Data Engineering Task

Part 1: Understanding the data from two endpoints

There are 2 endpoints mentioned in the challenge document:

URL 1:

https://619ca0ea68ebaa001753c9b0.mockapi.io/evaluation/dataengineer/jr/v1/users

URL 2:

https://619ca0ea68ebaa001753c9b0.mockapi.io/evaluation/dataengineer/jr/v1/messages

For URL 1:

This endpoint contains the user data in JSON format(**open standard file format and data interchange format** that uses human-readable text to store and transmit data objects consisting of attribute-value pairs).


```
{
   "createdAt":"2021-11-23T16:10:33.614Z",
   "updatedAt":"2021-11-24T13:34:15.404Z",
   "firstName":"Levi",
   "lastName":"Bins",
   "address":"Ernestine Shore",
   "city":"Pembroke Pines",
   "country":"United States",
   "zipCode":"05734",
   "email":"Kianna_Nicolas@hotmail.com",
   "birthDate":"2020-12-16T02:41:21.036Z",
   "profile":{
      "gender":"male",
      "isSmoking":true,
      "profession":"Central Configuration Planner",
      "income":"3709.61"
   },
   "subscription":[
      {
         "createdAt":"2021-11-24T16:58:46.581Z",
         "startDate":"2021-11-24T05:12:49.301Z",
         "endDate":"2022-09-15T06:05:59.630Z",
         "status":"Active",
         "amount":"43.18"
      }
   ],
   "id":"1"
}
```


Fig. 1. users endpoint data example

Deep diving into the JSON in fig. 1, we can understand that this is the user data as

	Field	Sub-field	Comments	Additional Comments
1	createdAt		Contain date and time when the user was created	

2	updatedAt		Contain date and time when any update occurs in the user's account	
3	firstName		User's first name	(*)
4	lastName		User's last name	(*)
5	address		User's address	Can be null. (*)
6	city		User's city	Can be null. (*)
7	country		User's country	Can be null. (*)
8	zipCode		User's location zip code	Can be null. (*)
9	email		User's email	Can be null. We can record only the email domain and not the user name.
10	birthdate		User's birthdate	Can be null. (*)
10	profile	gender	User's gender	The sub-field can contain null values.
11	profile	isSmoking	Contain 'true' if user smokes or 'false' if the user doesn't smoke. Can null(field contains no data)	
12	profile	profession	User's profession	
13	profile	income	User's income (*)	
14	subscription	createdAt	Subscription details: subscription created at(can not be null)	subscription field can be empty. Eg. 'subscription':[]
15	subscription	startDate	Subscription details: subscription start date(can not be null)	
16	subscription	endDate	Subscription details: subscription end date(can not be null)	
17	subscription	status	Subscription details: subscription status(Can be one of the values: Active, Inactive, Rejected)	
18	subscription	amount	Subscription details: subscription amount/ charge	
19	id		User ID(Suppose to be unique for all users)	Can not be null.

(*) The rows in pink color indicate that this is sensitive data and we need to consider PII so that this information is not accessed by the final users. To achieve this purpose, we will be applying AES Encryption on those marked fields.

For URL 2:

This endpoint contains the message data in JSON format as shown in fig. 2.

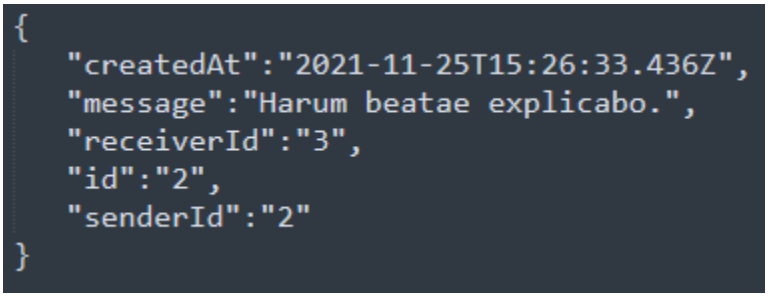

Fig. 2. messages endpoint data example

Deep diving into the JSON in fig. 2, we can understand that this is the message data as

Sr.	Field	Sub-field	Comments	Additional Comments
1	createdAt		Contain date and time when the message was sent.	This field cannot be null.
2	message		Message to send	This data is not imported as this is extremely sensitive information
3	receiverId		Receiver's user id	This field cannot be null.
4	senderId		Sender's user id	This field cannot be null.

Part 2: Collect all the data just once from these endpoints and create/populate the tables.

Assumptions:

1. In the users response, id is considered as the primary key and will be unique for all users. If the same id comes again in the users API response, the data for this id won't be inserted in the database.
2. In the users table, email is assumed to be a valid email id.
3. In the response generated by the users endpoint, the subscription filed can either be blank or must contain all the fields (createdAt, startDate, endDate, status, amount).

4. If users responses do not contain the data for the fields ('firstName', 'lastName', 'address', 'zipCode', 'birthDate', 'city', 'country', 'email') and under the 'profile' for the subfields ('income', 'isSmoking', 'profession', 'gender'), then the default value for these fields will be null(empty).
5. In the response generated by the messages endpoint, the response must have all the fields (id, createdAt, receiver_id, sender_id).

Pointers for the data:

1. For the email field, the user name part is not imported and only the email domain is stored in our system.
2. Encrypted data fields: firstName, lastName, birthdate, address, zipcode, income
3. If the value of isSmoking sub-field in the profile field is true, the value in the database table will be 1. If the value is false, the value in the database table will be 0.

Data modeling:

After understanding the responses generated by the two mentioned endpoints, we can start data modeling.

We need tables to record the following:

1. To store user data
2. To store subscription data for the user
3. To store message data

users	subscriptions	message_table

The following ER diagram represents the attributes for the above three tables:

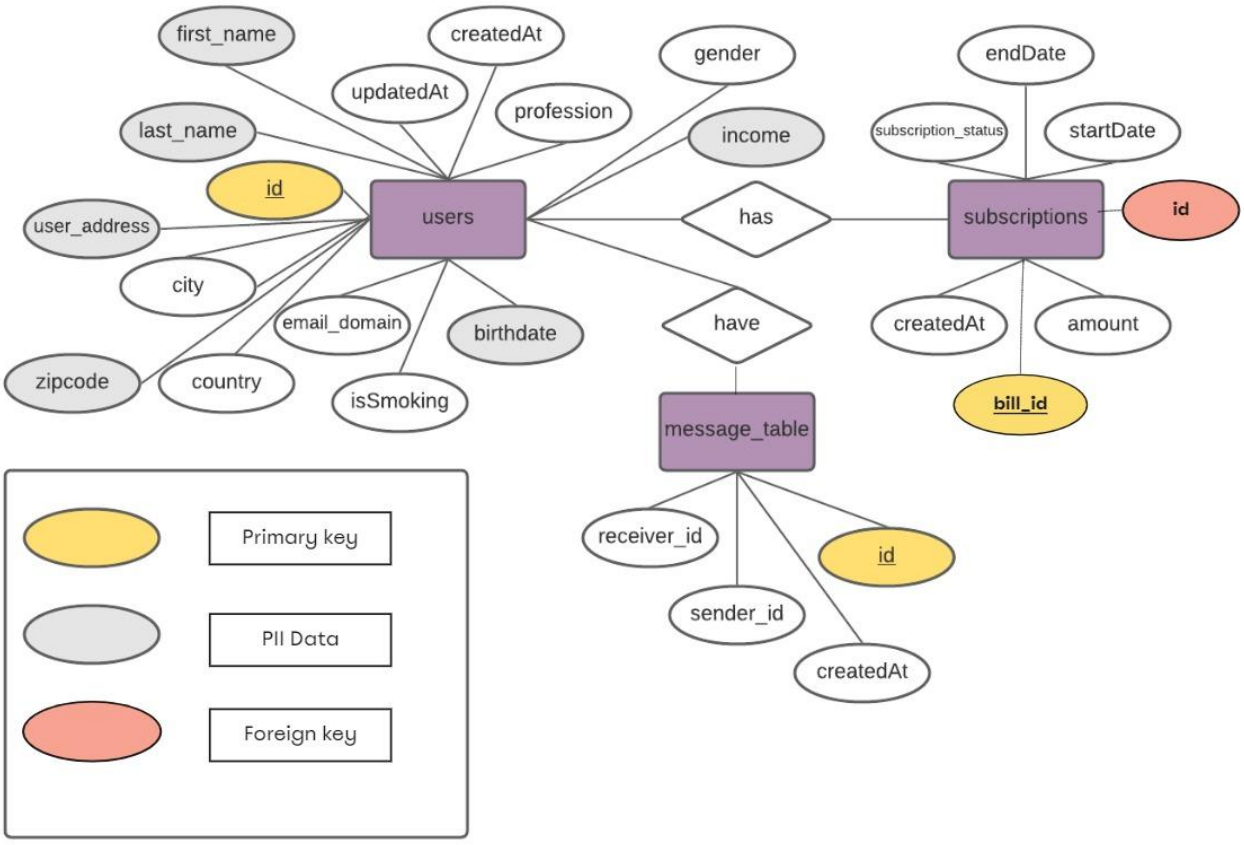

Fig. 3. ER Diagram

Why MS SQL Server as a database?

1. Database Supported Platforms: Linux, Mac OS X, Microsoft Windows Server, and Microsoft Windows operating systems
2. SQL server syntax is simple and easy to use.
3. While backing up the data, the database is not blocked. This allows users to complete the backup and data restoration process completed in less time and effort.
4. Supports row-based filtering which filters out the records on a database by database way. Gives the advantage of filtering multiple rows without considering the number of databases. Moreover, the filtered data is kept in a separate distribution database.
5. SQL Server is more secure than MySQL. It does not let any process access and manipulate the database files at run time. Users need to perform specific functions or manipulate files by executing an instance. This prevents hackers to access or manipulate the data directly.

Data encryption using AES algorithm:

AES Encryption algorithm in short:

- Encryption works by taking plain text and converting it into ciphertext, which is made up of seemingly random characters.
- Only those who have the special key can decrypt it.
- AES uses symmetric key encryption, which involves the use of only one secret key to cipher and decipher the information.

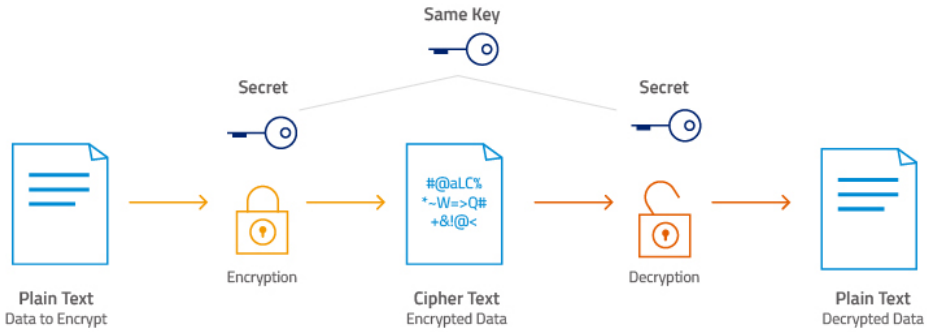

Fig. 4. Symmetric Key Encryption working [check for reference]

Python modules used:

Module name	Purpose	Reference link
requests	To connect to the API endpoint	requests documentation
pyodbc	To connect to MS SQL Server database	pyodbc documentation
unittest	To perform unit testing	unittest documentation

Work Flow:

High-level Design:

Fig. 5. High-level design for data insertion

Flowchart:

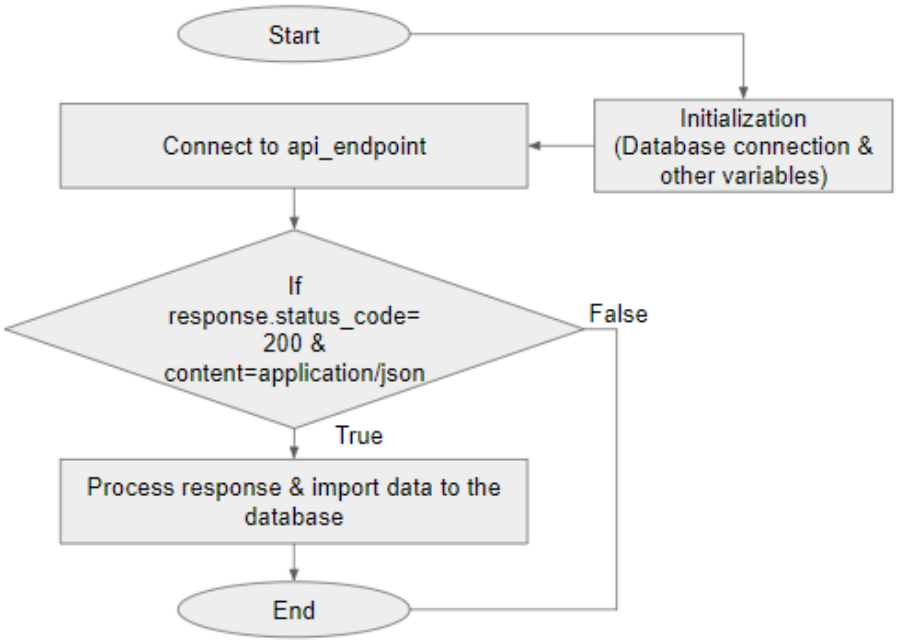

Fig. 6. Flowchart to collect the data from the endpoints

Part 3: Project Folder Structure & Environment Details

Project Folder Structure:

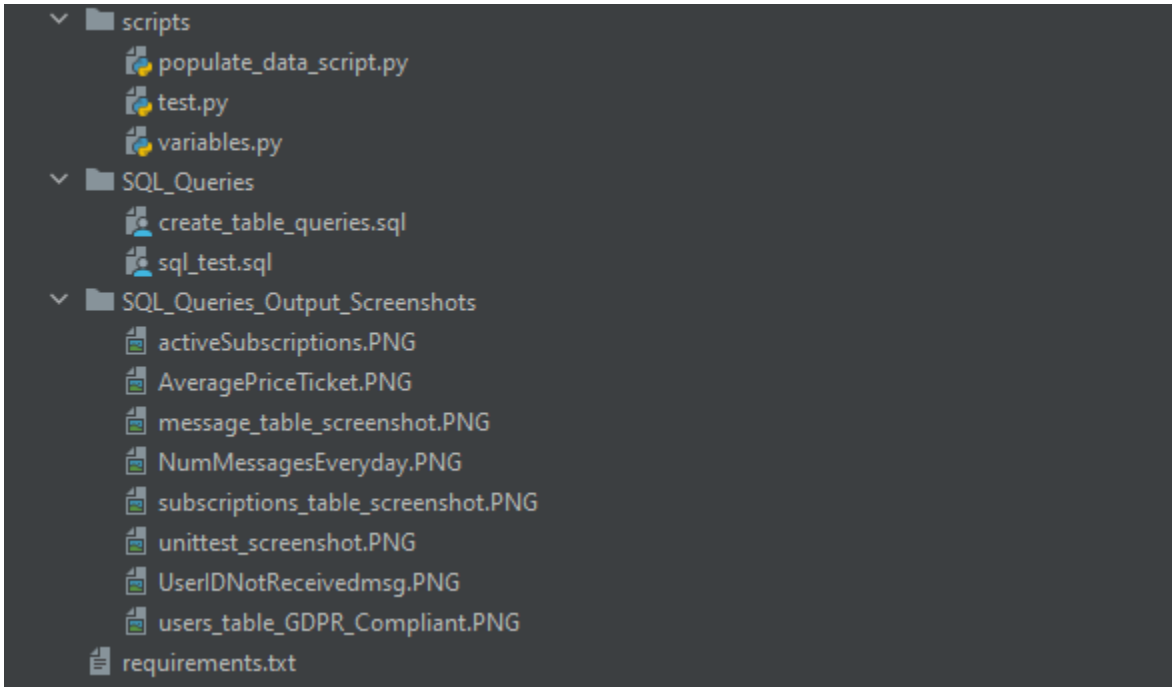

Fig. 7. Project Folder Structure

Environment Details:

Operating System: Windows 10 64 bit

Programming Language: Python 3.8.6

Text Editor to write the code: Sublime Text Editor

Database: MS SQL Server 18.10

Database GUI Software: MS SQL Server 2019 Developer Edition(Free)

Part 4: How to reproduce the project

Installing MS SQL Server

Go to MS SQL Server download link.

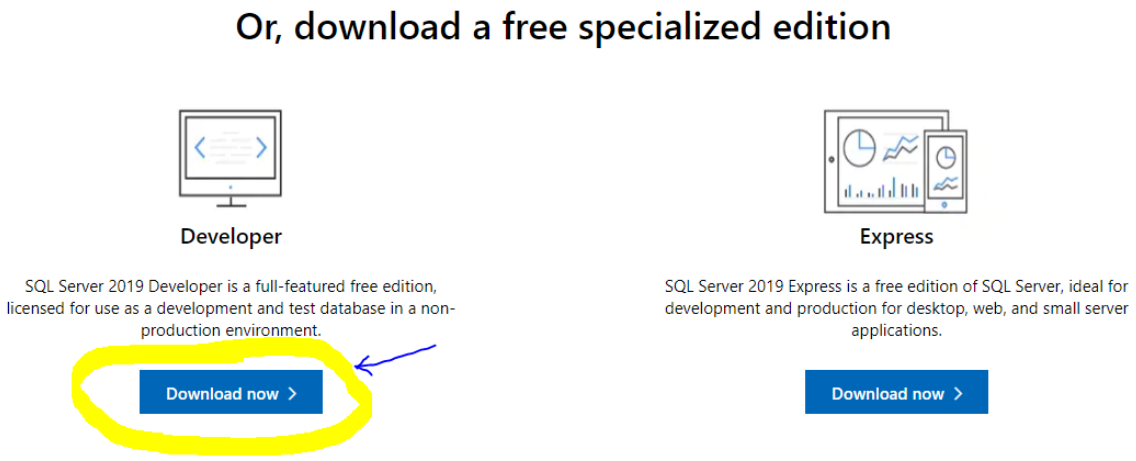

Fig. 8. MS SQL Server download page

Once the setup file is downloaded, install the setup on the system.

Installing MS SQL Server Management Studio

Go to MS SQL Server Management Studio download link.

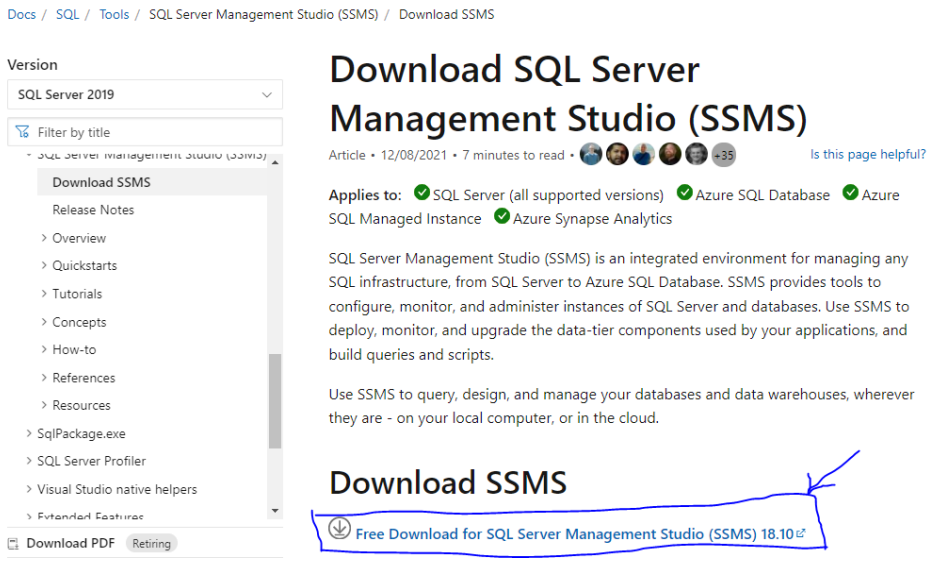

Fig. 8. MS SQL Server Management studio download page

Once the setup file is downloaded, install the setup on the system.

Cloning the project from GitHub:

Step 1: Using the Git CLI, copy the following command:

git clone https://github.com/itsvaishnavi/Data_Import_From_API.git

To download git, visit https://git-scm.com/downloads

Step 2: Navigate to the folder where the project folder is cloned. Open the command prompt at this path and run the following command:

pip install -r requirements.txt

This downloads all the essential python modules required for the project.

Step 3: Navigate to the **SQL_Queries** folder and open the **create_table_queries.sql** file in MS SQL Server Management Studio.

Run the queries one by one in order to create the database, tables, and encryption key.

Step 4: We are now ready to run the python scripts. There are three Python scripts.

File Name	Parent folder	Comments
populate_data_script.py	scripts	Script to collect response from the API endpoints and store the processed response in the database.
variables.py	scripts	Script where all the constant values are defined. Eg. database name
test.py	scripts	Script contains the unit test cases.

To collect the data from the API endpoints and populate the tables, navigate to the **scripts** folder and run the following command to run populate_data_script.py:

python populate_data_script.py

To run the test cases, navigate to the **scripts** folder and run the following command to run test.py:

python -m unittest

Part 5: Scenario-based SQL queries

The Product Owner asked you to provide the queries for some scenarios, please add a file sql_test.sql in your project with the queries that solve the below questions:

• How many total messages are being sent every day?

• Are there any users that did not receive any message?

• How many active subscriptions do we have today?

• How much is the average price ticket (sum amount subscriptions/count subscriptions) breakdown by year/month (format YYYY-MM)?

Navigate to the **SQL_Queries** folder and open the **sql_test.sql** file in MS SQL Server Management Studio.

Run the queries one by one for each scenario.

Author

Vaishnavi Piyush Kand