



โครงการเรียนรู้ตลอดชีวิตและพัฒนาทักษะเพื่ออนาคต (Upskill/Reskill)

หลักสูตรวิศวกรรมข้อมูลขั้นพื้นฐาน
Basic Data Engineering

การจัดการข้อมูลพิกัดบนแผนที่ ประเทศไทย **รุ่นที่ 1**

วันที่ 16 ธันวาคม 2566 เวลา 09.00 - 17.00 น.
โดย

รณัทเมศรี รณรัตน์นันท์

gis

บริษัทจีไอเอส จำกัด
แผนก: GISC/STI/APD

สนับสนุนโดย กระทรวงการอุดมศึกษา วิทยาศาสตร์ วิจัยและนวัตกรรม (อว.)

เค้าโครง

ความสำคัญและที่มาของโครงการ : งาน GIS เป็นงานที่ต้องความถูกต้องของข้อมูลในเรื่องของ Geometry type สูงเนื่องจาก หากมี Geometry ที่ผิดรูปแบบบางครั้งอาจจะไม่สามารถ Plot ข้อมูลไปบนแผนที่ได้ หรือการทำอะไรบางอย่างจะผิดเพี้ยนได้ทันที

วัตถุประสงค์ : กรองพิกัดที่อยู่ภายนอกประเทศไทย และจังหวัดที่ไม่ได้สนใจออกจากชุดข้อมูล

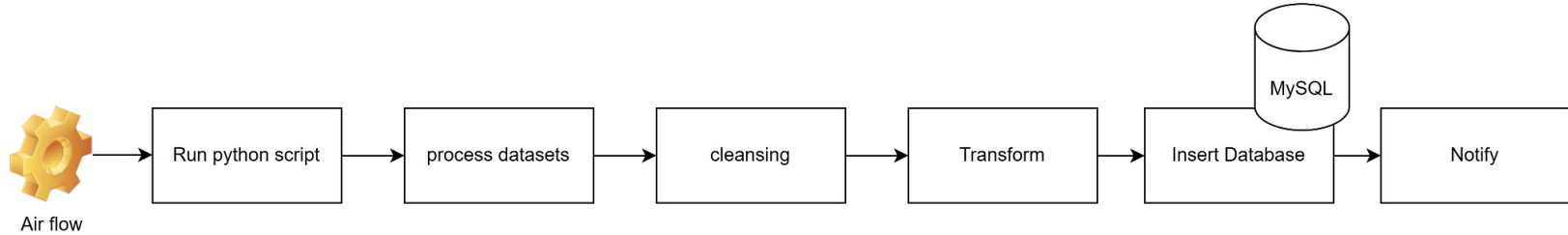
การดำเนินงาน :

- Cleansing data
- Transform data
- Database/Data warehouse/Management
- ETL Flow

สรุปผลการดำเนินงาน :

- สามารถประมวลผลข้อมูลได้อย่างถูกต้อง
- Flow การทำงานเป็น Automation มากขึ้น

Flow การดำเนินงาน



ภาพรวมการทำงานของ Program เริ่มต้นที่ Airflow มีการตั้ง Task scheduler ไว้เป็นการทำงานทุกวัน และทุกๆ 5 นาที เมื่อโปรแกรม Launch ขึ้นมาจะ Process datasets > Cleansing data เกี่ยวกับค่า Nan, ตรวจสอบความถูกต้องของข้อมูล และกรองฟังก์ชันที่ไม่ได้สนใจออกไป > Transform geometry column เพื่อนำไปใช้กับ Library Geopandas ในการประมวลผลข้อมูลแผนที่ > เมื่อประมวลเสร็จจะนำผลลัพธ์กลับมา Insert ลงฐานข้อมูล > แจ้ง Notify ผ่าน Line application

Coding - Import library

- Step1: ทำการ Import library ที่ใช้ในการประมวลผล

```
+ Code + Markdown | ▶ Run All ↺ Restart ☰ Clear All Outputs | 📄 Variables ☰ Outline ...
```

```
# Process
import pandas as pd
import numpy as np
import geopandas
import shapely.wkt
import folium
import mysql.connector
import json

# Set env
import os
os.environ["USE_PYGEOS"] = "0"
import contextily as cx
```

[1] ✓ 9.3s

Coding - Load datasets

- Step2: ใช้ Pandas ในการจัดการข้อมูล datasets และทำการ Print ส่วนหัว 5 รายการ

```
Presentation > Present-notebook-process.ipynb > df = pd.read_csv('./POINT_20181024_1.csv')
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...

df = pd.read_csv('./POINT_20181024_1.csv')
[2] ✓ 0.0s

# หัว
df.head()
[3] ✓ 0.0s

...
  OPERATION_TYPE  SITE_CELL_ID  SITE_ID  CELL_STATUS  UTILIZATION PEAK TIME  LATITUDE  LONGITUDE
0              I      909664.0  429371.0      ACTIVE      23/10/2018 17:00    13.42507    99.955180
1              I      909665.0  429371.0      ACTIVE      23/10/2018 16:00    13.42507    99.955180
2              I      909666.0  429371.0      ACTIVE      23/10/2018 17:00    13.42507    99.955180
3              I      909671.0  429372.0      ACTIVE      23/10/2018 19:00    13.42537    99.954059
4              I      909672.0  429372.0      ACTIVE      23/10/2018 10:00    13.42537    99.954059
```

Coding - Load datasets (ต่อ)

- Step2 (ต่อ): Print ส่วนท้าย 5 รายการ

```
Present-notebook-process.ipynb •
Presentation > Present-notebook-process.ipynb > df = pd.read_csv('./POINT_20181024_1.csv')
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables 📄 Outline ...
```

```
▶ ▾
# ท้าย
df.tail()
```

```
[4] ✓ 0.0s
```

	OPERATION_TYPE	SITE_CELL_ID	SITE_ID	CELL_STATUS	UTILIZATION_PEAK_TIME	LATITUDE	LONGITUDE
38114	I	916011.0	495781.0	ACTIVE	NaN	5.983300	101.780900
38115	I	915365.0	495623.0	ACTIVE	NaN	13.753514	100.541077
38116	I	915366.0	495623.0	ACTIVE	NaN	13.753514	100.541077
38117	I	915367.0	495623.0	ACTIVE	NaN	13.753514	100.541077
38118	38118	NaN	NaN	NaN	NaN	NaN	NaN

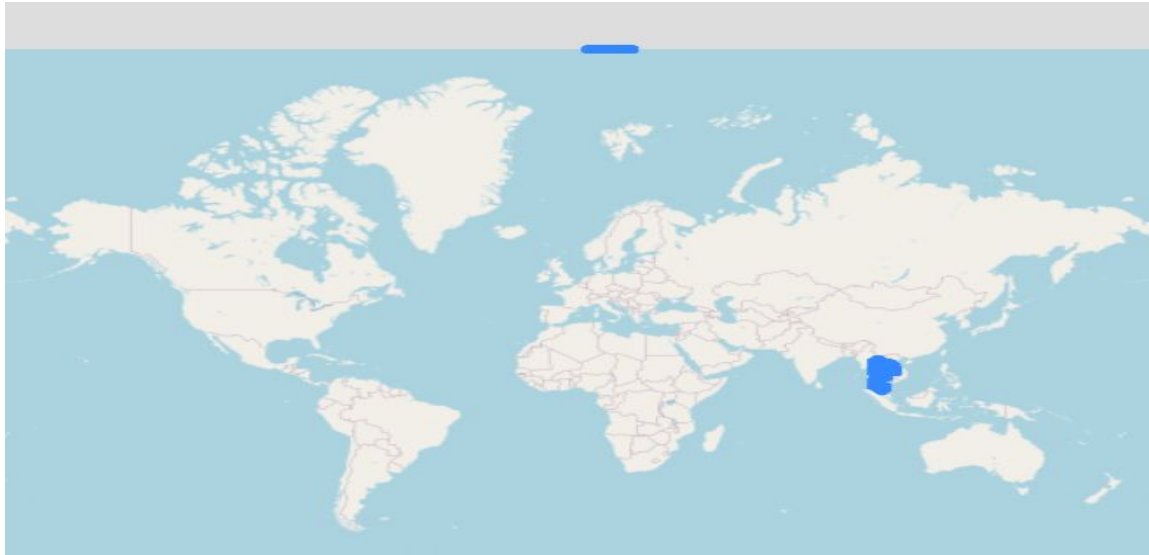
```
df.shape
```

```
[5] ✓ 0.0s
```

```
... (38119, 7)
```

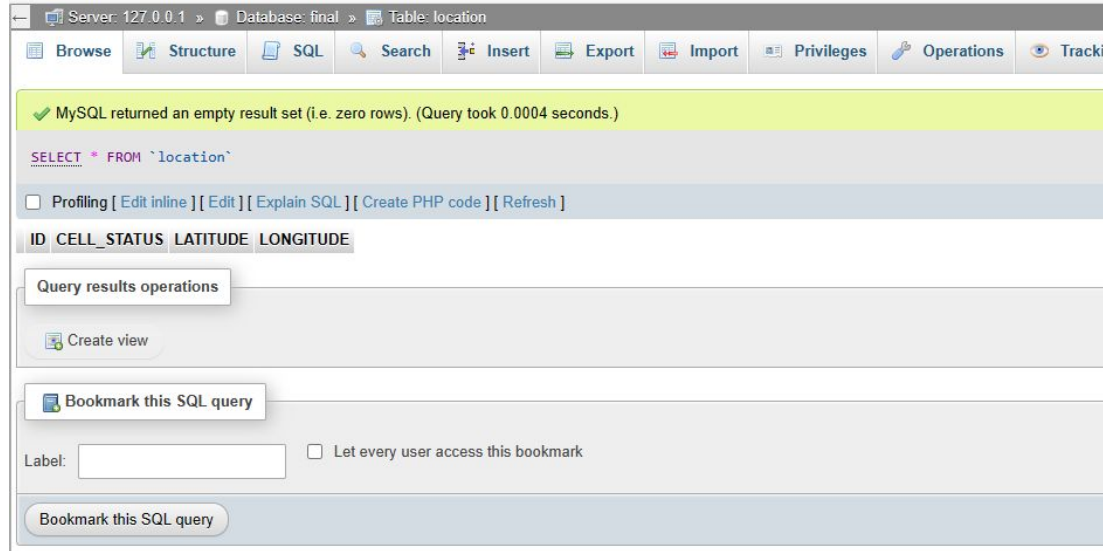
Coding - Before cleansing data

- ภาพตัวอย่างก่อนมีการ Cleansing data จะพบว่ามีพิกัดอยู่ที่ขั้วโลกเหนือ



Coding - Before inserts

- ภาพตัวอย่างก่อนมีการ Insert ข้อมูล



Coding - Check data

- Step3: Check ประเภท Type ของข้อมูลด้วยคำสั่ง info

```
+ Code + Markdown | ▶ Run All ↺ Restart ⌵ Clear All Outputs | 📄 Variables 📄 Outline ...
```

```
▶ df.info()
```

```
[6] ✓ 0.0s
```

```
... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 38119 entries, 0 to 38118
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   OPERATION_TYPE         38119 non-null  object
1   SITE_CELL_ID           38118 non-null  float64
2   SITE_ID                38118 non-null  float64
3   CELL_STATUS            38118 non-null  object
4   UTILIZATION_PEAK_TIME  2144 non-null   object
5   LATITUDE               38118 non-null  float64
6   LONGITUDE              38118 non-null  float64
dtypes: float64(4), object(3)
memory usage: 2.0+ MB
```

Coding - Check data (ต่อ)

- Step4: แสดงข้อมูลสถิติออกมาเพื่อดูความสัมพันธ์ของข้อมูลด้วยคำสั่ง describe()

```
+ Code + Markdown | ▶ Run All ↺ Restart ☰ Clear All Outputs | 📄 Variables ☰
```

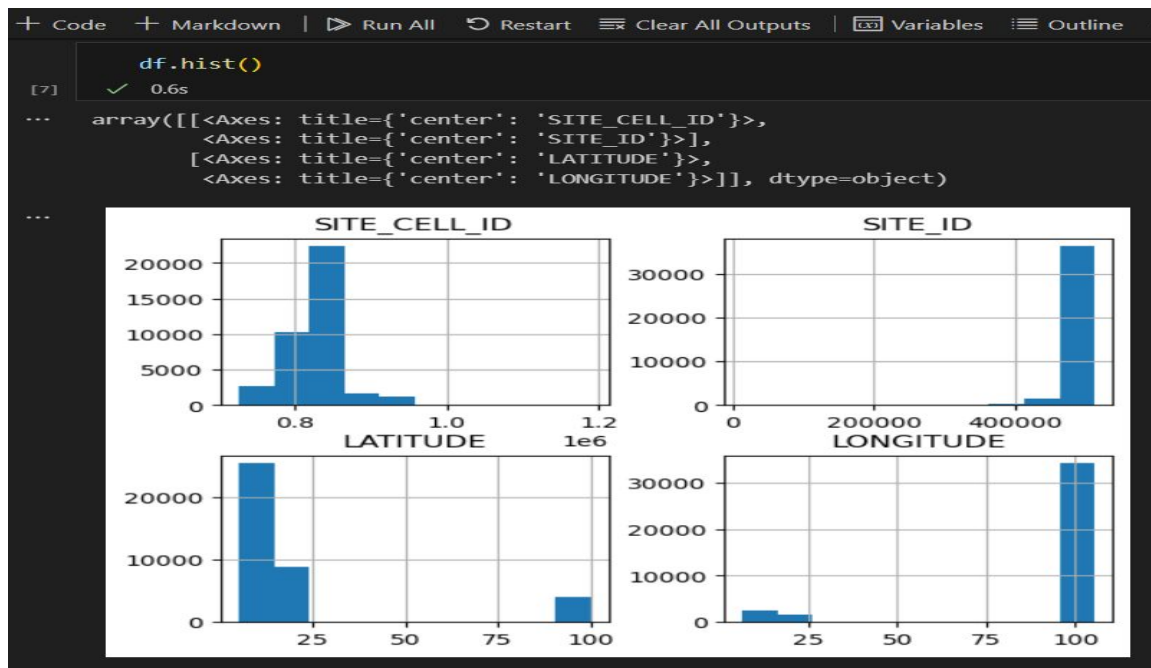
```
[8] ✓ 0.0s
```

```
df.describe()
```

	SITE_CELL_ID	SITE_ID	LATITUDE	LONGITUDE
count	3.811800e+04	38118.000000	38118.000000	38118.000000
mean	8.305706e+05	471155.042159	22.891080	91.773989
std	3.779074e+04	10903.350917	26.208496	27.146992
min	7.273000e+05	14333.000000	5.230000	6.490470
25%	8.134172e+05	468299.000000	13.670670	100.422731
50%	8.374365e+05	473483.000000	13.856475	100.609424
75%	8.500368e+05	476370.750000	15.494410	101.067160
max	1.189442e+06	512669.000000	99.998365	105.466420

Coding - Check data (ต่อ)

- Step5: Plot ข้อมูลสถิติออกมาเพื่อดูความสัมพันธ์ของข้อมูลด้วยคำสั่ง hist()



ขั้นตอน - Cleansing data

- Step6 : ตรวจสอบ Column LATITUDE และ LONGITUDE ว่าพบค่าว่าง หรือไม่

```
[9] df['LATITUDE'].isnull()
✓ 0.0s
...
0      False
1      False
2      False
3      False
4      False
...
38114   False
38115   False
38116   False
38117   False
38118    True
Name: LATITUDE, Length: 38119, dtype: bool

[10] df['LONGITUDE'].isnull()
✓ 0.0s
...
0      False
1      False
2      False
3      False
4      False
...
38114   False
38115   False
38116   False
38117   False
38118    True
Name: LONGITUDE, Length: 38119, dtype: bool
```

Coding - Cleansing data (ต่อ)

- Step7 : เมื่อพบค่าว่างจะใช้ Dropna() เพื่อลบข้อมูลแถวนั้นออกไป

```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | (x) Variables ≡ Outline ...
```

```
# ลบแถวที่ latitude,longitude เป็น Nan
df = df.dropna(subset=['LATITUDE','LONGITUDE'],axis='rows')
```

[11] ✓ 0.0s

Coding - Cleansing data (ต่อ)

- Step8 : ตรวจสอบ Categories ข้อมูล Column CELL_STATUS เพื่อดูว่ามีค่านอกเหนือจาก ACTIVE / DEACTIVE หรือไม่

```
+ Code + Markdown | ▶ Run All ↺ Restart ☰ Clear All Outputs | [x] Variables ☰ Outline ...
```

```
# Check categories
df['CELL_STATUS'].value_counts()
```

[12] ✓ 0.0s

```
... CELL_STATUS
ACTIVE      36111
DEACTIVE    2003
XACTIVE       2
XDEACTIVE     2
Name: count, dtype: int64
```

Coding - Cleansing data (ต่อ)

- Step9 : เมื่อพบว่ามี Categories นอกจากที่ต้องการจะทำการค้นหาข้อมูลที่มี CELL_STATUS ไม่ถูกต้อง และทำการอัปเดตให้เป็นสถานะ DEACTIVE

```
+ Code + Markdown | ▶ Run All ↺ Restart ☰ Clear All Outputs | 📄 Variables 📄 Outline ...
```

```
# Get index of CELL_STATUS is warning
dfs = df[((df['CELL_STATUS'] != 'ACTIVE') & (df['CELL_STATUS'] != 'DEACTIVE'))]
dfs.head()
```

[13] ✓ 0.0s

	OPERATION_TYPE	SITE_CELL_ID	SITE_ID	CELL_STATUS	UTILIZATION	PEAK_TIME	LATITUDE	LONGITUDE
5	I	909673.0	429372.0	XACTIVE	23/10/2018 16:00	13.425370	99.954059	
24	I	915607.0	404617.0	XACTIVE	23/10/2018 11:00	13.711807	100.584305	
52	I	915649.0	431499.0	XDEACTIVE	NaN	13.766166	100.642703	
159	I	915546.0	397828.0	XDEACTIVE	NaN	13.696351	100.753258	

```
# find to set
df.loc[dfs.index, 'CELL_STATUS'] = 'DEACTIVE'
```

[14] ✓ 0.0s

```
# view after set
df.iloc[dfs.index]
```

[15] ✓ 0.0s

	OPERATION_TYPE	SITE_CELL_ID	SITE_ID	CELL_STATUS	UTILIZATION	PEAK_TIME	LATITUDE	LONGITUDE
5	I	909673.0	429372.0	DEACTIVE	23/10/2018 16:00	13.425370	99.954059	
24	I	915607.0	404617.0	DEACTIVE	23/10/2018 11:00	13.711807	100.584305	
52	I	915649.0	431499.0	DEACTIVE	NaN	13.766166	100.642703	
159	I	915546.0	397828.0	DEACTIVE	NaN	13.696351	100.753258	

Coding - Transform data

- Step10 : ทำการ Transform data โดยนำ Column LATITUDE และ LONGITUDE มาสร้างเป็น Column geometry สำหรับนำไปใช้งานประมวลผลงานแผนที่

```
+ Code + Markdown | ▶ Run All ↺ Restart ⌵ Clear All Outputs | 📄 Variables 📄 Outline ... base (Pyth
```

```
# transform column geometry
df['geometry'] = df[['LATITUDE','LONGITUDE']].apply(lambda row: 'POINT (' + row['LONGITUDE'].astype(str) + ' ' + row['LATITUDE'].a
```

[16] ✓ 1.1s

```
df.head()
```

[17] ✓ 0.0s

	OPERATION_TYPE	SITE_CELL_ID	SITE_ID	CELL_STATUS	UTILIZATION	PEAK_TIME	LATITUDE	LONGITUDE	geometry
0	I	909664.0	429371.0	ACTIVE	23/10/2018	17:00	13.42507	99.955180	POINT (99.95518 13.42507)
1	I	909665.0	429371.0	ACTIVE	23/10/2018	16:00	13.42507	99.955180	POINT (99.95518 13.42507)
2	I	909666.0	429371.0	ACTIVE	23/10/2018	17:00	13.42507	99.955180	POINT (99.95518 13.42507)
3	I	909671.0	429372.0	ACTIVE	23/10/2018	19:00	13.42537	99.954059	POINT (99.954059 13.42537)
4	I	909672.0	429372.0	ACTIVE	23/10/2018	10:00	13.42537	99.954059	POINT (99.954059 13.42537)

Coding - Transform data (ต่อ)

- Step11 : นำข้อมูลที่มี Geometry column แล้วมาเข้า Geopandas library เพื่อใช้งานประมวลผลภาพแผนที่ โดยกำหนด Spatial References เป็น EPSG: 4326 (WGS84)

หมายเหตุ: SR ที่นิยมจะใช้ WGS84 (4326) และ Web Mercator (3857)

```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | [x] Variables ≡ Outline ...

# convert dataframe to geo dataframe
locations = geopandas.GeoDataFrame(df, geometry=df['geometry'].apply(shapely.wkt.loads),crs='epsg:4326')
[18] ✓ 0.4s

locations.shape
[19] ✓ 0.0s

... (38118, 8)
```

Coding - Transform data (ต่อ)

- Step12 : Load datasets polygon จังหวัดเพชรบุรี และนำมาสร้าง Polygon ด้วย Geopandas

```
+ Code + Markdown | ▶ Run All ↺ Restart ☒ Clear All Outputs | 📄 Variables ☰ Outline ...

# load geometry json จังหวัด เพชรบุรี
with open('./petchaburi.json','r',encoding='utf8') as f:
    province = json.load(f)
    f.close()

print(province)

[22] ✓ 0.0s

... {'type': 'FeatureCollection', 'features': [{'type': 'Feature', 'properties': {'OBJECTID': 1350, 'CHANGWAT_NAME': 'เพชรบุรี'}
```

```
+ Code + Markdown | ▶ Run All ↺ Restart ☒ Clear All Outputs | 📄 Variables ☰ Outline ...

# create polygon petchaburi
petchaburi = geopandas.GeoDataFrame.from_features(province['features'],crs=4326)

[23] ✓ 0.0s
```

Coding - Transform data (ต่อ)

- Step13 : ทำการ Spatial Join ระหว่าง set ของ locations และ polygon ด้วย การ Match แบบ Within และเป็นภายใน จะพบข้อมูลจำนวน 62 รายการ

```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | Variables Outline ...

[ ] point_in_polygon = geopandas.sjoin(locations,petchaburi,op='within',how='inner')

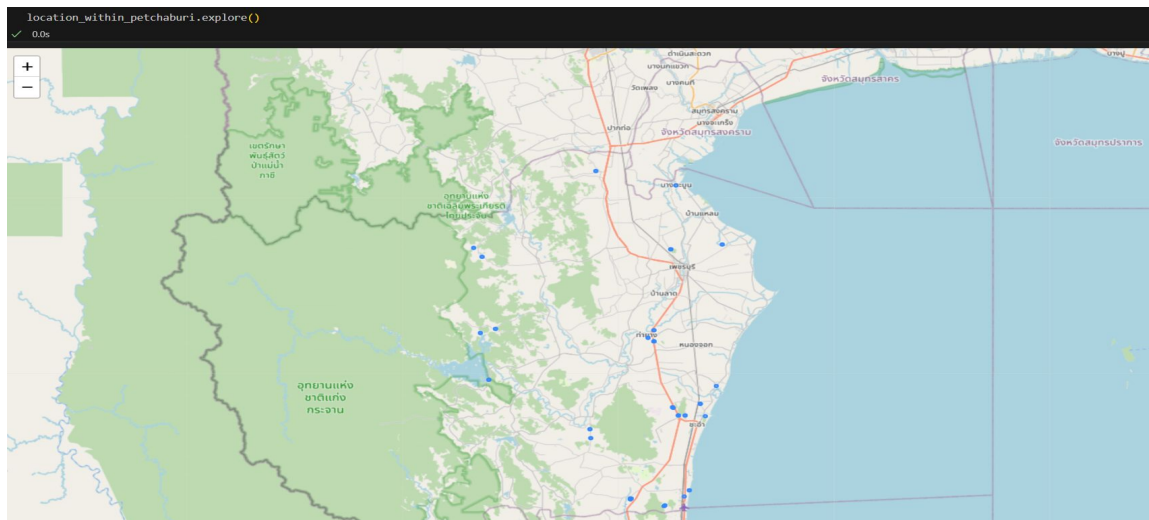
[25] ✓ 0.0s location_within_petchaburi = locations.loc[point_in_polygon.index]

[26] ✓ 0.0s location_within_petchaburi.shape

... (62, 8)
```

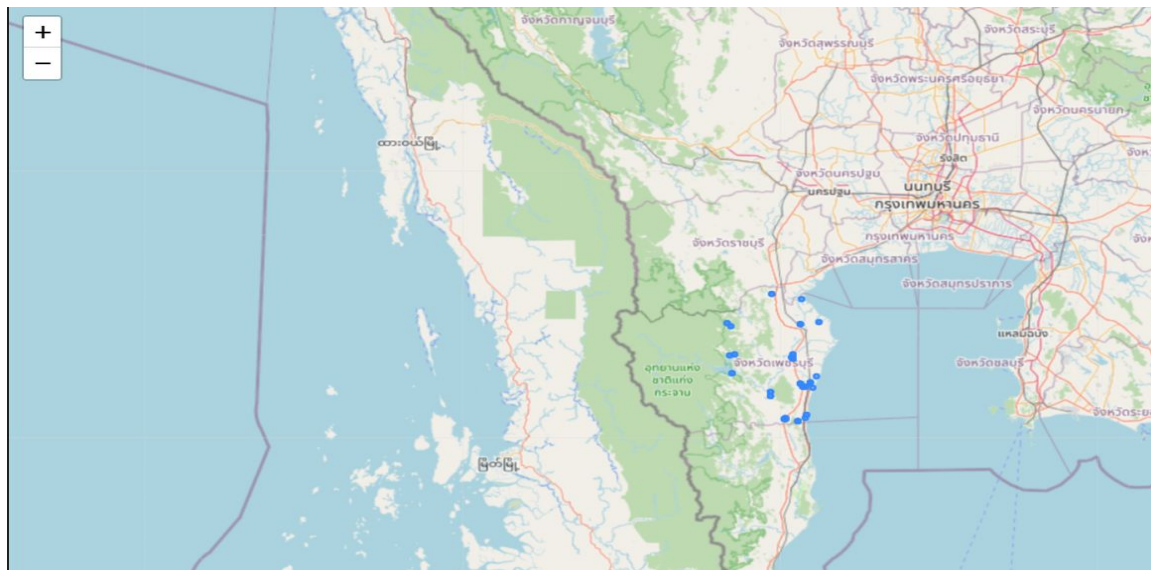
Coding - Transform data (ต่อ)

- Step14 : การ Plot ผลลัพธ์ของ Point ที่อยู่ในจังหวัดเพชรบุรี



Coding - Transform data (ต่อ)

- Step14 (ต่อ) : การ Plot พลาต์ของ Point ที่อยู่ในจังหวัดเพชรบุรี



Coding - Database structure

- ภาพตัวอย่างหน้าตาการเก็บข้อมูล Location

The screenshot displays the phpMyAdmin interface for a database named 'final'. The left sidebar shows a tree view of the database structure, including 'location', 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. The main panel shows the 'Table structure' view for the 'location' table. The table has four columns: 'ID' (bigint(20), primary key, auto-increment), 'CELL_STATUS' (varchar(255), utf8_general_ci), 'LATITUDE' (double(16,6)), and 'LONGITUDE' (double(16,6)). Below the table structure, there are options to check all, browse, change, drop, primary, unique, index, spatial, and fulltext. There are also links for print, propose table structure, track table, move columns, and normalize. At the bottom, there is an 'Indexes' section showing a primary index on the 'ID' column.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 ID	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 CELL_STATUS	varchar(255)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 LATITUDE	double(16,6)			No	None			Change Drop More
<input type="checkbox"/>	4 LONGITUDE	double(16,6)			No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	ID	45	A	No	

Coding - After inserts

- ภาพตัวอย่างข้อมูลภายหลังการนำเข้าข้อมูล

Server: 127.0.0.1 > Database: final > Table: location

Browse Structure SQL Search Insert Export Import Privi

✓ Showing rows 0 - 24 (62 total, Query took 0.0007 seconds.)

`SELECT * FROM `location``

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

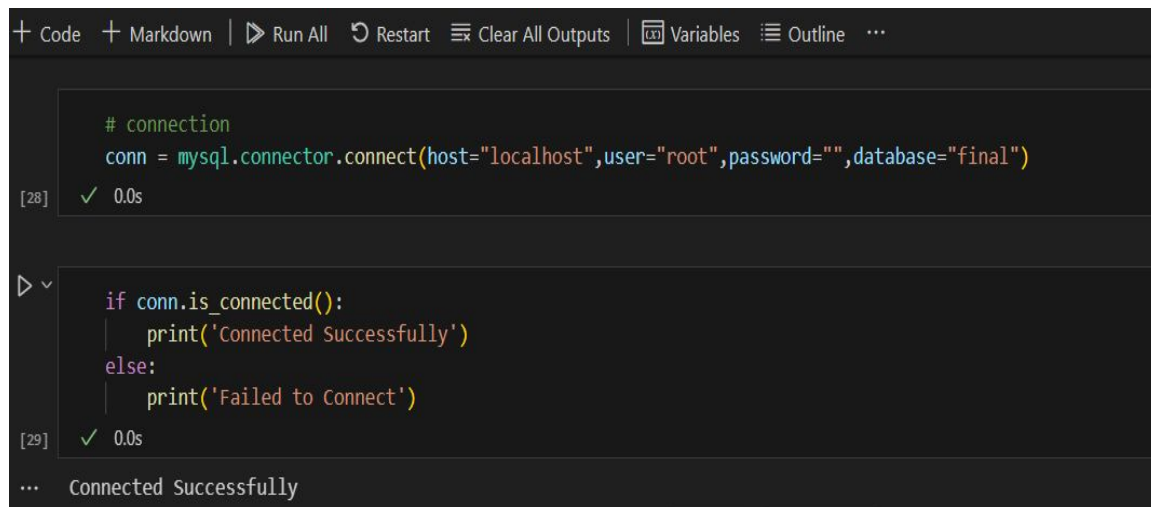
1 > >> ☐ Show all Number of rows: 25 Filter rows: Search this table

Extra options

				ID	CELL_STATUS	LATITUDE	LONGITUDE
<input type="checkbox"/>	Edit	Copy	Delete	1	ACTIVE	12.984736	99.643243
<input type="checkbox"/>	Edit	Copy	Delete	2	ACTIVE	12.984736	99.643243
<input type="checkbox"/>	Edit	Copy	Delete	3	ACTIVE	12.872900	100.002600
<input type="checkbox"/>	Edit	Copy	Delete	4	ACTIVE	12.976111	99.618670
<input type="checkbox"/>	Edit	Copy	Delete	5	ACTIVE	12.976111	99.618670
<input type="checkbox"/>	Edit	Copy	Delete	6	ACTIVE	13.264607	99.937636

Coding - Connect to database

- Step14 : ตรวจสอบการเชื่อมต่อฐานข้อมูล Connect to database



```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables 📄 Outline ...

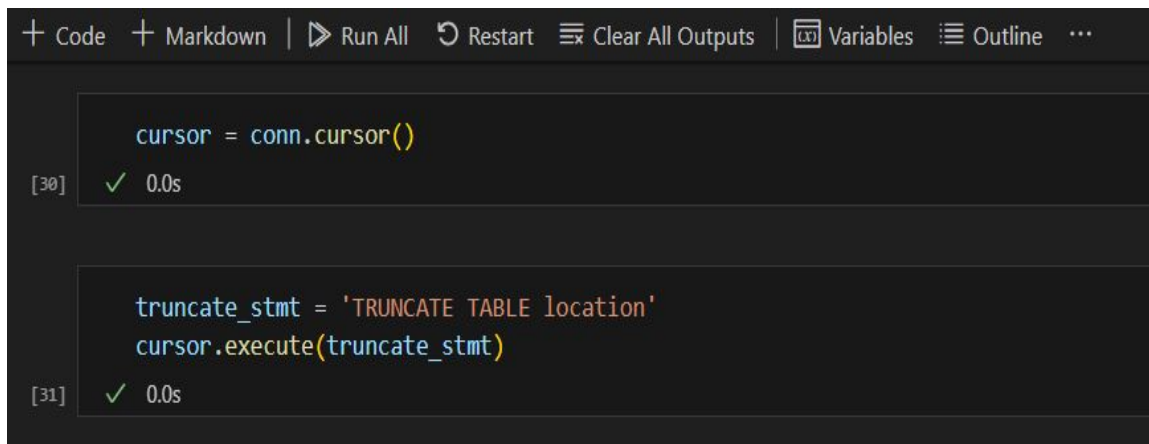
# connection
conn = mysql.connector.connect(host="localhost",user="root",password="",database="final")
[28] ✓ 0.0s

▶ ▾
if conn.is_connected():
    print('Connected Successfully')
else:
    print('Failed to Connect')
[29] ✓ 0.0s

... Connected Successfully
```


Coding - Truncate table

- Step15 : ทำการลบข้อมูลภายในตารางก่อนนำเข้าใหม่อีกครั้ง



```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ☰ Outline ...
```

```
[30] ✓ 0.0s
```

```
cursor = conn.cursor()
```

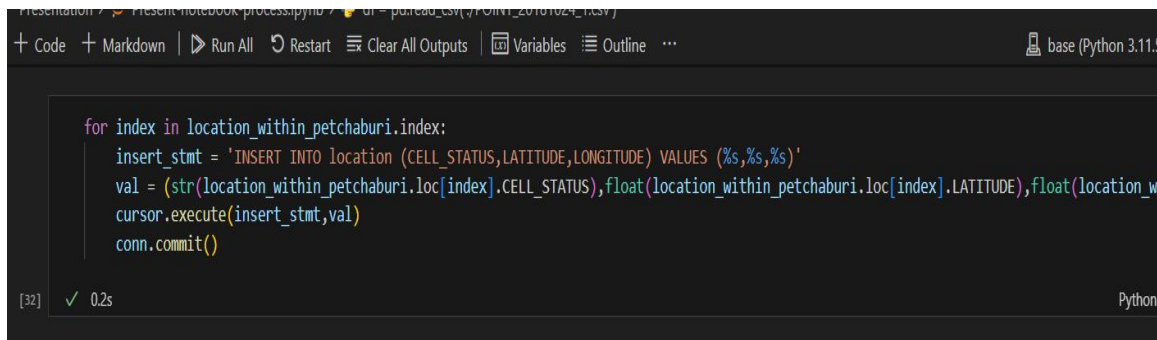
```
[31] ✓ 0.0s
```

```
truncate_stmt = 'TRUNCATE TABLE location'
```

```
cursor.execute(truncate_stmt)
```

Coding - Insert to database

- Step16 : วนลูปเพื่อทำการบันทึกข้อมูล



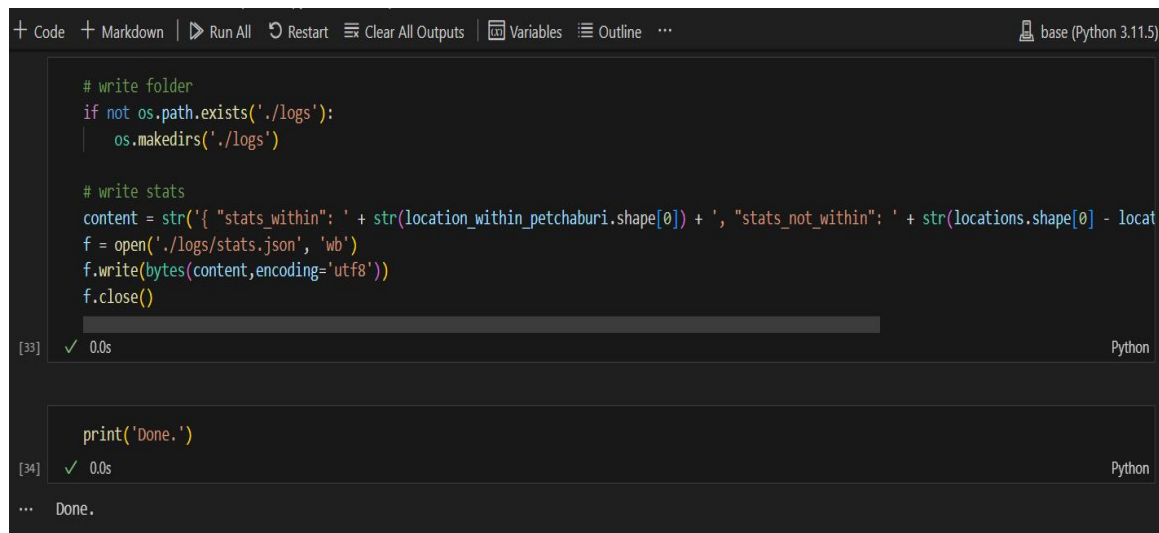
```
Presentation 7 - Present notebook process.ipynb 7 - df = pandas_csv('FOMIT_20161024_1.csv')
+ Code + Markdown | ▶ Run All | ↺ Restart | ☰ Clear All Outputs | 📄 Variables | 📖 Outline | ... | base (Python 3.11.5)

for index in location_within_petchaburi.index:
    insert_stmt = 'INSERT INTO location (CELL_STATUS,LATITUDE, LONGITUDE) VALUES (%s,%s,%s)'
    val = (str(location_within_petchaburi.loc[index].CELL_STATUS),float(location_within_petchaburi.loc[index].LATITUDE),float(location_w
    cursor.execute(insert_stmt,val)
    conn.commit()

[32] ✓ 0.2s Python
```

Coding - Save output to file

- Step17 : บันทึกข้อมูลลง Folder logs ได้แก่ค่า ที่อยู่ภายใน และค่าที่อยู่ภายนอก จังหวัดที่สนใจ



```
+ Code + Markdown | ▶ Run All ⏮ Restart ⌵ Clear All Outputs | 📄 Variables 📖 Outline ... base (Python 3.11.5)

# write folder
if not os.path.exists('./logs'):
    os.makedirs('./logs')

# write stats
content = str('{ "stats_within": ' + str(location_within_petchaburi.shape[0]) + ', "stats_not_within": ' + str(locations.shape[0] - locat
f = open('./logs/stats.json', 'wb')
f.write(bytes(content,encoding='utf8'))
f.close()

[33] ✓ 0.0s Python

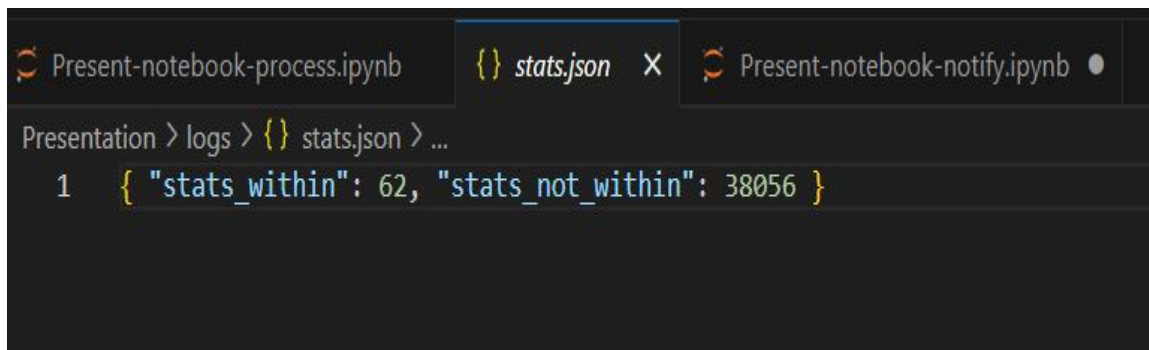
print('Done.')

[34] ✓ 0.0s Python

... Done.
```

Coding - Results

- ตัวอย่าง JSON ที่เป็นผลลัพธ์



The screenshot shows a Jupyter Notebook interface with three tabs at the top: 'Present-notebook-process.ipynb', 'stats.json', and 'Present-notebook-notify.ipynb'. The 'stats.json' tab is active. Below the tabs, the breadcrumb navigation reads 'Presentation > logs > stats.json > ...'. The main content area displays a single line of code, numbered '1', which is a JSON object: `{ "stats_within": 62, "stats_not_within": 38056 }`.

```
Present-notebook-process.ipynb  {} stats.json  X  Present-notebook-notify.ipynb ●
Presentation > logs > {} stats.json > ...
1  { "stats_within": 62, "stats_not_within": 38056 }
```

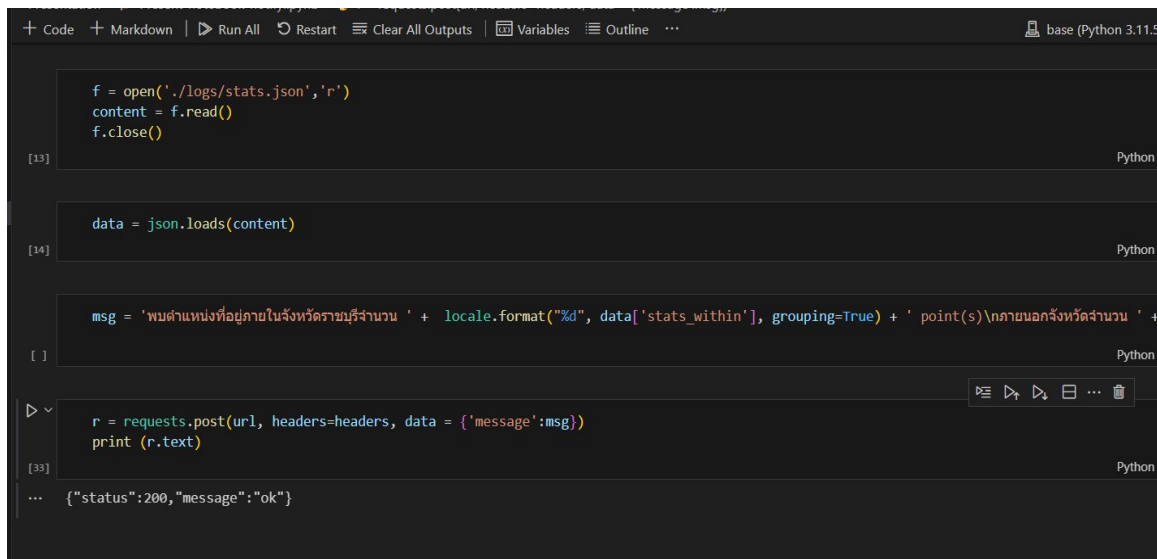
Coding - Line notify

- Step17 : กำหนด Token สำหรับ Line notify

```
+ Code + Markdown | ▶ Run All ↺ Restart ☰ Clear All Outputs | 📄 Variables 📄 Outline ...  
  
▶ ~  
import requests  
import json  
import locale  
locale.setlocale(locale.LC_ALL, 'en_US')  
[24]  
... 'en_US'  
  
url = 'https://notify-api.line.me/api/notify'  
[10]  
  
token = 'X3esjhIASjGdwkNRRIkTsGLa7076xFSYmzQI231JXTX'  
[11]  
  
headers = {'content-type': 'application/x-www-form-urlencoded', 'Authorization': 'Bearer '+token}  
[12]
```

Coding - Line notify (ต่อ)

- Step18 : Load result file เพื่ออ่านข้อมูลสำหรับป้อน Line message เพื่อส่งแจ้งเตือน



```
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables 📖 Outline ... base (Python 3.11.5)

f = open('./logs/stats.json','r')
content = f.read()
f.close()

[13] Python

data = json.loads(content)

[14] Python

msg = 'พบตำแหน่งที่อยู่ภายในจังหวัดราชบุรีจำนวน ' + locale.format("%d", data['stats_within'], grouping=True) + ' point(s)\nภายนอกจังหวัดจำนวน ' +

[ ] Python

r = requests.post(url, headers=headers, data = {'message':msg})
print (r.text)

[33] Python

... {"status":200,"message":"ok"}
```

Coding - Line notify (ต่อ)

- Step18 (ต่อ) : ภาพตัวอย่างการส่งแจ้งเตือนมายัง Line application



DataEngineerPSU:

พบตำแหน่งที่อยู่ในจังหวัดเพชรบุรีจำนวน 62 point(s)

ภายนอกจังหวัดจำนวน 38,056 point(s).

5:32 PM

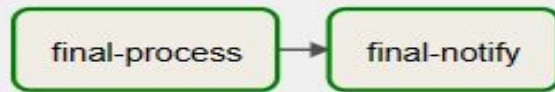
Coding - DAG

- Step19 : เขียนโปรแกรมสำหรับสร้าง Task บน Airflow

```
Present-notebook-process.ipynb • Present-notebook-notify.ipynb • final-etl-flow.py 3 X
final-project > dags > final-etl-flow.py > ...
1  from datetime import timedelta
2  from airflow import DAG
3  from airflow.operators.dummy_operator import DummyOperator
4  from airflow.operators.bash_operator import BashOperator
5  from airflow.operators.python_operator import PythonOperator
6  import logging
7
8  from airflow.utils import timezone
9
10 default_args = {
11     'owner': 'airflow',
12 }
13
14 with DAG (
15     'final-etl-flow',
16     schedule_interval='*/5 * * * *',
17     default_args=default_args,
18     start_date=timezone.datetime(2021, 7, 9),
19     dagrun_timeout=timedelta(minutes=5),
20     tags = ['ETL','Notify'],
21     catchup=False,
22 ) as dag :
23     process_every_daily_task = BashOperator(
24         task_id='final-process',
25         bash_command='python /python_scripts/final-process.py',
26         dag=dag)
27
28     notify_every_daily_task = BashOperator(
29         task_id='final-notify',
30         bash_command='python /python_scripts/final-notify.py',
31         dag=dag)
32     process_every_daily_task >> notify_every_daily_task
```


ETL Flow

- ຮູບກາງ Flow ທີ່ຖືກ Generate ບຸ Airflow



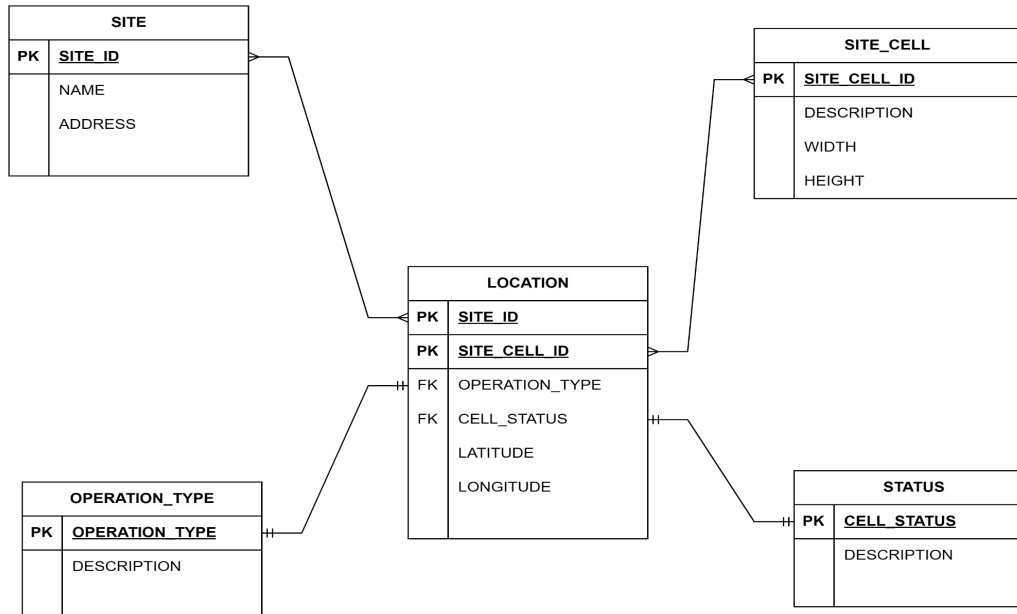
Airflow logs

- ภาพตัวอย่างการ Logs ข้อมูล

List Log							
Search							
<div><div>< 1 2 3 4 5 6 7 ></div><div>Page size</div><div></div></div>							Record Count: 696
Id	Dttm	Dag Id	Task Id	Event	Execution Date	Owner	Extra
696	2023-12-14, 16:24:22	final-etl-flow		tree		airflow	[{"dag_id", "final-etl-flow"}]
695	2023-12-14, 16:24:22	final-etl-flow		dagrun_success	2023-12-14, 15:43:42	airflow	[{"confirmed", true}, {"dag_id", "final-etl-flow"}, {"execution_date", "2023-12-14T15:43:42.765509+00:00"}, {"origin", "http://localhost:8080/tree?dag_id=final-etl-flow"}]
694	2023-12-14, 16:24:20	final-etl-flow		dagrun_success	2023-12-14, 15:43:42	airflow	[{"dag_id", "final-etl-flow"}, {"execution_date", "2023-12-14T15:43:42.765509+00:00"}, {"origin", "http://localhost:8080/tree?dag_id=final-etl-flow"}]
693	2023-12-14, 16:24:15	final-etl-flow		tree		airflow	[{"dag_id", "final-etl-flow"}]

Normalization

- การทำ Normalization จำลองจาก Datasets ที่โหลดครั้งแรก สามารถพิจารณาแยกความสัมพันธ์ได้ดังภาพ ซึ่งทุกตารางจะอยู่ในรูปแบบ 2NF แล้วส่วนตาราง Location จะเป็นแบบ 3NF



Q/A

Thank you