

PART 1: FETCH RECORD AND CONVERT INTO THE DICTIONARY

Project Code (GitHub): https://github.com/ajay9889/de_assignment/

https://github.com/ajay9889/de_assignment/blob/main/src/part_1_api.py

The image displays two screenshots of a Visual Studio Code editor window, showing the implementation of a Python script to fetch records from an API and convert them into a dictionary.

Top Screenshot: The editor shows the file `part_1_api.py` in the `src` directory. The code includes imports for `json`, `requests`, and `os`. It defines variables for `base_url`, `resource_id`, `offset`, `limit`, and `source_record`. A `while` loop is used to fetch records by changing the `offset` value. The code includes a `requests.get` call to fetch the data and a `status_code` check to handle failed responses.

```
17 import json
18 import requests
19 import os
20 # initialize defined variables
21
22 base_url = 'https://eservices.mas.gov.sg/api/action/datastore/search.json'
23 resource_id = '3a5b732e-9490-4629-a398-d0c414204ee0'
24 offset = 0
25 limit = 100
26 source_record = []
27
28 # Iterate and fetch all records by changing the offset value as a parameter
29 # 2. Get a response from all requests i.e. 100 records at most.
30 while True:
31     # request parameter
32     params = {
33         'resource_id': resource_id,
34         'limit': limit,
35         'offset': offset
36     }
37
38     # 1. Request Web API
39     response = requests.get(base_url, params=params)
40
41     # 4. Handle the failed response
42     if (response.status_code == 200):
43         data = response.json()
44
```

Bottom Screenshot: The editor shows the continuation of the `part_1_api.py` file. It includes a `status_code` check to handle failed responses. If the status code is 200, it extracts the records from the response and appends them to the `source_record` list. It also updates the `offset` value using the `limit`. If the status code is not 200, it prints an error message and breaks the loop. Finally, it converts the `source_record` list into a dictionary and prints it.

```
44 if (response.status_code == 200):
45     data = response.json()
46     records = data['result']['records']
47
48     # If records are null or empty
49     if not records:
50         break
51
52     source_record.extend(records)
53
54     # now the update the offset value using limit
55     offset += limit
56
57 else:
58     response = {
59         "status_code": response.status_code,
60         "status": "Error",
61         "message": "Failed to fetch the response from end points"
62     }
63     print(response)
64     break
65
66
67 # finally
68 # 3. Convert it into the Dictionary
69 for record in source_record[:5]:
70     print(record)
71
```

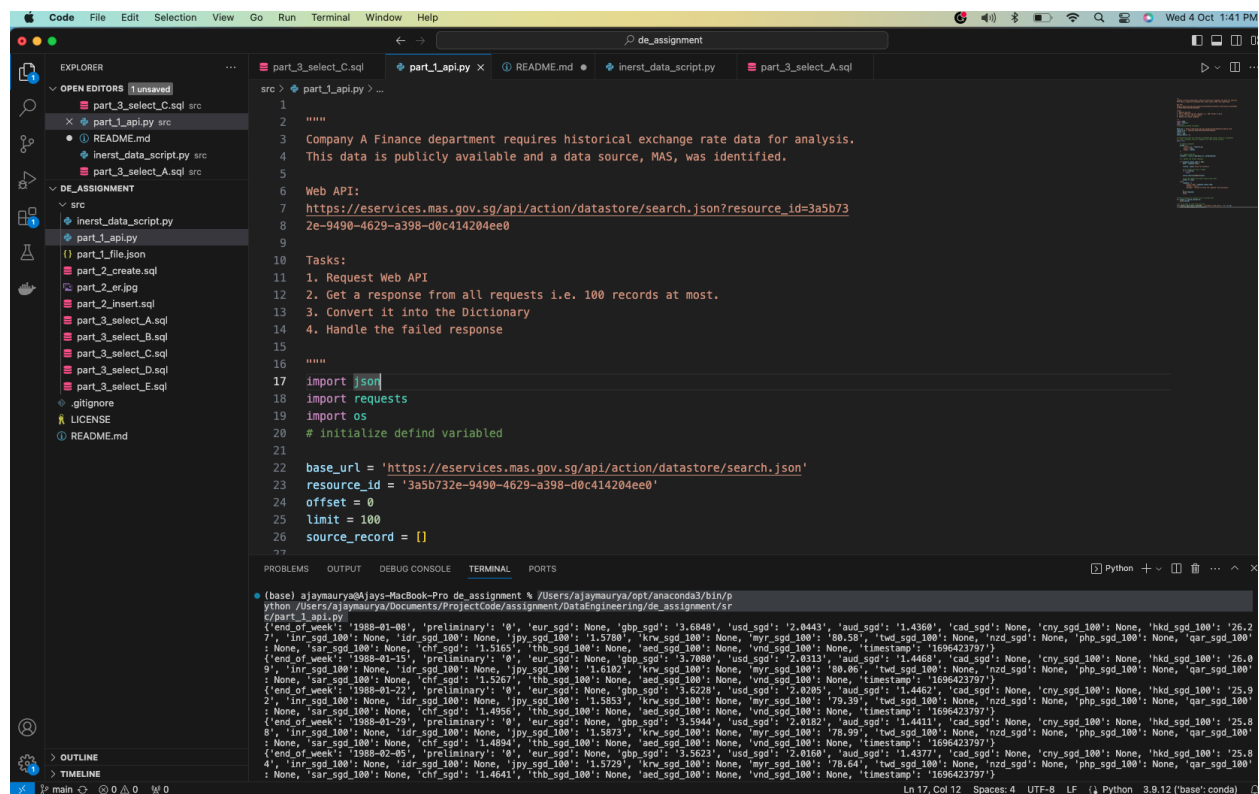
Response Handl:

```
## finally display on console first 5 records only
for record in source_record[:5]:
    print(record)

# or if we want to save into the file
with open(os.path.join(os.getcwd(), 'src/part_1_file.json'), 'w+') as
    fp:
        fp.write(json.dumps(source_record))
```

Display 5 records on the console and dump data into the the file in same src/**part_1_file** folder

https://github.com/ajay9889/de_assignment/blob/main/src/part_1_file.json



The screenshot shows a VS Code editor with a project named 'de_assignment'. The Explorer sidebar on the left shows the file structure, including a 'src' folder with files like 'part_1_file.json', 'part_2_create.sql', 'part_2_er.jpg', 'part_2_insert.sql', 'part_3_select_A.sql', 'part_3_select_B.sql', 'part_3_select_C.sql', 'part_3_select_D.sql', and 'part_3_select_E.sql'. The main editor window displays the 'part_1_api.py' file. The script is a REST API endpoint that fetches exchange rate data from the MAS.gov.sg API. It includes comments, imports for 'json', 'requests', and 'os', and defines variables for 'base_url', 'resource_id', 'offset', 'limit', and 'source_record'. The terminal at the bottom shows the output of the script, displaying a list of 5 records of exchange rate data for various currencies (USD, AUD, CAD, CNY, HKD, NZD, PHP, QAR, SGD) for the week of 1988-01-08 to 1988-01-15. The status bar at the bottom indicates the file is at line 17, column 12, with 4 spaces, in UTF-8 encoding, using the Python 3.9.12 interpreter.

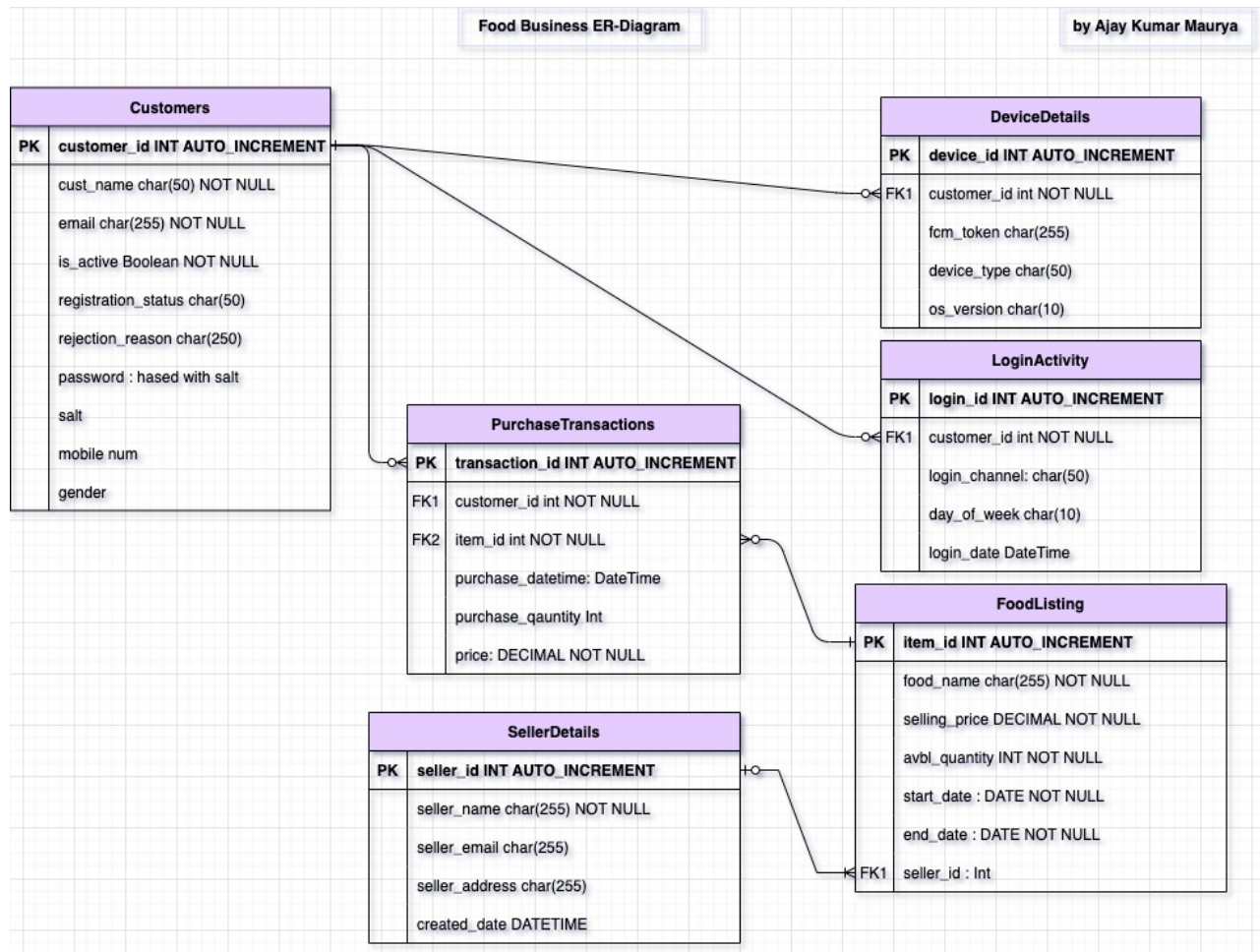
```
1
2
3 Company A Finance department requires historical exchange rate data for analysis.
4 This data is publicly available and a data source, MAS, was identified.
5
6 Web API:
7 https://eservices.mas.gov.sg/api/action/datastore/search.json?resource_id=3a5b73
8 2e-9490-4629-a398-d0c414204ee0
9
10 Tasks:
11 1. Request Web API
12 2. Get a response from all requests i.e. 100 records at most.
13 3. Convert it into the Dictionary
14 4. Handle the failed response
15
16
17 import json
18 import requests
19 import os
20 # initialize defined variables
21
22 base_url = 'https://eservices.mas.gov.sg/api/action/datastore/search.json'
23 resource_id = '3a5b732e-9490-4629-a398-d0c414204ee0'
24 offset = 0
25 limit = 100
26 source_record = []
27
```

```
(base) ajaymaurya@ajays-MacBook-Pro de_assignment % cd /Users/ajaymaurya/opt/anaconda3/bin/p
ython /Users/ajaymaurya/Documents/ProjectCode/assignment/DataEngineering/de_assignment/sr
c/part_1_api.py
{"end_of_week": "1988-01-08", "preliminary": "0", "eur_sgd": None, "gbp_sgd": "3.6848", "usd_sgd": "2.0443", "aud_sgd": "1.4368", "cad_sgd": None, "cny_sgd_100": None, "hkd_sgd_100": "26.2
7", "inr_sgd_100": None, "idr_sgd_100": None, "jpy_sgd_100": "1.5780", "krw_sgd_100": None, "myr_sgd_100": "80.58", "twd_sgd_100": None, "nzd_sgd": None, "php_sgd_100": None, "qar_sgd_100"
: None, "sar_sgd_100": None, "chf_sgd": "1.5165", "thb_sgd_100": None, "aed_sgd_100": None, "vnd_sgd_100": None, "timestamp": "1696423797"}
{"end_of_week": "1988-01-15", "preliminary": "0", "eur_sgd": None, "gbp_sgd": "3.7888", "usd_sgd": "2.0313", "aud_sgd": "1.4468", "cad_sgd": None, "cny_sgd_100": None, "hkd_sgd_100": "26.0
9", "inr_sgd_100": None, "idr_sgd_100": None, "jpy_sgd_100": "1.6102", "krw_sgd_100": None, "myr_sgd_100": "80.06", "twd_sgd_100": None, "nzd_sgd": None, "php_sgd_100": None, "qar_sgd_100"
: None, "sar_sgd_100": None, "chf_sgd": "1.5267", "thb_sgd_100": None, "aed_sgd_100": None, "vnd_sgd_100": None, "timestamp": "1696423797"}
{"end_of_week": "1988-01-22", "preliminary": "0", "eur_sgd": None, "gbp_sgd": "3.6228", "usd_sgd": "2.0205", "aud_sgd": "1.4462", "cad_sgd": None, "cny_sgd_100": None, "hkd_sgd_100": "25.9
2", "inr_sgd_100": None, "idr_sgd_100": None, "jpy_sgd_100": "1.5853", "krw_sgd_100": None, "myr_sgd_100": "79.39", "twd_sgd_100": None, "nzd_sgd": None, "php_sgd_100": None, "qar_sgd_100"
: None, "sar_sgd_100": None, "chf_sgd": "1.4956", "thb_sgd_100": None, "aed_sgd_100": None, "vnd_sgd_100": None, "timestamp": "1696423797"}
{"end_of_week": "1988-01-29", "preliminary": "0", "eur_sgd": None, "gbp_sgd": "3.5944", "usd_sgd": "2.0182", "aud_sgd": "1.4411", "cad_sgd": None, "cny_sgd_100": None, "hkd_sgd_100": "25.8
8", "inr_sgd_100": None, "idr_sgd_100": None, "jpy_sgd_100": "1.5873", "krw_sgd_100": None, "myr_sgd_100": "78.99", "twd_sgd_100": None, "nzd_sgd": None, "php_sgd_100": None, "qar_sgd_100"
: None, "sar_sgd_100": None, "chf_sgd": "1.4894", "thb_sgd_100": None, "aed_sgd_100": None, "vnd_sgd_100": None, "timestamp": "1696423797"}
{"end_of_week": "1988-02-05", "preliminary": "0", "eur_sgd": None, "gbp_sgd": "3.5623", "usd_sgd": "2.0168", "aud_sgd": "1.4377", "cad_sgd": None, "cny_sgd_100": None, "hkd_sgd_100": "25.8
4", "inr_sgd_100": None, "idr_sgd_100": None, "jpy_sgd_100": "1.5729", "krw_sgd_100": None, "myr_sgd_100": "78.64", "twd_sgd_100": None, "nzd_sgd": None, "php_sgd_100": None, "qar_sgd_100"
: None, "sar_sgd_100": None, "chf_sgd": "1.4641", "thb_sgd_100": None, "aed_sgd_100": None, "vnd_sgd_100": None, "timestamp": "1696423797"}
Ln 17, Col 12 Spaces: 4 UTF-8 LF (Python 3.9.12 ("base:conda"))
```

PART 2: Data Modeling, Creating Table and Insert Required Data into the respective Tables.

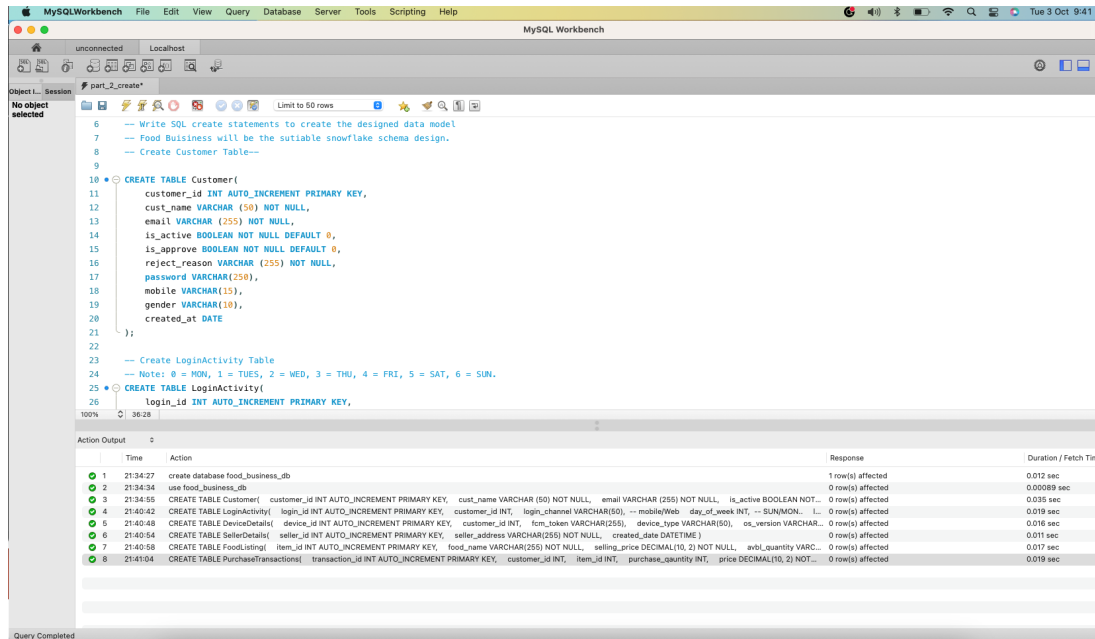
Created ER-Diagram:

https://github.com/ajay9889/de_assignment/blob/main/src/part_2_er.jpg



Accordingly, run the create Tables command to create all required tables based on the above ER diagram in MySQL WorkBench.

https://github.com/ajay9889/de_assignment/blob/main/src/part_2_create.sql

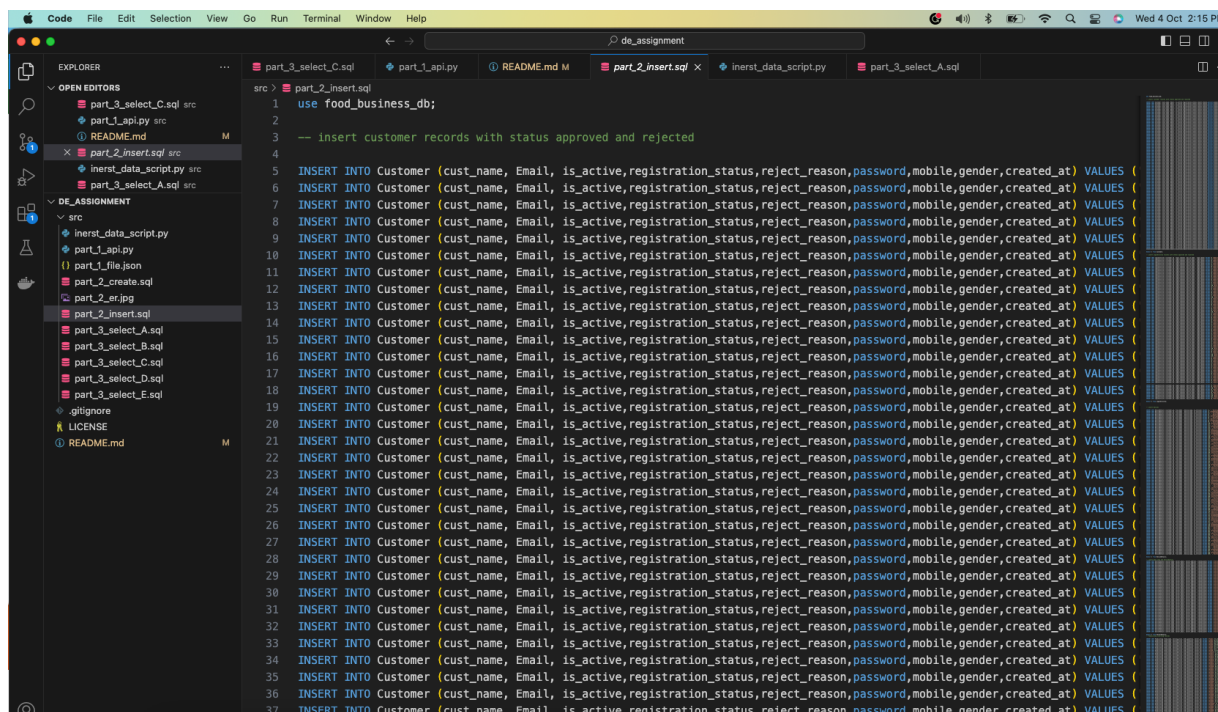


```
6 -- Write SQL create statements to create the designed data model
7 -- Food Business will be the suitable snowflake schema design.
8 -- Create Customer Table--
9
10 CREATE TABLE Customer(
11     customer_id INT AUTO_INCREMENT PRIMARY KEY,
12     cust_name VARCHAR(50) NOT NULL,
13     email VARCHAR(255) NOT NULL,
14     is_active BOOLEAN NOT NULL DEFAULT 0,
15     is_approve BOOLEAN NOT NULL DEFAULT 0,
16     reject_reason VARCHAR(255) NOT NULL,
17     password VARCHAR(250),
18     mobile VARCHAR(15),
19     gender VARCHAR(10),
20     created_at DATE
21 );
22
23 -- Create LoginActivity Table
24 -- Note: 0 = MON, 1 = TUES, 2 = WED, 3 = THU, 4 = FRI, 5 = SAT, 6 = SUN.
25 CREATE TABLE LoginActivity(
26     login_id INT AUTO_INCREMENT PRIMARY KEY,
```

Time	Action	Response	Duration / Fetch Time
21:34:27	create database food_business_db	1 row(s) affected	0.012 sec
21:34:34	use food_business_db	0 row(s) affected	0.000399 sec
21:34:55	CREATE TABLE Customer(0 row(s) affected	0.035 sec
21:40:42	CREATE TABLE LoginActivity(0 row(s) affected	0.019 sec
21:40:48	CREATE TABLE DeviceDetails(0 row(s) affected	0.016 sec
21:40:54	CREATE TABLE SellerDetails(0 row(s) affected	0.011 sec
21:40:58	CREATE TABLE FoodListing(0 row(s) affected	0.017 sec
21:41:04	CREATE TABLE PurchaseTransactions(0 row(s) affected	0.019 sec

Create “part_2_insert.sql” with the help of insert_data_script created insert files.

https://github.com/ajay9889/de_assignment/blob/main/src/part_2_insert.sql



```
1 use food_business_db;
2
3 -- Insert customer records with status approved and rejected
4
5 INSERT INTO Customer (cust_name, Email, is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
6     'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
7     'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
8     'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
9     'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
10    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
11    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
12    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
13    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
14    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
15    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
16    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
17    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
18    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
19    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
20    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
21    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
22    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
23    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
24    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
25    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
26    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
27    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
28    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
29    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
30    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
31    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
32    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
33    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
34    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
35    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
36    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
37    'Customer', 'Email', is_active, registration_status, reject_reason, password, mobile, gender, created_at) VALUES (
```

Overview Summary and Running the Application Local Machine:

The project is uploaded on GitHub in a public repository.

> https://github.com/ajay9889/de_assignment

> Open it into the Visual Studio Code for Running the Python Script

> python3 part_1_api.py

```
# API Integrations and DataModeling.
```

```
Technical project for analysis.
```

```
# part_1_api.py
```

```
Company A's Finance department requires historical exchange rate data  
for analysis.
```

```
This data is publicly available and a data source, MAS, was  
identified.
```

```
> src/part_1_api.py
```

```
How to run the Applications:
```

```
> Install Python by going to the official website >
```

```
https://www.python.org/downloads/.
```

```
> Verify the installation using command> python3 --version
```

```
> Run the App by using the command > python3 src/part_1_api.py
```

```
# Part 2: Data Modeling for Food Business
```

```
![alt  
text] (https://github.com/ajay9889/de\_assignment/blob/main/src/part\_2\_er.jpg)
```

Create the Table

- > Install MySQL WorkBench
- > Create Database
- > Run all CREATE TABLE Command which is mentioned in the file src/part_2_create.sql
- > Insert required data with respect to each created table > run insert query one by one in the file src/part_2_insert.sql

Part 3: Business Insight

- A. How many accounts are pending approval, approved and rejected?
> src/part_3_select_A.sql
- B. What are the total logins and purchases for each account?
> src/part_3_select_B.sql
- C. How many accounts have purchases by the following buckets? In percentage.
> src/part_3_select_C.sql
- D. What is the third most purchased item in terms of quantity?
> src/part_3_select_D.sql
- E. Which day (Mon - Sun) have the highest logins? If there are draws, to provide all.
> src/part_3_select_E.sql