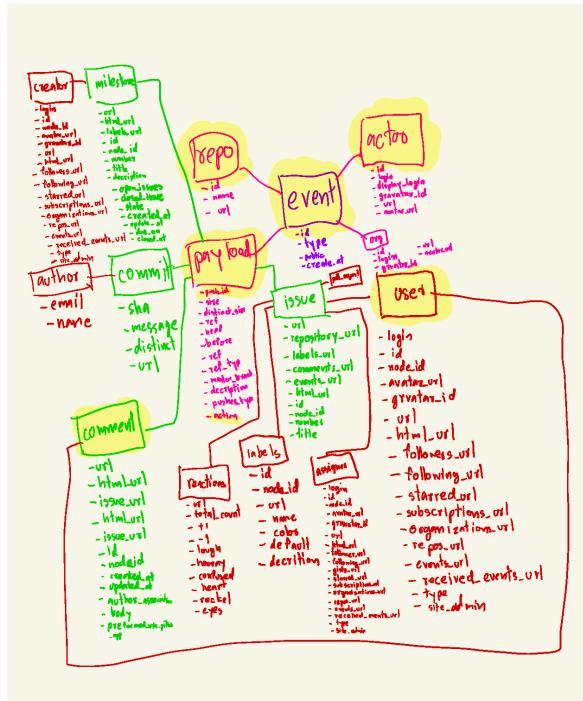


## การสร้างแบบจำลองข้อมูลด้วย Cassandra (NoSQL)

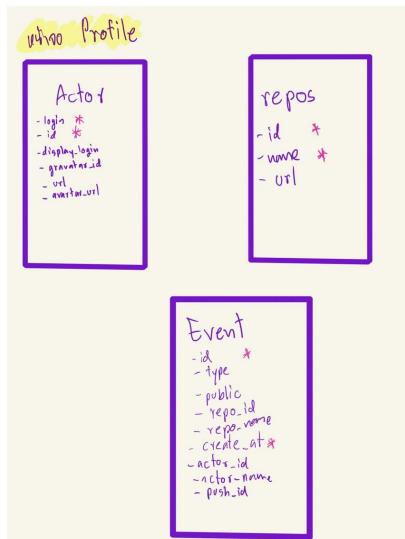
## วิธีการดำเนินการ

## 1. วิเคราะห์โครงสร้างข้อมูลของ JSON ไฟล์ ได้ดังรูป



จากภาพ ข้อมูลใน Json เป็นข้อมูล เหตุการณ์ หรือ Event ของ Github โดยมีโครงสร้างที่ยืดหยุ่น โดยแต่ละ record จะมีข้อมูลที่ไม่เท่ากัน และไม่เหมือนกัน

2. ดำเนินการออกแบบตาราง ได้ตารางดังนี้ (โดยทุกตารางไม่มีการเข้ามายิงกัน และทุกตารางมีการกำหนด Partition Key เพื่อการจัดกลุ่มข้อมูล และ Clustering Key เพื่อการจัดเรียงข้อมูลที่อยู่ใกล้เคียงกัน รวมเป็น Primary Key เพื่อทำให้การ Query เข้าถึงได้อย่างรวดเร็ว)



โดย ตาราง event คือ ตารางที่เก็บข้อมูลการดำเนินการหรือ Activity ต่างๆ ของผู้ใช้งาน เช่น มีการ push สิ่งใด จำนวนกี่ครั้ง โดย Partition Key คือ type เนื่องจาก Type เป็นลักษณะของหมวดหมู่อยู่แล้ว และ Clustering Key คือ Create\_at เป็นลักษณะของวันและเวลา หมายความว่าการนำมารอท์ต์ข้อมูล ตาราง repo คือ ตารางที่เก็บข้อมูลในส่วนของพื้นที่ workspace ของแต่ละคน โดย Partition Key คือ id เนื่องจากตารางนี้เป็นตารางเก็บข้อมูลพื้นที่การทำงานของแต่ละ actor 1 actor สามารถมีได้หลาย repo 1 repo สามารถแชร์การทำงานร่วมกันได้ และ Clustering Key คือ name จากในตารางไม่มีข้อมูล Datetime จึงใช้ name ควบคู่ไปกับ id ตาราง actor คือ ตารางที่เก็บข้อมูลเจ้าของ Profile โดย Partition Key คือ id เนื่องจากตารางนี้เป็นตารางเก็บข้อมูลต่างๆ ของแต่ละ actor 1 และ Clustering Key คือ login จากในตารางไม่มีข้อมูลDatetime จึงใช้ Login ควบคู่ไปกับ id

- ทำการเตรียม Gitpod สำหรับการเขียนโปรแกรม โดยทำการ Sync Github account และ Gitpod เข้าด้วยกัน และทำการเตรียม environment

```

14
15     create_table_queries = [
16         """
17             CREATE TABLE IF NOT EXISTS staging_events (
18                 id varchar,
19                 type varchar,
20                 actor_id bigint,
21                 actor_login varchar,
22                 actor_url varchar,
23                 repo_id bigint,
24                 repo_name varchar,
25                 repo_url varchar,
    
```

JUPITER: VARIABLES

Name	Type
No variables defined	

TERMINAL

```

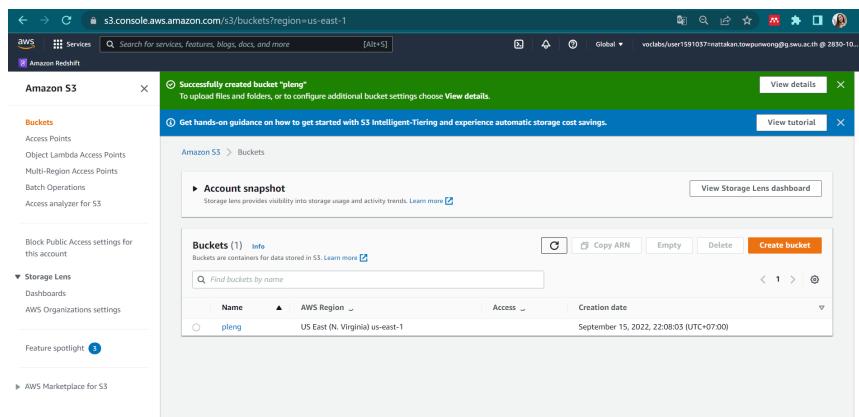
gitpod /workspace/swu-ds525/03-building-a-data-warehouse (main) $ python -m venv ENV
(ENV) gitpod /workspace/swu-ds525/03-building-a-data-warehouse (main) $ pip install -r requirements.txt
Requirement already satisfied: numpy==1.23.2 in ./ENV/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (1.23.2)
Requirement already satisfied: psycopg2==2.9.3 in ./ENV/lib/python3.8/site-packages (from -r requirements.txt (line 2)) (2.9.3)
Requirement already satisfied: python-dateutil==2.8.2 in ./ENV/lib/python3.8/site-packages (from -r requirements.txt (line 3)) (2.8.2)
Requirement already satisfied: pytz==2022.2.1 in ./ENV/lib/python3.8/site-packages (from -r requirements.txt (line 4)) (2022.2.1)
Requirement already satisfied: six==1.16.0 in ./ENV/lib/python3.8/site-packages (from -r requirements.txt (line 5)) (1.16.0)
WARNING: You are using pip version 22.0.4; however, version 22.2.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(ENV) gitpod /workspace/swu-ds525/03-building-a-data-warehouse (main) $
    
```

## 4. ทำการแก้ไขข้อมูลในไฟล์ events\_json\_path.json ให้ตรงกับที่ออกแบบ

```

1 { "jsonpaths": [
2     "$.id",
3     "$.type",
4     "$.actor.id",
5     "$.actor.login",
6     "$.actor.url",
7     "$.repo.id",
8     "$.repo.name",
9     "$.repo.url",
10    "$.public",
11    "$.created_at",
12    "$.actor.display_login",
13    "$.actor.gravatar_id",
14    "$.actor.url",
15    "$.payload.push_id",
16 ]
    }
    
```

## 5. เข้าไปที่ Amazon S3 สร้าง Bucket สำหรับเก็บไฟล์ต่างๆ พร้อมทั้ง Upload ไฟล์ events\_json\_path.json และ github\_events\_01.json



The screenshot shows the AWS S3 console with a successful upload message. The destination is s3://pleng. The summary table shows 2 files uploaded successfully (Succeeded) and 0 files failed. Below is a table of the uploaded files:

Name	Type	Size	Status	Error
events_json_path.json	application/json	125.0 B	Succeeded	-
github_events_01.json	application/json	68.5 KB	Succeeded	-

## 5. เข้าไปที่ Amazon Redshift เพื่อสร้าง Cluster รายละเอียดดังรูป

The screenshots show the process of creating a new Amazon Redshift cluster:

- Create cluster - Step 1: Cluster configuration**  
Cluster Identifier: redshift-cluster-1  
Plan: Production (selected)  
Node type: rd.s3.xlarge  
Number of nodes: 1
- Create cluster - Step 2: Admin user setup**  
Admin user name: password  
Admin user password: Plan06720900  
Show password
- Create cluster - Step 3: Cluster permissions**  
Associated IAM roles (0): Create, associate, or remove an IAM role. You can associate up to 50 IAM roles. You can also choose an IAM role and set it as the default for this cluster.  
Set default: Manage IAM roles
- Create cluster - Step 4: Additional configurations**  
Network: Using default VPC (sg-0174ef7614f66700) and default subnet.  
Backup: Automated snapshots are created about every eight hours or following every 5 GB per node of data changes, whichever comes first.  
Maintenance: Using current maintenance track.  
Configuration: Using default.redshift-1.0 parameter group with no database encryption.

In the final step, there is a "Create cluster" button at the bottom right.

## 6. เมื่อได้สถานะ Available ให้ทำการกำหนดให้ Cluster สามารถเข้าถึงได้ โดย Enable publicly accessible

The figure consists of three vertically stacked screenshots of the AWS Redshift console. Each screenshot shows the 'General information' section for a cluster named 'redshift-cluster-1'. In the first screenshot, the 'Status' is listed as 'Available'. In the second screenshot, a context menu is open over the 'Actions' button, and the 'Modify publicly accessible setting' option is highlighted. In the third screenshot, a modal dialog titled 'Edit publicly accessible' is displayed, showing the 'Enable' radio button selected under 'Publicly accessible'. A warning message at the bottom of the dialog states: '⚠ Your cluster might be unavailable for up to 10 minutes while this change to public accessibility is processed.' At the bottom right of the dialog are 'Cancel' and 'Save changes' buttons.

## 7. ทำการสร้างการเชื่อมต่อกับ Database

A screenshot of the AWS Redshift console showing the 'Connect to database' dialog. The dialog has several sections: 'Connection' (with 'Create a new connection' selected), 'Authentication' (with 'Temporary credentials' selected), 'Database' (with 'dev' selected from a dropdown), and 'Schema' (with 'public' selected). At the bottom right of the dialog are 'Cancel' and 'Connect' buttons.

## 8. ทำการ Copy S3 ไฟล์ทั้ง 2 ไฟล์ และ ARN ของ IAM ไปวางที่ไฟล์ etl.py

The first screenshot shows the AWS S3 console with the file 'github\_events\_01.json' selected. A 'Copy S3 URI' button is highlighted, indicating the action to copy the file's URL.

The second screenshot shows the AWS IAM Roles page for 'LabRole'. The ARN of the role is copied, as indicated by the 'ARN Copied' message.

The third screenshot shows a code editor with the file 'etl.py' open. It contains Python code for copying data from S3 to Redshift. The copied ARN is pasted into the code, specifically into the 'aws\_iam\_role' parameter of the 'copy\_table\_queries' function.

```

21     """,
22     ],
23     copy_table_queries = [
24         """
25         COPY staging_events FROM 's3://pleng/github_events_01.json'
26         CREDENTIALS 'aws_iam_role=arn:aws:iam::283010062450:role/LabRole'
27         JSON 's3://pleng/events_json_path.json'
28         REGION 'us-east-1'
29         """,
30         insert_table_queries = []
31         """
32         INSERT INTO
33             events (

```

## 9. ทำการแก้ไขรายละเอียดส่วน Function Main ของไฟล์ etl.py เพื่อเชื่อมโยงกับ Amazon Redshift

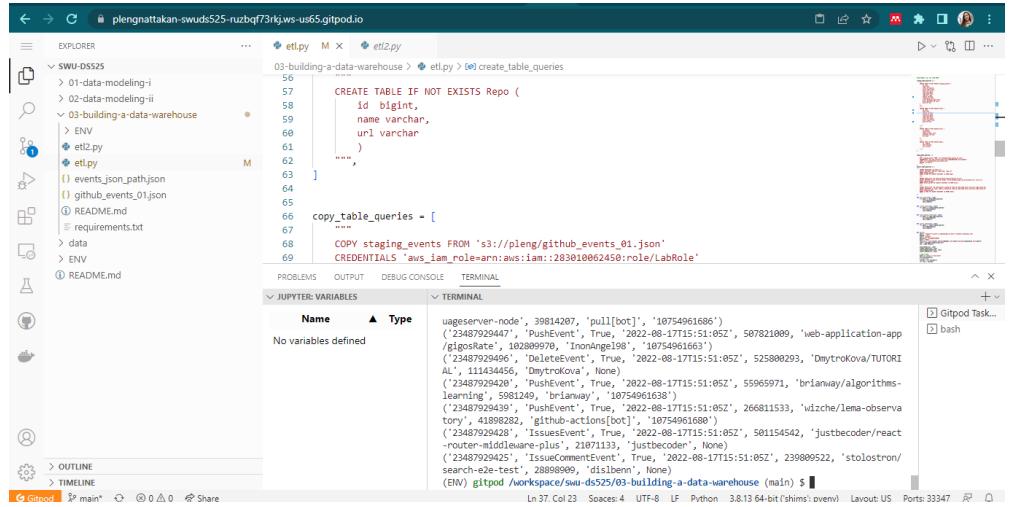
The screenshot shows the 'etl.py' file in a code editor. The 'main()' function has been modified to connect to an Amazon Redshift cluster. The 'host' variable is set to the cluster's endpoint, and the 'dbname', 'user', 'password', and 'port' variables are defined. The 'conn\_str' variable is constructed using these parameters, and a connection is established using 'psycopg2.connect'. Finally, the 'drop\_tables' function is called with the cursor and connection objects.

```

119
120
121     def main():
122         host = "redshift-cluster-1.ci0boaeqvdep.us-east-1.redshift.amazonaws.com"
123         dbname = "dev"
124         user = "awsuser"
125         password = "Pleng056720990"
126         port = "5439"
127         conn_str = f"host={host} dbname={dbname} user={user} password={password} port={port}"
128         conn = psycopg2.connect(conn_str)
129         cur = conn.cursor()
130
131         drop_tables(cur, conn)

```

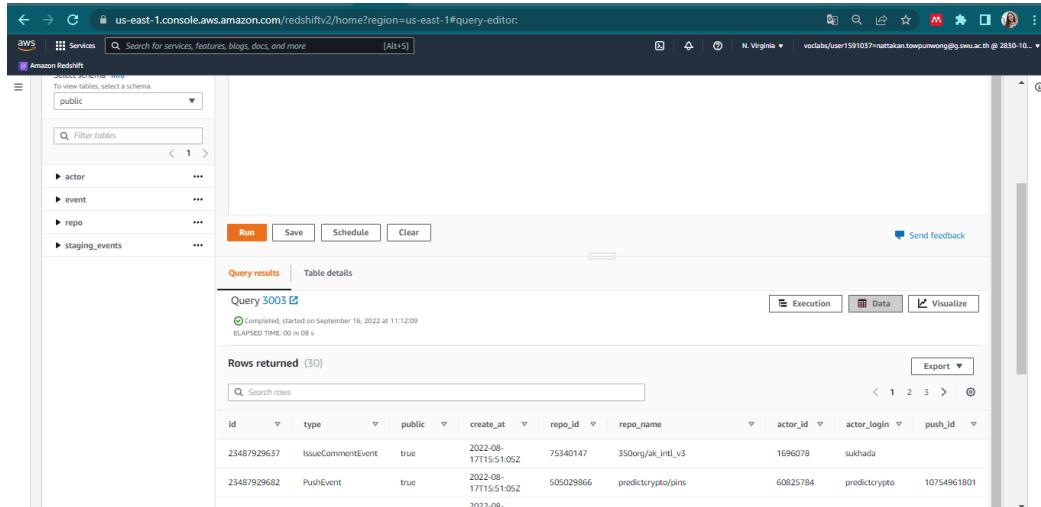
## 10. ทำการเขียนโปรแกรมเพื่อสร้างตาราง Insert ข้อมูล และ Query ข้อมูลจากไฟล์ JSON แล้ว Run โดยไปตรวจสอบที่หน้าจอ Amazon



```

CREATE TABLE IF NOT EXISTS Repo (
    id bigint,
    name varchar,
    url varchar
)

copy_table_queries = [
    """
    COPY staging_events FROM 's3://pleng/github_events_01.json'
    CREDENTIALS 'aws_iam_role=arn:aws:iam::283010062450:role/LabRole'
    """
]
    
```



Query results

id	type	public	create_at	repo_id	repo_name	actor_id	actor_login	push_id
23487929637	IssueCommentEvent	true	2022-08-17T15:51:05Z	75340147	3500org/ak_intl_v\$	1696078	sukhada	
23487929682	PushEvent	true	2022-08-17T15:51:05Z	505029866	predictcrypto/pins	60825784	predictcrypto	10754961801