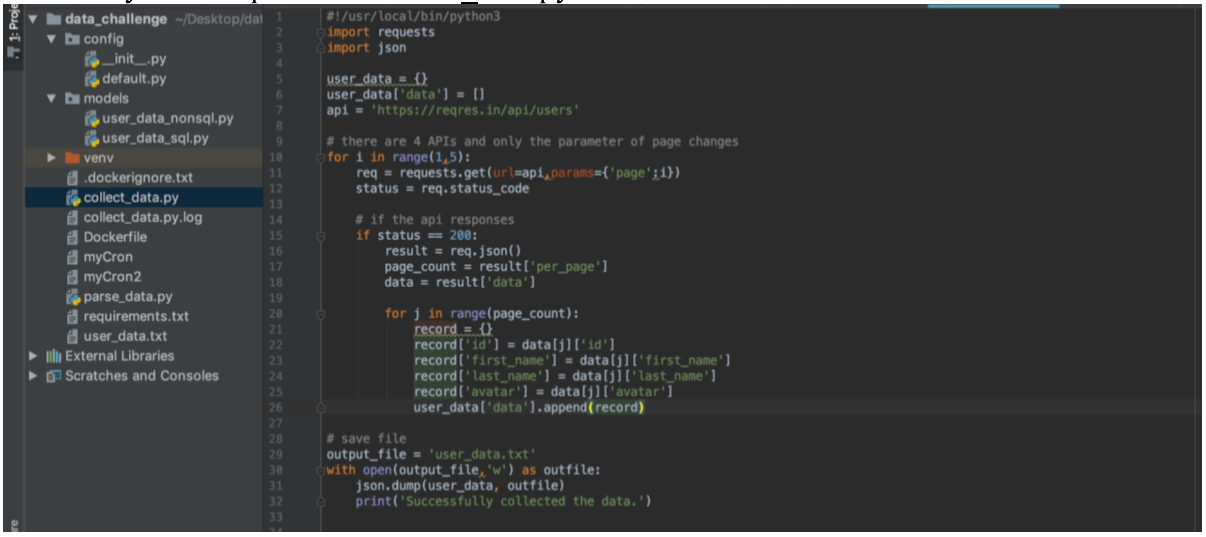
Data Challenge

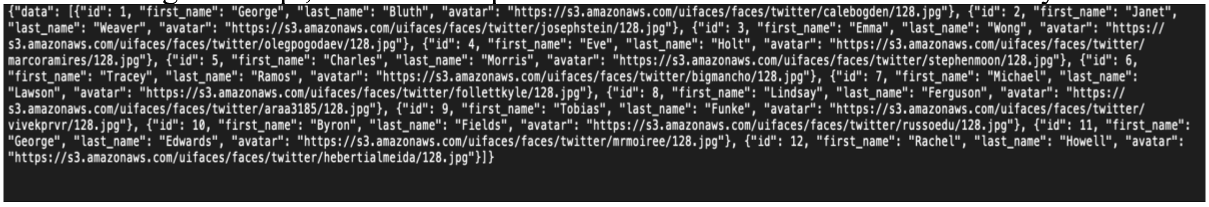
Xiaoyu(Alex) Zhu
04/10/2019

1. Create a Python program to collect the data
   a. Write a Python script named collect_data.py



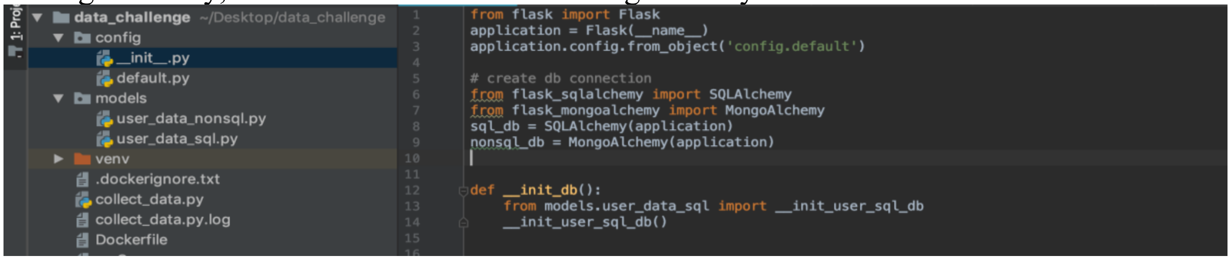
```python
#!/usr/local/bin/python3
import requests
import json

user_data = {}
user_data['data'] = []
api = 'https://reqres.in/api/users'

# there are 4 APIs and only the parameter of page changes
for i in range(1,5):
    req = requests.get(url=api,params={'page':i})
    status = req.status_code

    # if the api responses
    if status == 200:
        result = req.json()
        page_count = result['per_page']
        data = result['data']

        for j in range(page_count):
            record = {}
            record['id'] = data[j]['id']
            record['first_name'] = data[j]['first_name']
            record['last_name'] = data[j]['last_name']
            record['avatar'] = data[j]['avatar']
            user_data['data'].append(record)

# save file
output_file = 'user_data.txt'
with open(output_file,'w') as outfile:
    json.dump(user_data, outfile)
    print('Successfully collected the data.')
```

   b. After running the script, check the output file to see if data was collected correctly



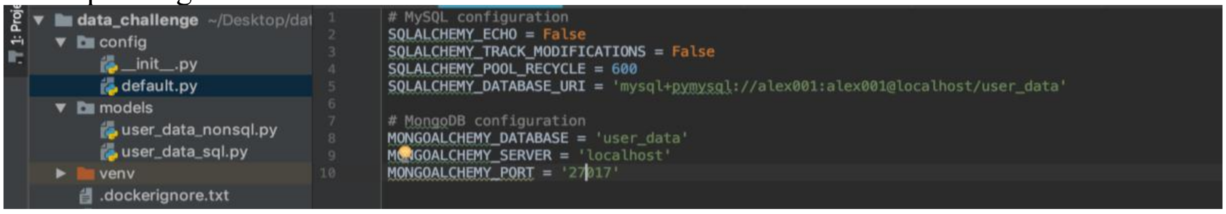
```
{"data": [{"id": 1, "first_name": "George", "last_name": "Bluth", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/calebogden/128.jpg"}, {"id": 2, "first_name": "Janet",
"last_name": "Weaver", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg"}, {"id": 3, "first_name": "Emma", "last_name": "Wong", "avatar": "https://
s3.amazonaws.com/uifaces/faces/twitter/olegpogodaev/128.jpg"}, {"id": 4, "first_name": "Eve", "last_name": "Holt", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/
marcoramires/128.jpg"}, {"id": 5, "first_name": "Charles", "last_name": "Morris", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/stephenmoon/128.jpg"}, {"id": 6,
"first_name": "Tracey", "last_name": "Ramos", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/bigmancho/128.jpg"}, {"id": 7, "first_name": "Michael", "last_name":
"Lawson", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/follettkyle/128.jpg"}, {"id": 8, "first_name": "Lindsay", "last_name": "Ferguson", "avatar": "https://
s3.amazonaws.com/uifaces/faces/twitter/araa3185/128.jpg"}, {"id": 9, "first_name": "Tobias", "last_name": "Funke", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/
vivekprvr/128.jpg"}, {"id": 10, "first_name": "Byron", "last_name": "Fields", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/russoedu/128.jpg"}, {"id": 11, "first_name":
"George", "last_name": "Edwards", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/mrmoiree/128.jpg"}, {"id": 12, "first_name": "Rachel", "last_name": "Howell", "avatar":
"https://s3.amazonaws.com/uifaces/faces/twitter/hebertialmeida/128.jpg"}]}
```

2. Create data model based on the data collected
   a. Use flask as the framework to initialize a web application
   b. Choose MySQL as RDBMS and MongoDB as non-relational DBMS
   c. Use ORM libraries built on Python, such as flask-sqlalchemy and flask-mongoalchemy, to interact with database management system



```python
from flask import Flask
application = Flask(__name__)
application.config.from_object('config.default')

# create db connection
from flask_sqlalchemy import SQLAlchemy
from flask_mongoalchemy import MongoAlchemy
sql_db = SQLAlchemy(application)
nonsql_db = MongoAlchemy(application)


def __init_db():
    from models.user_data_sql import __init_user_sql_db
    __init_user_sql_db()
```

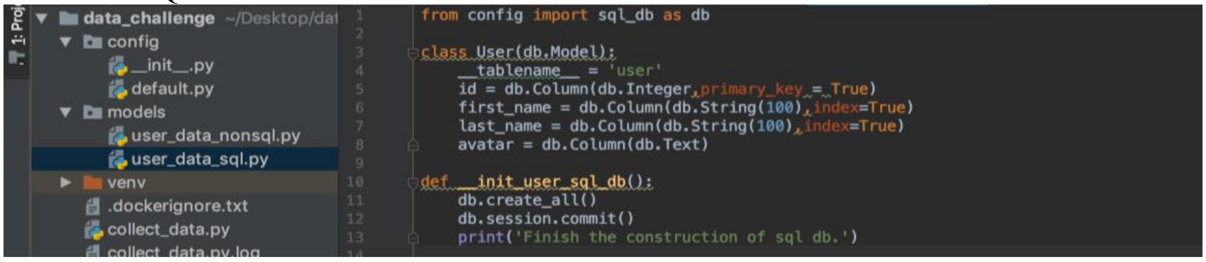
   d. Set up configuration of database connections



```python
# MySQL configuration
SQLALCHEMY_ECHO = False
SQLALCHEMY_TRACK_MODIFICATIONS = False
SQLALCHEMY_POOL_RECYCLE = 600
SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://alex001:alex001@localhost/user_data'

# MongoDB configuration
MONGOALCHEMY_DATABASE = 'user_data'
MONGOALCHEMY_SERVER = 'localhost'
MONGOALCHEMY_PORT = '27017'
```

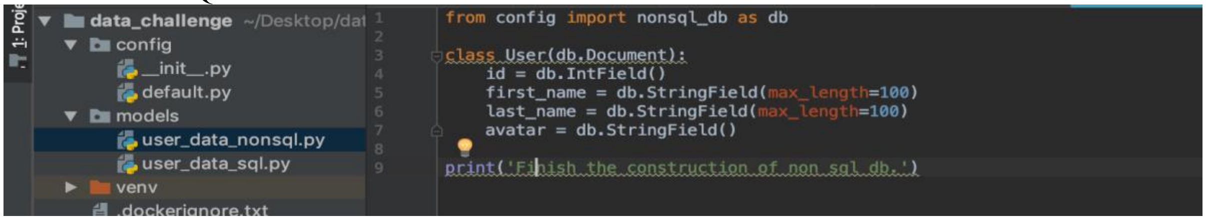
e. Build SQL and NoSQL models

i. SQL



```python
from config import sql_db as db

class User(db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer,primary_key = True)
    first_name = db.Column(db.String(100),index=True)
    last_name = db.Column(db.String(100),index=True)
    avatar = db.Column(db.Text)

def __init_user_sql_db():
    db.create_all()
    db.session.commit()
    print('Finish the construction of sql db.')
```

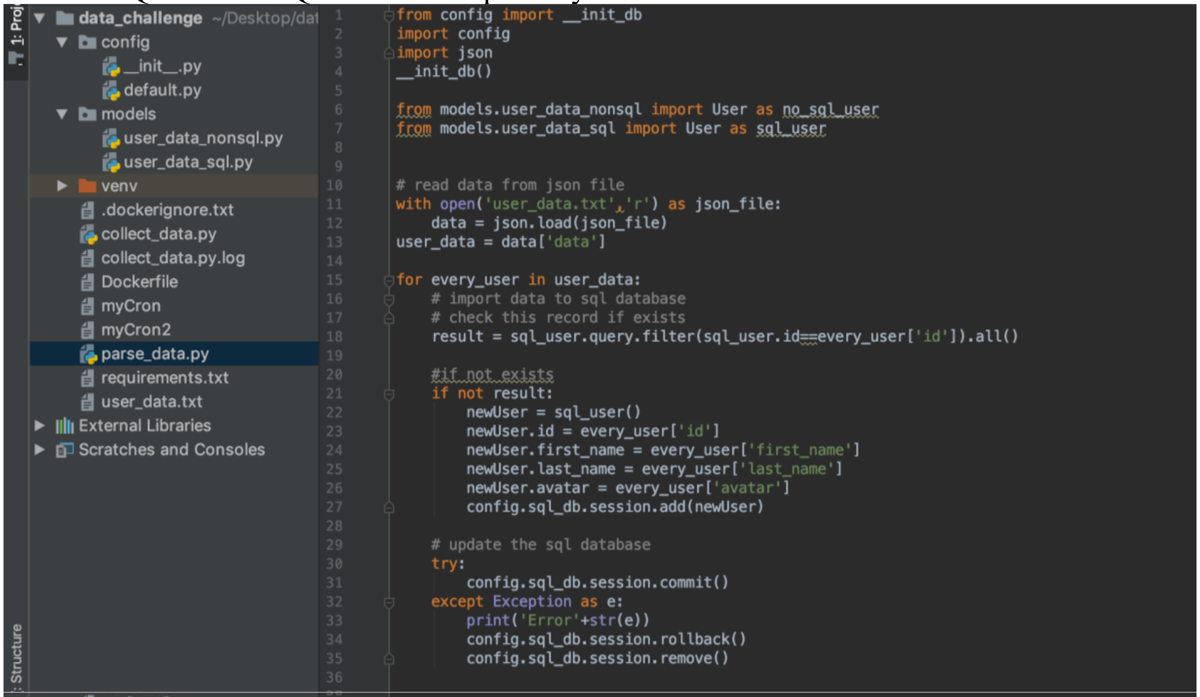
ii. NoSQL



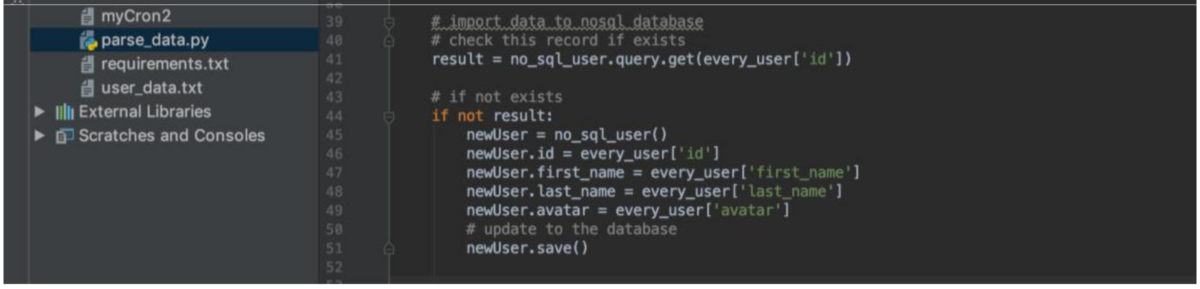
```python
from config import nonsql_db as db

class User(db.Document):
    id = db.IntField()
    first_name = db.StringField(max_length=100)
    last_name = db.StringField(max_length=100)
    avatar = db.StringField()

print('Finish the construction of non sql db.')
```

3. Parse the collected data in step 1 and using the data model created in step 2 create a program that performs the ETL with source being the collected data and target being SQL and No SQL source.

a. Write a Python script named parse_data.py and performs the ETL to import data into SQL and NoSQL database separately

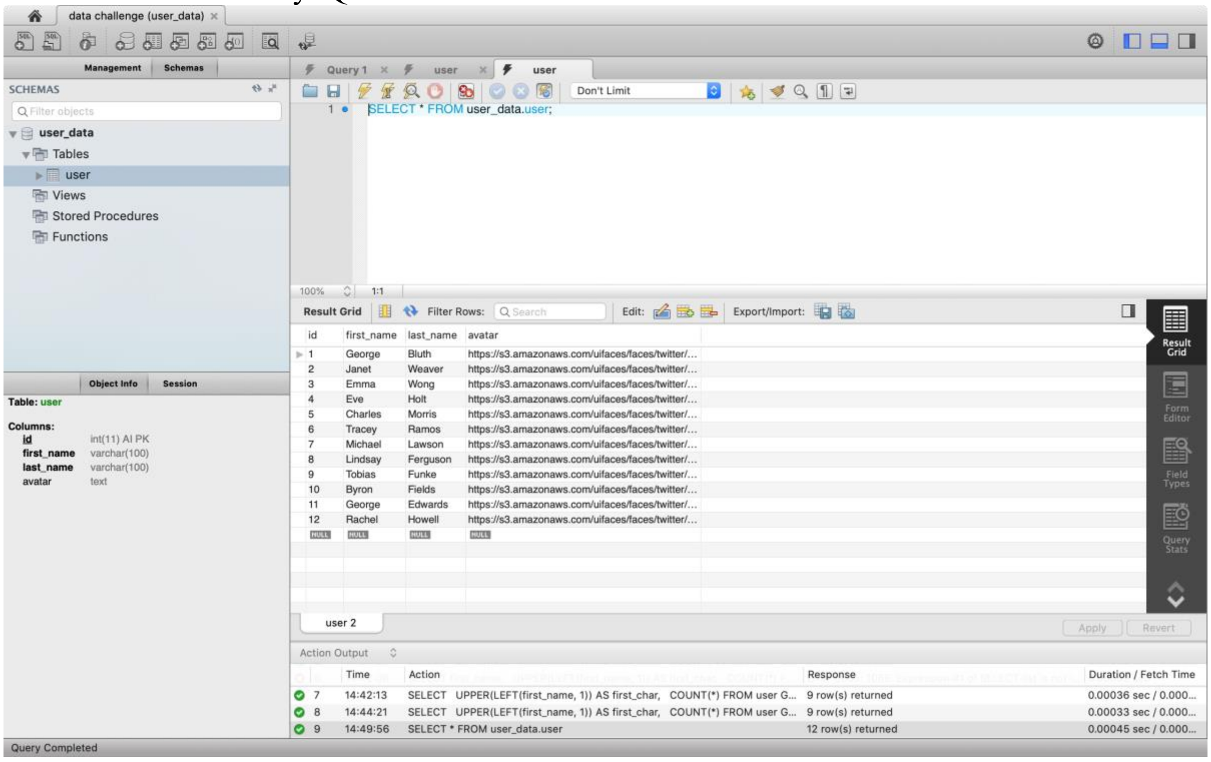


```python
from config import __init_db
import config
import json
__init_db()

from models.user_data_nonsql import User as no_sql_user
from models.user_data_sql import User as sql_user


# read data from json file
with open('user_data.txt','r') as json_file:
    data = json.load(json_file)
user_data = data['data']

for every_user in user_data:
    # import data to sql database
    # check this record if exists
    result = sql_user.query.filter(sql_user.id==every_user['id']).all()

    #if not exists
    if not result:
        newUser = sql_user()
        newUser.id = every_user['id']
        newUser.first_name = every_user['first_name']
        newUser.last_name = every_user['last_name']
        newUser.avatar = every_user['avatar']
        config.sql_db.session.add(newUser)

    # update the sql database
    try:
        config.sql_db.session.commit()
    except Exception as e:
        print('Error'+str(e))
        config.sql_db.session.rollback()
        config.sql_db.session.remove()

```



```python
    # import data to nosql database
    # check this record if exists
    result = no_sql_user.query.get(every_user['id'])

    # if not exists
    if not result:
        newUser = no_sql_user()
        newUser.id = every_user['id']
        newUser.first_name = every_user['first_name']
        newUser.last_name = every_user['last_name']
        newUser.avatar = every_user['avatar']
        # update to the database
        newUser.save()

```

b. Show results on MySQL



```
SELECT * FROM user_data.user;
```

Table: user

Columns:

id	int(11) AI PK
first_name	varchar(100)
last_name	varchar(100)
avatar	text

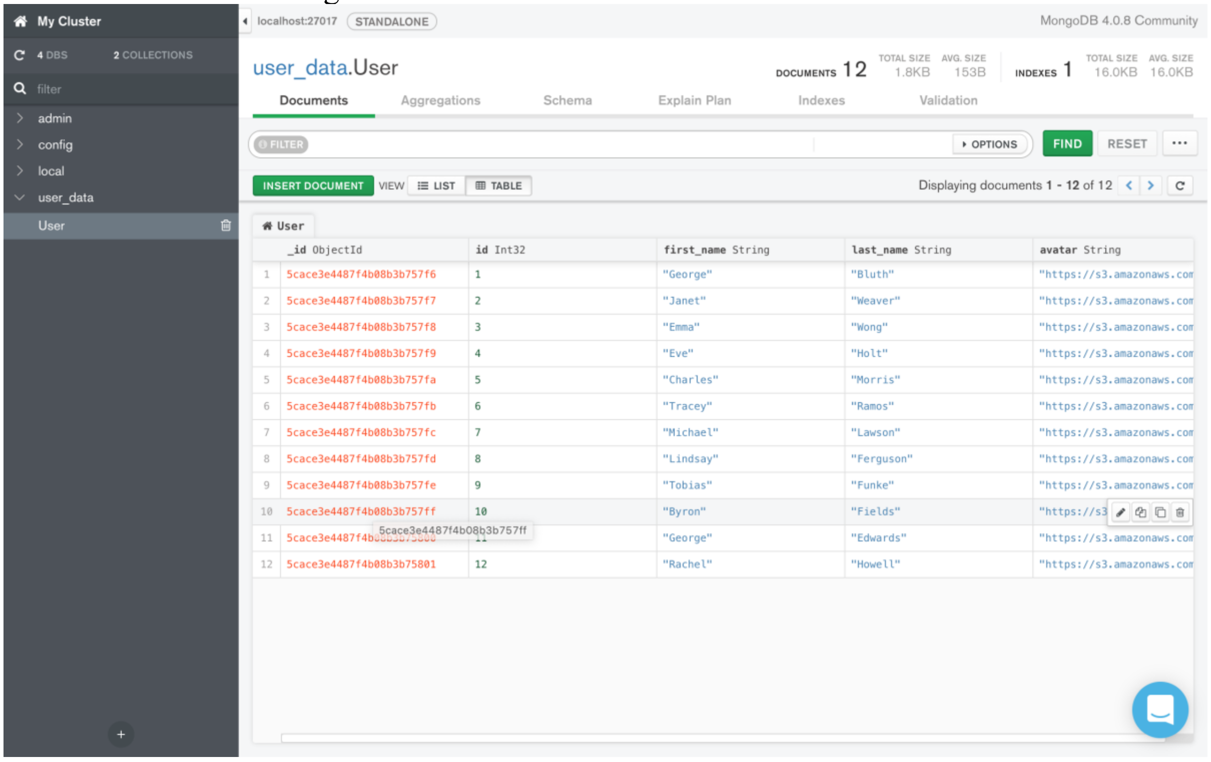
id	first_name	last_name	avatar
1	George	Bluth	https://s3.amazonaws.com/uifaces/faces/twitter/...
2	Janet	Weaver	https://s3.amazonaws.com/uifaces/faces/twitter/...
3	Emma	Wong	https://s3.amazonaws.com/uifaces/faces/twitter/...
4	Eve	Holt	https://s3.amazonaws.com/uifaces/faces/twitter/...
5	Charles	Morris	https://s3.amazonaws.com/uifaces/faces/twitter/...
6	Tracey	Ramos	https://s3.amazonaws.com/uifaces/faces/twitter/...
7	Michael	Lawson	https://s3.amazonaws.com/uifaces/faces/twitter/...
8	Lindsay	Ferguson	https://s3.amazonaws.com/uifaces/faces/twitter/...
9	Tobias	Funke	https://s3.amazonaws.com/uifaces/faces/twitter/...
10	Byron	Fields	https://s3.amazonaws.com/uifaces/faces/twitter/...
11	George	Edwards	https://s3.amazonaws.com/uifaces/faces/twitter/...
12	Rachel	Howell	https://s3.amazonaws.com/uifaces/faces/twitter/...
NULL	NULL	NULL	NULL

Action Output

	Time	Action	Response	Duration / Fetch Time
7	14:42:13	SELECT UPPER(LEFT(first_name, 1)) AS first_char, COUNT(*) FROM user G...	9 row(s) returned	0.00036 sec / 0.000...
8	14:44:21	SELECT UPPER(LEFT(first_name, 1)) AS first_char, COUNT(*) FROM user G...	9 row(s) returned	0.00033 sec / 0.000...
9	14:49:56	SELECT * FROM user_data.user	12 row(s) returned	0.00045 sec / 0.000...

Query Completed

c. Show results on MongoDB



user_data.User

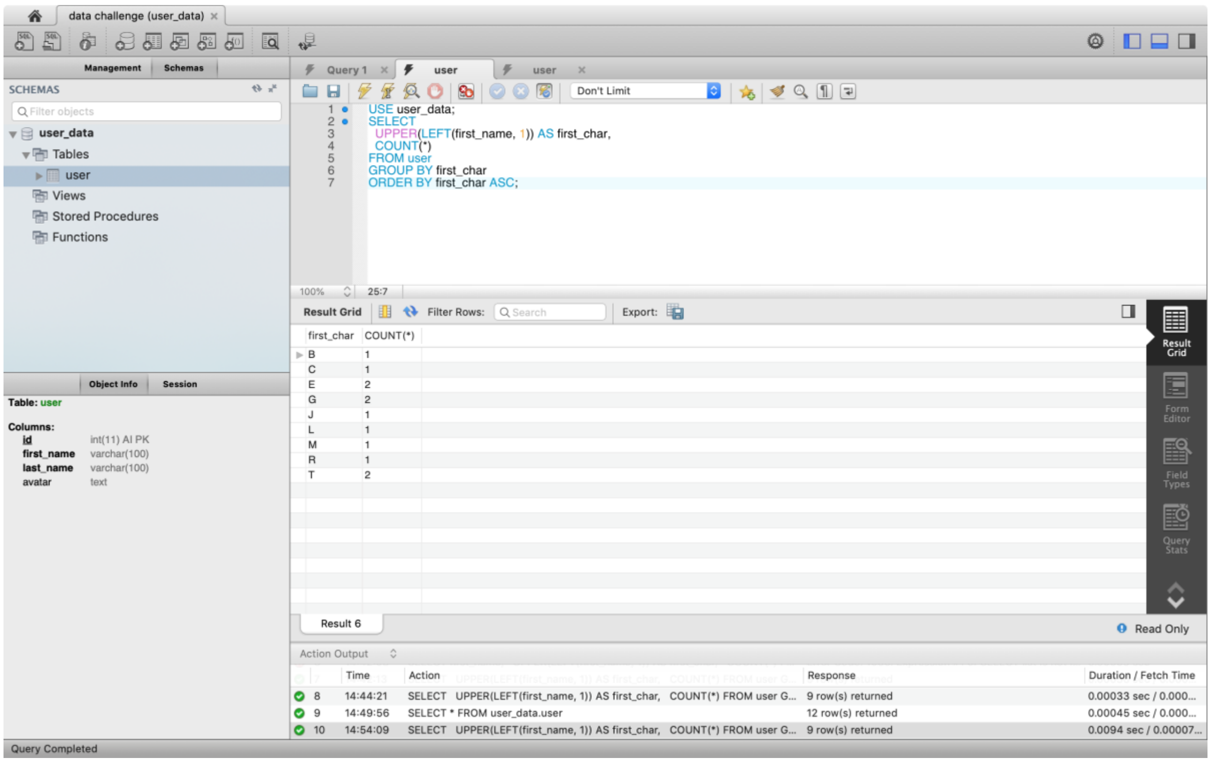
DOCUMENTS 12 TOTAL SIZE 1.8KB AVG. SIZE 153B INDEXES 1 TOTAL SIZE 16.0KB AVG. SIZE 16.0KB

Displaying documents 1 - 12 of 12

	_id ObjectId	id Int32	first_name String	last_name String	avatar String
1	5cace3e4487f4b08b3b757f6	1	"George"	"Bluth"	"https://s3.amazonaws.com
2	5cace3e4487f4b08b3b757f7	2	"Janet"	"Weaver"	"https://s3.amazonaws.com
3	5cace3e4487f4b08b3b757f8	3	"Emma"	"Wong"	"https://s3.amazonaws.com
4	5cace3e4487f4b08b3b757f9	4	"Eve"	"Holt"	"https://s3.amazonaws.com
5	5cace3e4487f4b08b3b757fa	5	"Charles"	"Morris"	"https://s3.amazonaws.com
6	5cace3e4487f4b08b3b757fb	6	"Tracey"	"Ramos"	"https://s3.amazonaws.com
7	5cace3e4487f4b08b3b757fc	7	"Michael"	"Lawson"	"https://s3.amazonaws.com
8	5cace3e4487f4b08b3b757fd	8	"Lindsay"	"Ferguson"	"https://s3.amazonaws.com
9	5cace3e4487f4b08b3b757fe	9	"Tobias"	"Funke"	"https://s3.amazonaws.com
10	5cace3e4487f4b08b3b757ff	10	"Byron"	"Fields"	"https://s3
11	5cace3e4487f4b08b3b75800	11	"George"	"Edwards"	"https://s3.amazonaws.com
12	5cace3e4487f4b08b3b75801	12	"Rachel"	"Howell"	"https://s3.amazonaws.com

5cace3e4487f4b08b3b757ff

4. Provide aggregation metrics for the collected data
   a. Use SQL language to calculate the count of first name starting with same letter and the count of last name starting with same letter
   b. Use LEFT() function to extract the first letter of words, group by the letter and count the number of records with same starting letter
   c. Show results



```sql
USE user_data;
SELECT
 UPPER(LEFT(first_name, 1)) AS first_char,
 COUNT(*)
FROM user
GROUP BY first_char
ORDER BY first_char ASC;
```

first_char	COUNT(*)
B	1
C	1
E	2
G	2
J	1
L	1
M	1
R	1
T	2

Table: user

Columns:

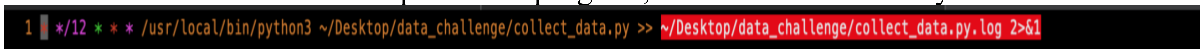
id	int(11) AI PK
first_name	varchar(100)
last_name	varchar(100)
avatar	text

Action Output

	Time	Action	Response	Duration / Fetch Time
8	14:44:21	SELECT UPPER(LEFT(first_name, 1)) AS first_char, COUNT(*) FROM user G...	9 row(s) returned	0.00033 sec / 0.000...
9	14:49:56	SELECT * FROM user_data.user	12 row(s) returned	0.00045 sec / 0.000...
10	14:54:09	SELECT UPPER(LEFT(first_name, 1)) AS first_char, COUNT(*) FROM user G...	9 row(s) returned	0.0094 sec / 0.00007...

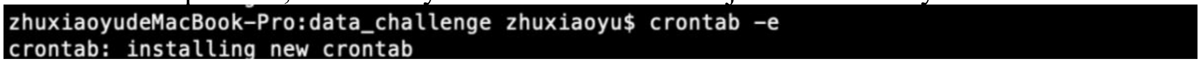
Query Completed

5. Create a schedule for above data extraction process step 1 that should be executed every 12 hours.
   a. Use crontab command to schedule a task
   b. *crontab -e* command will open a vim program, then write down the syntax



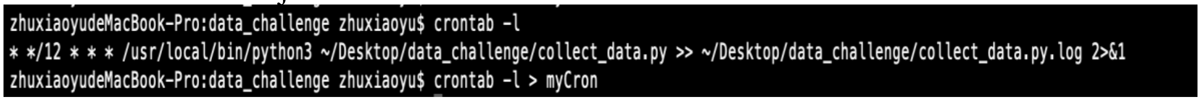
```
1 * */12 * * * /usr/local/bin/python3 ~/Desktop/data_challenge/collect_data.py >> ~/Desktop/data_challenge/collect_data.py.log 2>&1
```

   c. Save and quit vim, then the system will run the cron job immediately



```
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ crontab -e
crontab: installing new crontab
```

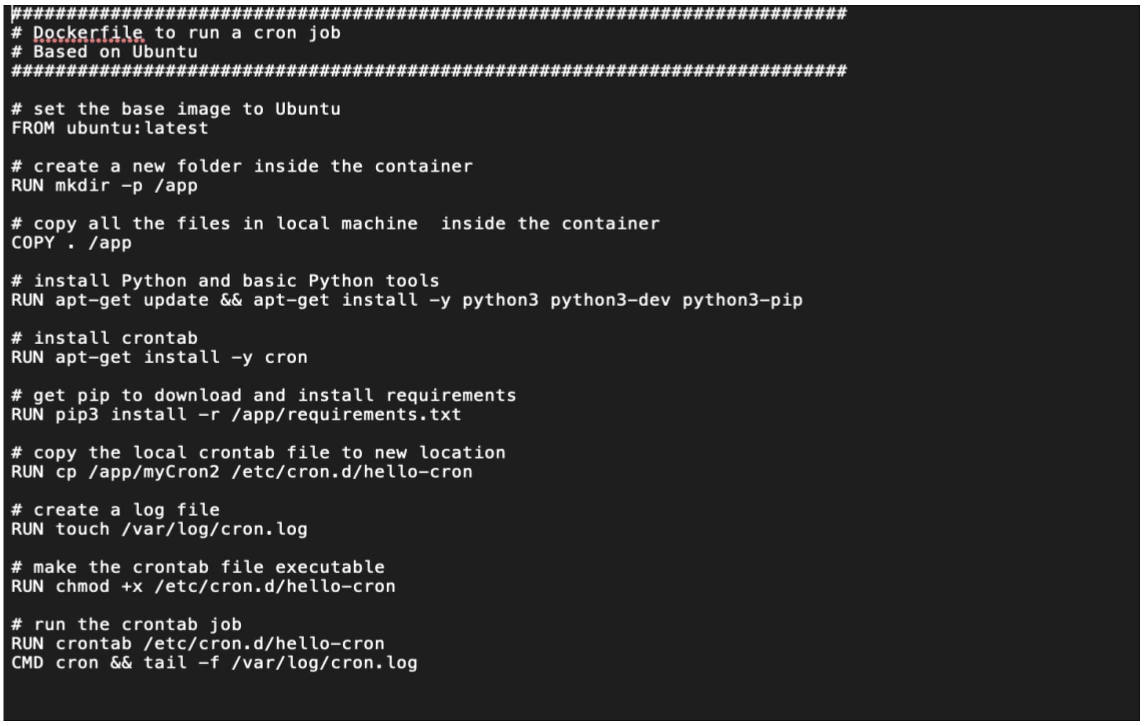
   d. Use *crontab -l* to check the cron job already created and use *crontab -l > filename* to save current cron job



```
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ crontab -l
* */12 * * * /usr/local/bin/python3 ~/Desktop/data_challenge/collect_data.py >> ~/Desktop/data_challenge/collect_data.py.log 2>&1
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ crontab -l > myCron
```

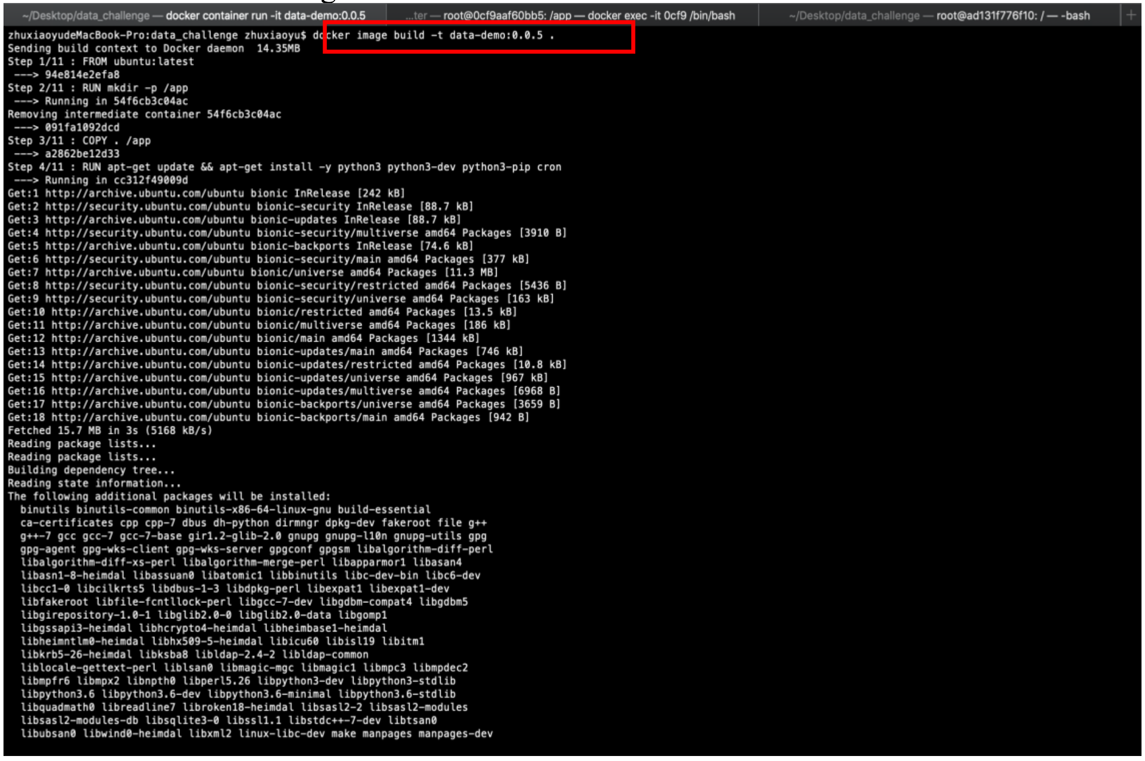
6. Extra points: use docker to containerize the project

a. Create a dockerfile



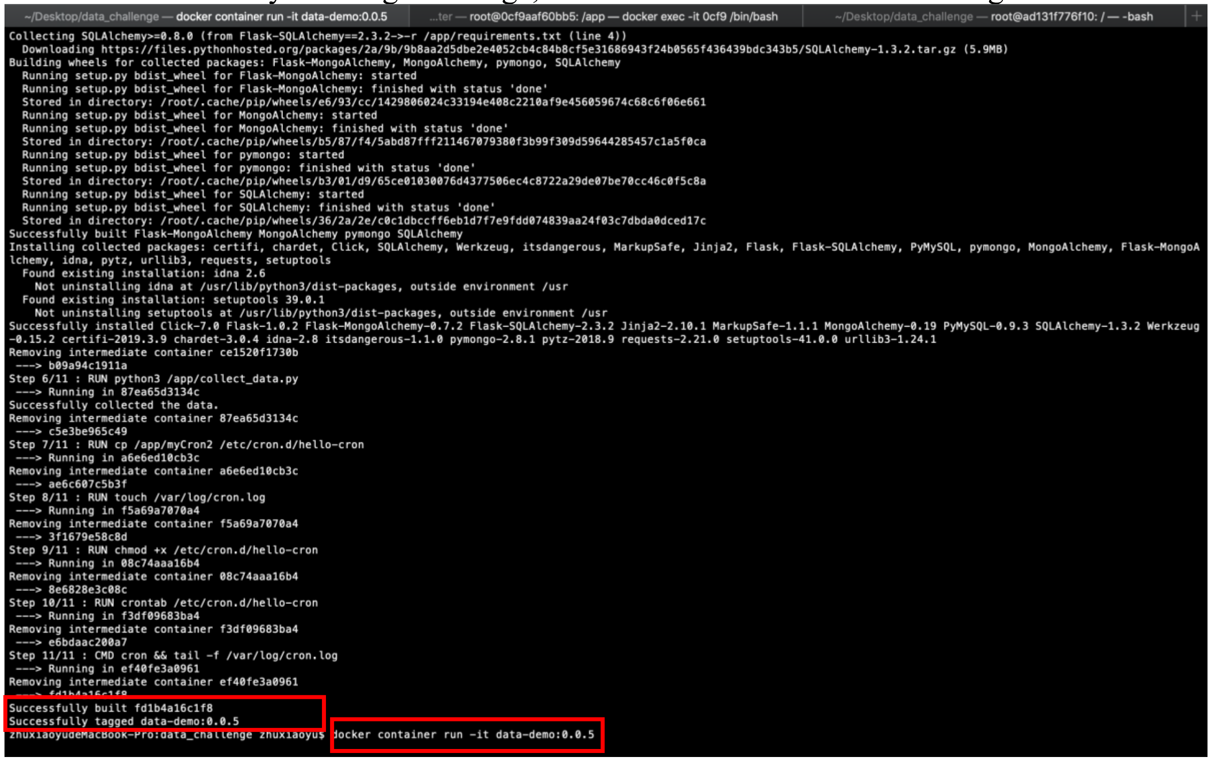
```
###########################################################################
# Dockerfile to run a cron job
# Based on Ubuntu
###########################################################################

# set the base image to Ubuntu
FROM ubuntu:latest

# create a new folder inside the container
RUN mkdir -p /app

# copy all the files in local machine  inside the container
COPY . /app

# install Python and basic Python tools
RUN apt-get update && apt-get install -y python3 python3-dev python3-pip

# install crontab
RUN apt-get install -y cron

# get pip to download and install requirements
RUN pip3 install -r /app/requirements.txt

# copy the local crontab file to new location
RUN cp /app/myCron2 /etc/cron.d/hello-cron

# create a log file
RUN touch /var/log/cron.log

# make the crontab file executable
RUN chmod +x /etc/cron.d/hello-cron

# run the crontab job
RUN crontab /etc/cron.d/hello-cron
CMD cron && tail -f /var/log/cron.log
```

b. Build the docker image


~/Desktop/data_challenge — docker container run -it data-demo:0.0.5
...ter — root@0cf9aaf60bb5: /app — docker exec -it 0cf9 /bin/bash
~/Desktop/data_challenge — root@ad131f776f10: / — -bash

```
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ docker image build -t data-demo:0.0.5 .
Sending build context to Docker daemon  14.35MB
Step 1/11 : FROM ubuntu:latest
 ---> 94e814e2efa8
Step 2/11 : RUN mkdir -p /app
 ---> Running in 54f6cb3c04ac
Removing intermediate container 54f6cb3c04ac
 ---> 091fa1092dcd
Step 3/11 : COPY . /app
 ---> a2862be12d33
Step 4/11 : RUN apt-get update && apt-get install -y python3 python3-dev python3-pip cron
 ---> Running in cc312f49009d
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [3910 B]
Get:5 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [377 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [5436 B]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [163 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [746 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [10.8 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [967 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [6968 B]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [3659 B]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [942 B]
Fetched 15.7 MB in 3s (5168 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential
  ca-certificates cpp cpp-7 dbus dh-python dirmngr dpkg-dev fakeroot file g++
  g++-7 gcc gcc-7 gcc-7-base gir1.2-glib-2.0 gnupg gnupg-l10n gnupg-utils gpg
  gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libapparmor1 libasan4
  libasn1-8-heimdal libassuan0 libatomic1 libbinutils libc-dev-bin libc6-dev
  libcc1-0 libcilkrts5 libdbus-1-3 libdpkg-perl libexpat1 libexpat1-dev
  libfakeroot libfile-fcntllock-perl libgcc-7-dev libgdbm-compat4 libgdbm5
  libgirepository-1.0-1 libglib2.0-0 libglib2.0-data libgomp1
  libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal
  libheimntlm0-heimdal libhx509-5-heimdal libicu60 libisl19 libitm1
  libkrb5-26-heimdal libksba8 libldap-2.4-2 libldap-common
  liblocale-gettext-perl liblsan0 libmagic-mgc libmagic1 libmpc3 libmpdec2
  libmpfr6 libmpx2 libnpth0 libperl5.26 libpython3-dev libpython3-stdlib
  libpython3.6 libpython3.6-dev libpython3.6-minimal libpython3.6-stdlib
  libquadmath0 libreadline7 libroken18-heimdal libsasl2-2 libsasl2-modules
  libsasl2-modules-db libsqlite3-0 libssl1.1 libstdc++-7-dev libtsan0
  libubsan0 libwind0-heimdal libxml2 linux-libc-dev make manpages manpages-dev
```

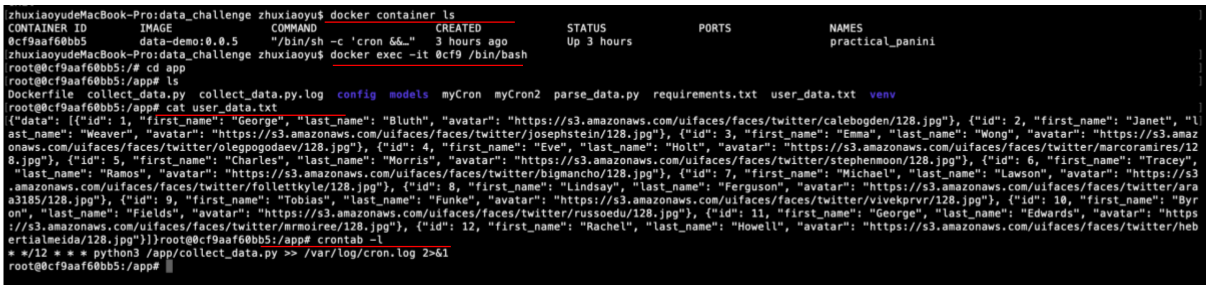
c. After successfully building the image, run the container based on the image



~/Desktop/data_challenge — docker container run -it data-demo:0.0.5
...ter — root@0cf9aaf60bb5: /app — docker exec -it 0cf9 /bin/bash
~/Desktop/data_challenge — root@ad131f776f10: / — -bash

```
Collecting SQLAlchemy>=0.8.0 (from Flask-SQLAlchemy==2.3.2->-r /app/requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/2a/9b/9b8aa2d5dbe2e4052cb4c84b8cf5e31686943f24b0565f436439bdc343b5/SQLAlchemy-1.3.2.tar.gz (5.9MB)
Building wheels for collected packages: Flask-MongoAlchemy, MongoAlchemy, pymongo, SQLAlchemy
  Running setup.py bdist_wheel for Flask-MongoAlchemy: started
  Running setup.py bdist_wheel for Flask-MongoAlchemy: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/e6/93/cc/1429806024c33194e408c2210af9e456059674c68c6f06e661
  Running setup.py bdist_wheel for MongoAlchemy: started
  Running setup.py bdist_wheel for MongoAlchemy: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/b5/87/f4/5abd87fff211467079380f3b99f309d59644285457c1a5f0ca
  Running setup.py bdist_wheel for pymongo: started
  Running setup.py bdist_wheel for pymongo: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/b3/01/d9/65ce01030076d4377506ec4c8722a29de07be70cc46c0f5c8a
  Running setup.py bdist_wheel for SQLAlchemy: started
  Running setup.py bdist_wheel for SQLAlchemy: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/36/2a/2e/c0c1dbccff6eb1d7f7e9fdd074839aa24f03c7dbda0dced17c
Successfully built Flask-MongoAlchemy MongoAlchemy pymongo SQLAlchemy
Installing collected packages: certifi, chardet, Click, SQLAlchemy, Werkzeug, itsdangerous, MarkupSafe, Jinja2, Flask, Flask-SQLAlchemy, PyMySQL, pymongo, MongoAlchemy, Flask-MongoA
lchemy, idna, pytz, urllib3, requests, setuptools
  Found existing installation: idna 2.6
    Not uninstalling idna at /usr/lib/python3/dist-packages, outside environment /usr
  Found existing installation: setuptools 39.0.1
    Not uninstalling setuptools at /usr/lib/python3/dist-packages, outside environment /usr
Successfully installed Click-7.0 Flask-1.0.2 Flask-MongoAlchemy-0.7.2 Flask-SQLAlchemy-2.3.2 Jinja2-2.10.1 MarkupSafe-1.1.1 MongoAlchemy-0.19 PyMySQL-0.9.3 SQLAlchemy-1.3.2 Werkzeug
-0.15.2 certifi-2019.3.9 chardet-3.0.4 idna-2.8 itsdangerous-1.1.0 pymongo-2.8.1 pytz-2018.9 requests-2.21.0 setuptools-41.0.0 urllib3-1.24.1
Removing intermediate container ce1520f1730b
 ---> b09a94c1911a
Step 6/11 : RUN python3 /app/collect_data.py
 ---> Running in 87ea65d3134c
Successfully collected the data.
Removing intermediate container 87ea65d3134c
 ---> c5e3be965c49
Step 7/11 : RUN cp /app/myCron2 /etc/cron.d/hello-cron
 ---> Running in a6e6ed10cb3c
Removing intermediate container a6e6ed10cb3c
 ---> ae6c607c5b3f
Step 8/11 : RUN touch /var/log/cron.log
 ---> Running in f5a69a7070a4
Removing intermediate container f5a69a7070a4
 ---> 3f1679e58c8d
Step 9/11 : RUN chmod +x /etc/cron.d/hello-cron
 ---> Running in 08c74aaa16b4
Removing intermediate container 08c74aaa16b4
 ---> 8e6828e3c08c
Step 10/11 : RUN crontab /etc/cron.d/hello-cron
 ---> Running in f3df09683ba4
Removing intermediate container f3df09683ba4
 ---> e6bdaac200a7
Step 11/11 : CMD cron && tail -f /var/log/cron.log
 ---> Running in ef40fe3a0961
Removing intermediate container ef40fe3a0961
 ---> fd1b4a16c1f8
Successfully built fd1b4a16c1f8
Successfully tagged data-demo:0.0.5
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ docker container run -it data-demo:0.0.5
```

d. During running the container, open a new window of terminal, find the container id and use *docker exec* command to walk into the container



```
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
0cf9aaf60bb5        data-demo:0.0.5     "/bin/sh -c 'cron &&…"   3 hours ago         Up 3 hours                              practical_panini
zhuxiaoyudeMacBook-Pro:data_challenge zhuxiaoyu$ docker exec -it 0cf9 /bin/bash
root@0cf9aaf60bb5:/# cd app
root@0cf9aaf60bb5:/app# ls
Dockerfile  collect_data.py  collect_data.py.log  config  models  myCron  myCron2  parse_data.py  requirements.txt  user_data.txt  venv
root@0cf9aaf60bb5:/app# cat user_data.txt
{"data": [{"id": 1, "first_name": "George", "last_name": "Bluth", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/calebogden/128.jpg"}, {"id": 2, "first_name": "Janet", "l
ast_name": "Weaver", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg"}, {"id": 3, "first_name": "Emma", "last_name": "Wong", "avatar": "https://s3.amaz
onaws.com/uifaces/faces/twitter/olegpogodaev/128.jpg"}, {"id": 4, "first_name": "Eve", "last_name": "Holt", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/marcoramires/12
8.jpg"}, {"id": 5, "first_name": "Charles", "last_name": "Morris", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/stephenmoon/128.jpg"}, {"id": 6, "first_name": "Tracey",
 "last_name": "Ramos", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/bigmancho/128.jpg"}, {"id": 7, "first_name": "Michael", "last_name": "Lawson", "avatar": "https://s3
.amazonaws.com/uifaces/faces/twitter/follettkyle/128.jpg"}, {"id": 8, "first_name": "Lindsay", "last_name": "Ferguson", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/ara
a3185/128.jpg"}, {"id": 9, "first_name": "Tobias", "last_name": "Funke", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/vivekprvr/128.jpg"}, {"id": 10, "first_name": "Byr
on", "last_name": "Fields", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/russoedu/128.jpg"}, {"id": 11, "first_name": "George", "last_name": "Edwards", "avatar": "https
://s3.amazonaws.com/uifaces/faces/twitter/mrmoiree/128.jpg"}, {"id": 12, "first_name": "Rachel", "last_name": "Howell", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/heb
ertialmeida/128.jpg"}]}root@0cf9aaf60bb5:/app# crontab -l
* */12 * * * python3 /app/collect_data.py >> /var/log/cron.log 2>&1
root@0cf9aaf60bb5:/app#
```