023 1')
    prediction = pd.read_parquet(output_file, storage_options=options)
    print(f"Sum of prediction is  {prediction['predicted_duration'].sum()}")









if __name__ == "__main__":
    prediction()
```

Bonus

Running the test (ungraded)

The rest is ready, but we need to write a shell script for doing that.

batch.py | batch_q5.py | batch_q6.py | predict.sh | docker-compose.yaml .../homework

module-6 > homework > predict.sh

```bash
#!/usr/bin/env bash
export AWS_ACCESS_KEY_ID="dummy"
export AWS_SECRET_ACCESS_KEY="dummy"
export AWS_DEFAULT_REGION="us-east-1"
export INPUT_FILE_PATTERN="s3://nyc-duration/in/{year:04d}-{month:02d}.parquet"
export OUTPUT_FILE_PATTERN="s3://nyc-duration/out/{year:04d}-{month:02d}.parquet"
export S3_ENDPOINT_URL="http://localhost:4566"

docker-compose up -d
sleep 5
aws  s3 mb s3://nyc-duration --endpoint-url=${S3_ENDPOINT_URL}
sleep 5
pipenv run python integration_test/batch_q5.py 2023 01
pipenv run python batch_q6.py
sleep 5

ERROR_CODE=$?
if [ ${ERROR_CODE} != 0 ]; then
    docker-compose logs
    docker-compose down
    exit ${ERROR_CODE}
fi

echo "SUCCESS"
docker-compose down
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS 7

```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$ sh predict.sh
WARN[0000] Found orphan containers ([homework-adminer-1 homework-grafana-1 homework-db-1]) for this project
ose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 2/2
 ✓ Network homework_default  Created
 ✓ Container homework-S3-1   Started
make_bucket: nyc-duration
predicted mean duration: 18.138625226015364
Sum of prediction is  36.27725045203073
SUCCESS
[+] Running 2/2
 ✓ Container homework-S3-1   Removed
 ✓ Network homework_default  Removed
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-6/homework$
```