Database Management System Final Project Report

0711257 葉長瀚

Motivation

For every devoted movie fan including me, it will be very handy if we can know the latest available showings in the specific area of the movie we want to watch. In fact, there are already some webs providing these kind of service, and by this final project, I want to implement this kind of function by myself, trying to figure out how they actually work.

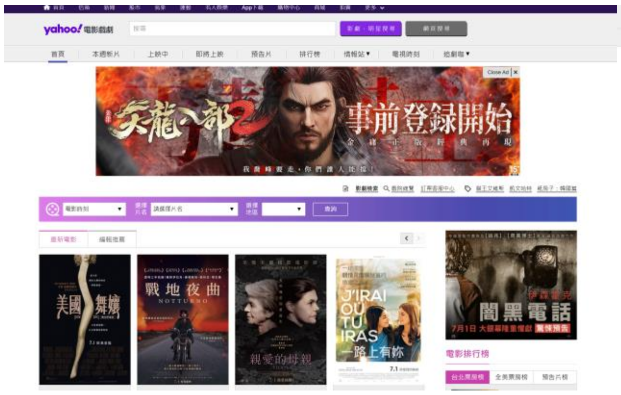
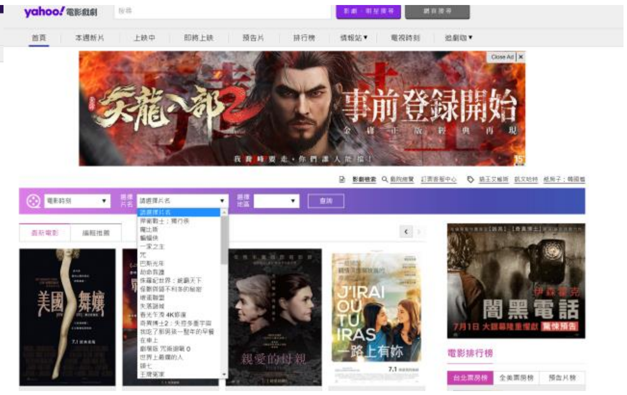
Application description

The program will crawl all the premiere movies and the showings of every movie in every theater, and store them in the database to provide a query service that user can select the movie he wants to watch and the city he wants to watch at, and then the query system will show all available showings of the designated movie in the order of the approach of time.

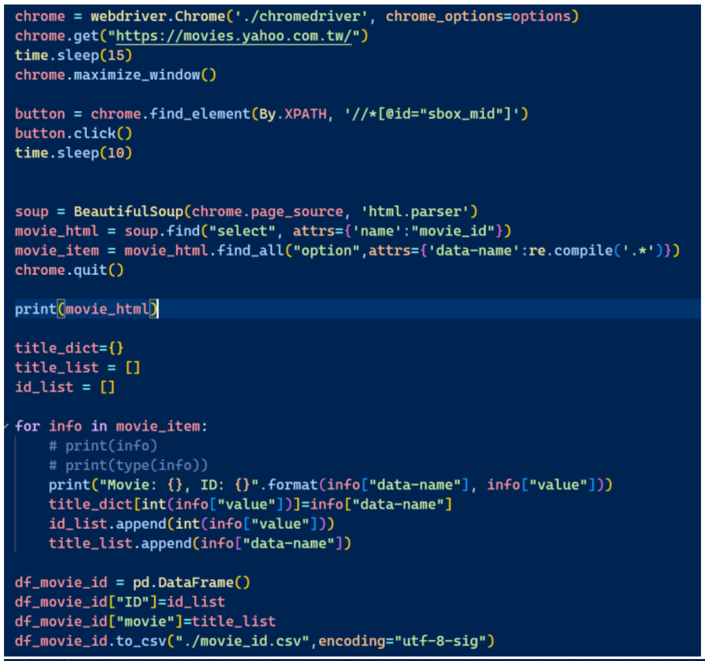
Data sources and how you collect and import the data

I write a program to crawl the premiere movies and all of their showings in every theater from *https://movies.yahoo.com.tw/*, and import them into the database. They are all in one program, which means we can collect and import data by merely run it without any further manual setting. The detail of every part will be introduced in the following.

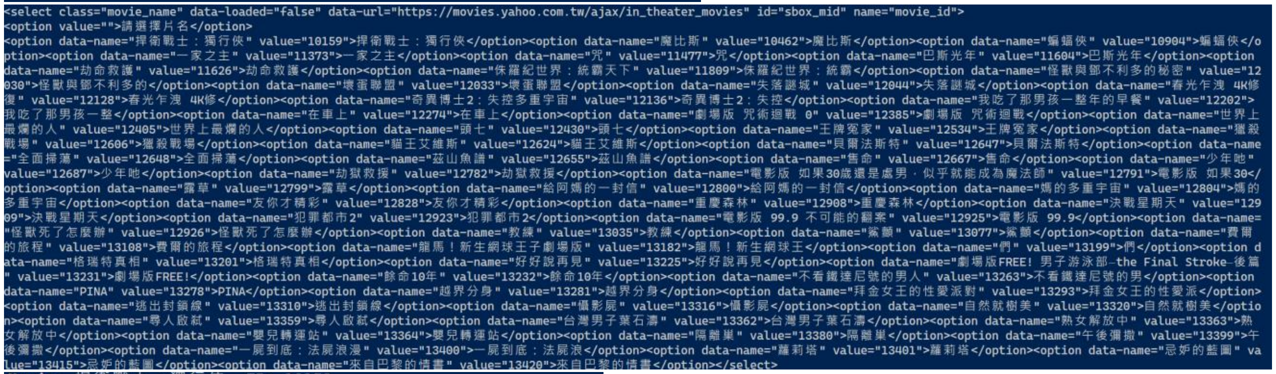
Crawling premiere movies

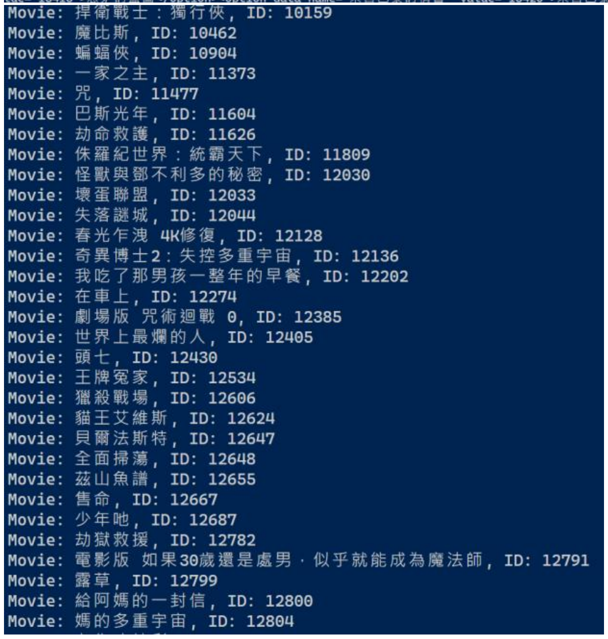
We can crawl all the premiere movies and corresponding movie IDs from the *https://movies.yahoo.com.tw/* website, but here is a tricky part, if we didn’t click that button shown above before crawling, we aren’t able to crawl any of movies, so to make it works, I use webdriver to simulate the click event and find the button position by Xpath. As the figure shown below, I can finally crawl all the premiere movies and IDs and store them as a dictionary.



```python
chrome = webdriver.Chrome('./chromedriver', chrome_options=options)
chrome.get("https://movies.yahoo.com.tw/")
time.sleep(15)
chrome.maximize_window()

button = chrome.find_element(By.XPATH, '//*[@id="sbox_mid"]')
button.click()
time.sleep(10)


soup = BeautifulSoup(chrome.page_source, 'html.parser')
movie_html = soup.find("select", attrs={'name':"movie_id"})
movie_item = movie_html.find_all("option",attrs={'data-name':re.compile('.*')})
chrome.quit()

print(movie_html)

title_dict={}
title_list = []
id_list = []

for info in movie_item:
    # print(info)
    # print(type(info))
    print("Movie: {}, ID: {}".format(info["data-name"], info["value"]))
    title_dict[int(info["value"])]=info["data-name"]
    id_list.append(int(info["value"]))
    title_list.append(info["data-name"])

df_movie_id = pd.DataFrame()
df_movie_id["ID"]=id_list
df_movie_id["movie"]=title_list
df_movie_id.to_csv("./movie_id.csv",encoding="utf-8-sig")
```

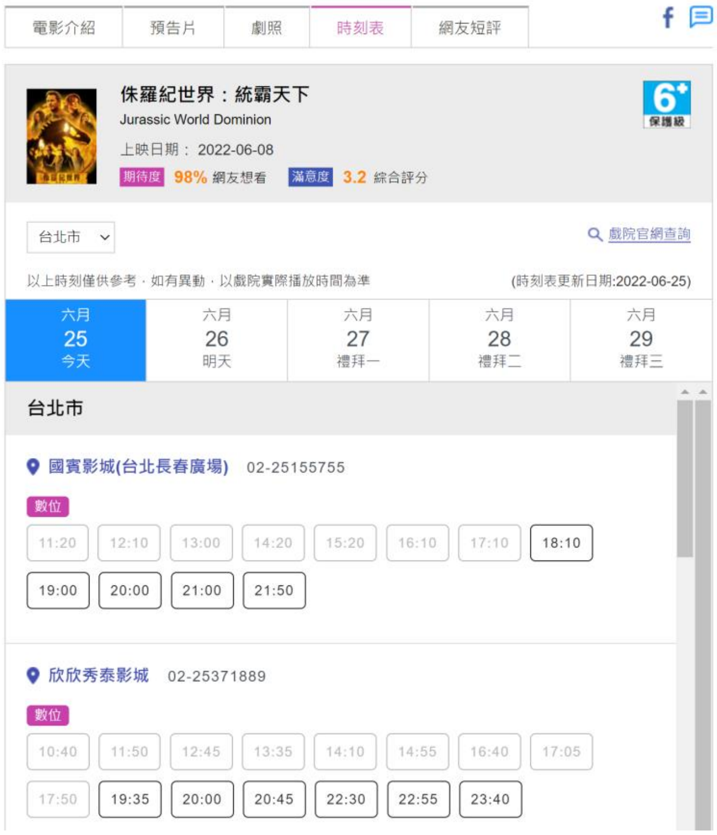


```
<select class="movie_name" data-loaded="false" data-url="https://movies.yahoo.com.tw/ajax/in_theater_movies" id="sbox_mid" name="movie_id">
<option value="">請選擇片名</option>
<option data-name="捍衛戰士：獨行俠" value="10159">捍衛戰士：獨行俠</option><option data-name="魔比斯" value="10462">魔比斯</option><option data-name="蝙蝠俠" value="10904">蝙蝠俠</o
ption><option data-name="一家之主" value="11373">一家之主</option><option data-name="咒" value="11477">咒</option><option data-name="巴斯光年" value="11604">巴斯光年</option><option
data-name="劫命救護" value="11626">劫命救護</option><option data-name="侏羅紀世界：統霸天下" value="11809">侏羅紀世界：統霸</option><option data-name="怪獸與鄧不利多的秘密" value="12
030">怪獸與鄧不利多的</option><option data-name="壞蛋聯盟" value="12033">壞蛋聯盟</option><option data-name="失落謎城" value="12044">失落謎城</option><option data-name="春光乍洩 4K修
復" value="12128">春光乍洩 4K修</option><option data-name="奇異博士2：失控多重宇宙" value="12136">奇異博士2：失控</option><option data-name="我吃了那男孩一整年的早餐" value="12202">
我吃了那男孩一整</option><option data-name="在車上" value="12274">在車上</option><option data-name="劇場版 咒術迴戰 0" value="12385">劇場版 咒術迴戰</option><option data-name="世界上
最爛的人" value="12405">世界上最爛的人</option><option data-name="頭七" value="12430">頭七</option><option data-name="王牌冤家" value="12534">王牌冤家</option><option data-name="獵殺
戰場" value="12606">獵殺戰場</option><option data-name="貓王艾維斯" value="12624">貓王艾維斯</option><option data-name="貝爾法斯特" value="12647">貝爾法斯特</option><option data-name
="全面掃蕩" value="12648">全面掃蕩</option><option data-name="茲山魚譜" value="12655">茲山魚譜</option><option data-name="售命" value="12667">售命</option><option data-name="少年吔"
value="12687">少年吔</option><option data-name="劫獄救援" value="12782">劫獄救援</option><option data-name="電影版 如果30歲還是處男，似乎就能成為魔法師" value="12791">電影版 如果30</
option><option data-name="露草" value="12799">露草</option><option data-name="給阿媽的一封信" value="12800">給阿媽的一封信</option><option data-name="媽的多重宇宙" value="12804">媽的
多重宇宙</option><option data-name="友你才精彩" value="12828">友你才精彩</option><option data-name="重慶森林" value="12908">重慶森林</option><option data-name="決戰星期天" value="129
09">決戰星期天</option><option data-name="犯罪都市2" value="12923">犯罪都市2</option><option data-name="電影版 99.9 不可能的翻案" value="12925">電影版 99.9</option><option data-name=
"怪獸死了怎麼辦" value="12926">怪獸死了怎麼辦</option><option data-name="教練" value="13035">教練</option><option data-name="鯊顫" value="13077">鯊顫</option><option data-name="費爾
的旅程" value="13108">費爾的旅程</option><option data-name="龍馬！新生網球王子劇場版" value="13182">龍馬！新生網球王</option><option data-name="們" value="13199">們</option><option d
ata-name="格瑞特真相" value="13201">格瑞特真相</option><option data-name="好好說再見" value="13225">好好說再見</option><option data-name="劇場版FREE! 男子游泳部–the Final Stroke–後篇
" value="13231">劇場版FREE!</option><option data-name="餘命10年" value="13232">餘命10年</option><option data-name="不看鐵達尼號的男人" value="13263">不看鐵達尼號的男</option><option
data-name="PINA" value="13278">PINA</option><option data-name="越界分身" value="13281">越界分身</option><option data-name="拜金女王的性愛派對" value="13293">拜金女王的性愛派</option>
<option data-name="逃出封鎖線" value="13310">逃出封鎖線</option><option data-name="儡影屍" value="13316">儡影屍</option><option data-name="自然就樹美" value="13320">自然就樹美</optio
n><option data-name="尋人啟弒" value="13359">尋人啟弒</option><option data-name="台灣男子葉石濤" value="13362">台灣男子葉石濤</option><option data-name="熟女解放中" value="13363">熟
女解放中</option><option data-name="嬰兒轉運站" value="13364">嬰兒轉運站</option><option data-name="隔離巢" value="13380">隔離巢</option><option data-name="午後彌撒" value="13399">午
後彌撒</option><option data-name="一屍到底：法屍浪漫" value="13400">一屍到底：法屍浪</option><option data-name="蘿莉塔" value="13401">蘿莉塔</option><option data-name="忌妒的藍圖" va
lue="13415">忌妒的藍圖</option><option data-name="來自巴黎的情書" value="13420">來自巴黎的情書</option></select>
```



```
Movie: 捍衛戰士：獨行俠, ID: 10159
Movie: 魔比斯, ID: 10462
Movie: 蝙蝠俠, ID: 10904
Movie: 一家之主, ID: 11373
Movie: 咒, ID: 11477
Movie: 巴斯光年, ID: 11604
Movie: 劫命救護, ID: 11626
Movie: 侏羅紀世界：統霸天下, ID: 11809
Movie: 怪獸與鄧不利多的秘密, ID: 12030
Movie: 壞蛋聯盟, ID: 12033
Movie: 失落謎城, ID: 12044
Movie: 春光乍洩 4K修復, ID: 12128
Movie: 奇異博士2：失控多重宇宙, ID: 12136
Movie: 我吃了那男孩一整年的早餐, ID: 12202
Movie: 在車上, ID: 12274
Movie: 劇場版 咒術迴戰 0, ID: 12385
Movie: 世界上最爛的人, ID: 12405
Movie: 頭七, ID: 12430
Movie: 王牌冤家, ID: 12534
Movie: 獵殺戰場, ID: 12606
Movie: 貓王艾維斯, ID: 12624
Movie: 貝爾法斯特, ID: 12647
Movie: 全面掃蕩, ID: 12648
Movie: 茲山魚譜, ID: 12655
Movie: 售命, ID: 12667
Movie: 少年吔, ID: 12687
Movie: 劫獄救援, ID: 12782
Movie: 電影版 如果30歲還是處男，似乎就能成為魔法師, ID: 12791
Movie: 露草, ID: 12799
Movie: 給阿媽的一封信, ID: 12800
Movie: 媽的多重宇宙, ID: 12804
```

Crawling movie showings



電影介紹 預告片 劇照 時刻表 網友短評

侏羅紀世界：統霸天下
Jurassic World Dominion
上映日期： 2022-06-08
期待度 98% 網友想看 滿意度 3.2 綜合評分

台北市

戲院官網查詢

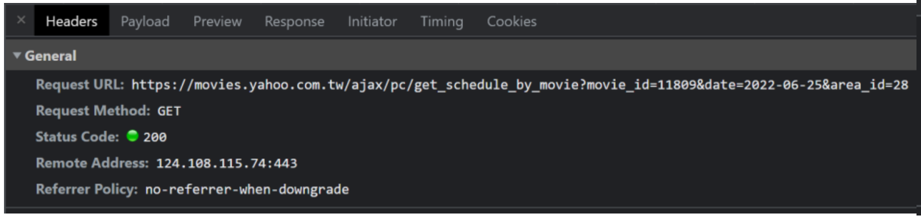
以上時刻僅供參考，如有異動，以戲院實際播放時間為準 (時刻表更新日期:2022-06-25)

六月 25 今天 | 六月 26 明天 | 六月 27 禮拜一 | 六月 28 禮拜二 | 六月 29 禮拜三

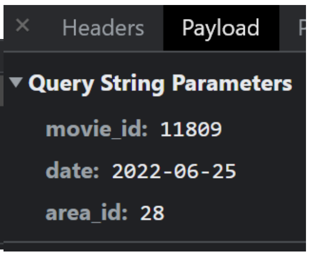
台北市

國賓影城(台北長春廣場) 02-25155755
數位
11:20 12:10 13:00 14:20 15:20 16:10 17:10 18:10
19:00 20:00 21:00 21:50

欣欣秀泰影城 02-25371889
數位
10:40 11:50 12:45 13:35 14:10 14:55 16:40 17:05
17:50 19:35 20:00 20:45 22:30 22:55 23:40

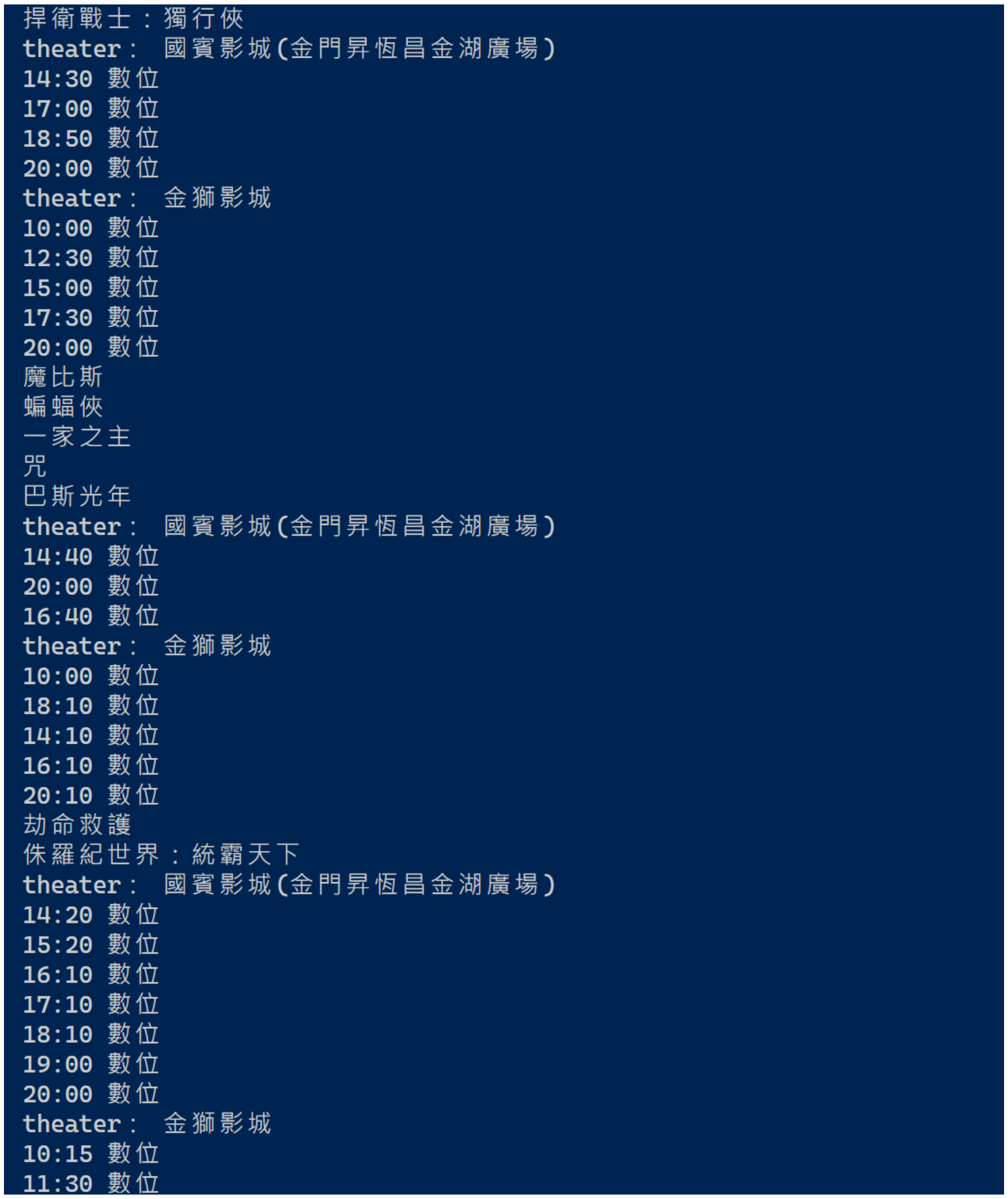


```
▼ General
  Request URL: https://movies.yahoo.com.tw/ajax/pc/get_schedule_by_movie?movie_id=11809&date=2022-06-25&area_id=28
  Request Method: GET
  Status Code: ● 200
  Remote Address: 124.108.115.74:443
  Referrer Policy: no-referrer-when-downgrade
```

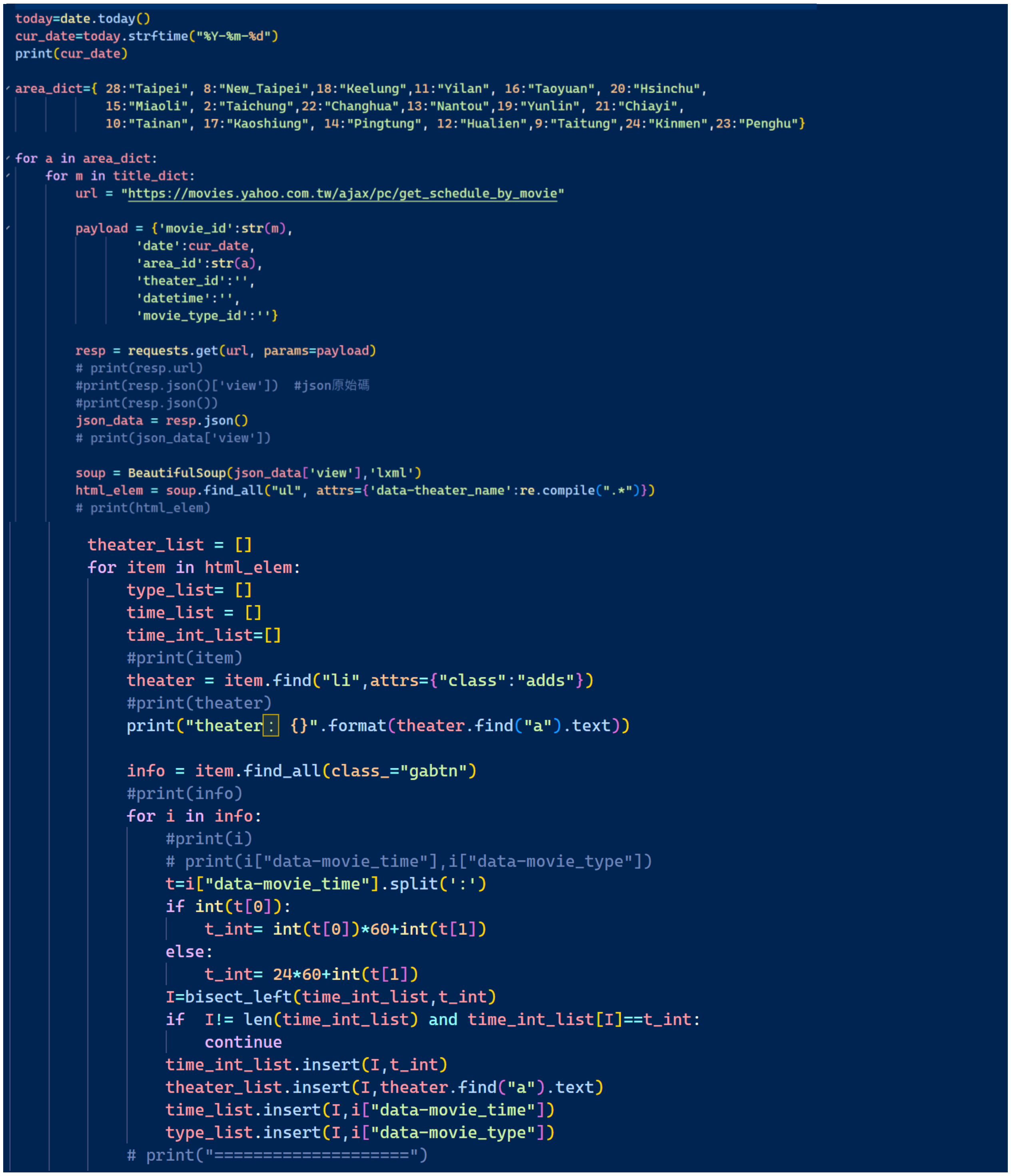


```
▼ Query String Parameters
  movie_id: 11809
  date: 2022-06-25
  area_id: 28
```

From the developer tools we know that the service of querying movie showings is based on an API, and we have to make request with designated movie ID, date and area ID as payload. Premiere movie IDs are gotten already. I input an area_id dictionary manually, and now, with the current date, I am able to crawl all the showings in every theater in all of the cities of every movie. The result is shown below.



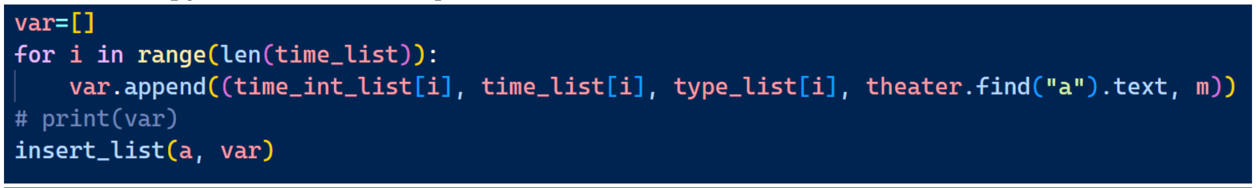
```
捍衛戰士：獨行俠
theater： 國賓影城(金門昇恆昌金湖廣場)
14:30 數位
17:00 數位
18:50 數位
20:00 數位
theater： 金獅影城
10:00 數位
12:30 數位
15:00 數位
17:30 數位
20:00 數位
魔比斯
蝙蝠俠
一家之主
咒
巴斯光年
theater： 國賓影城(金門昇恆昌金湖廣場)
14:40 數位
20:00 數位
16:40 數位
theater： 金獅影城
10:00 數位
18:10 數位
14:10 數位
16:10 數位
20:10 數位
劫命救護
侏羅紀世界：統霸天下
theater： 國賓影城(金門昇恆昌金湖廣場)
14:20 數位
15:20 數位
16:10 數位
17:10 數位
18:10 數位
19:00 數位
20:00 數位
theater： 金獅影城
10:15 數位
11:30 數位
```



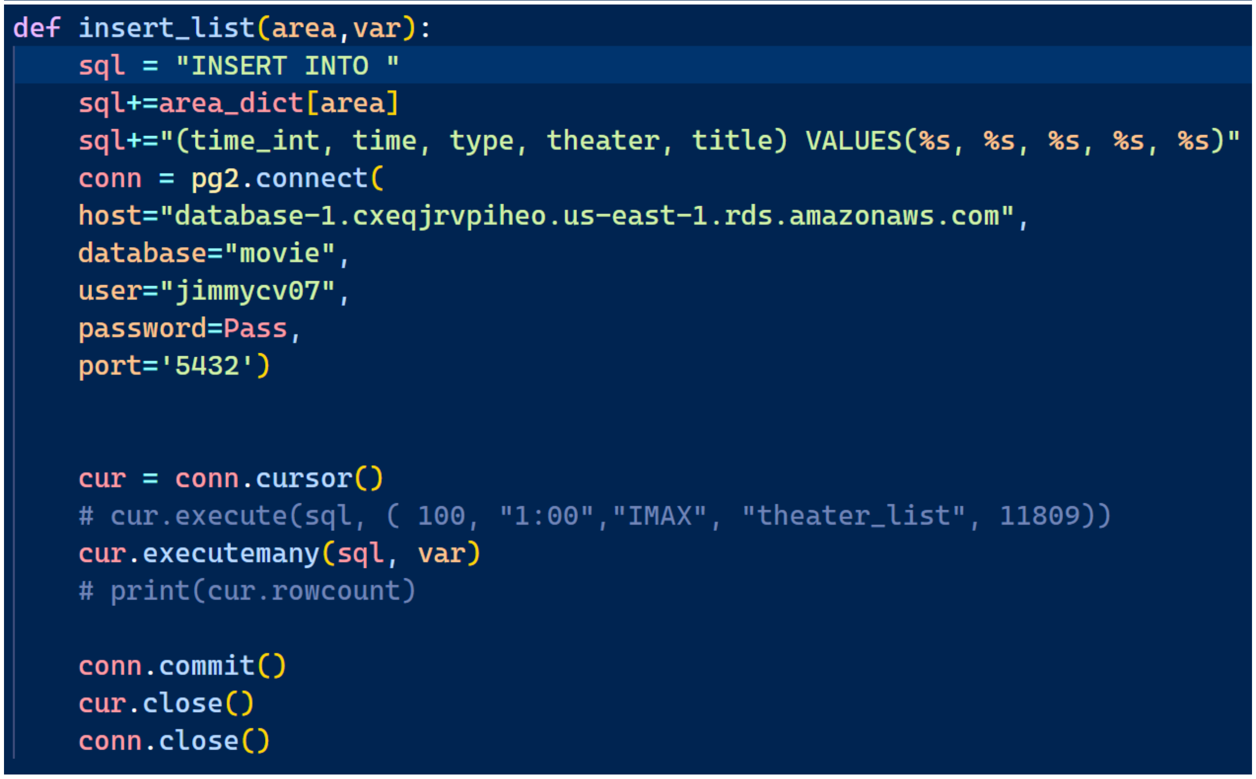
```
today=date.today()
cur_date=today.strftime("%Y-%m-%d")
print(cur_date)

area_dict={ 28:"Taipei", 8:"New_Taipei",18:"Keelung",11:"Yilan", 16:"Taoyuan", 20:"Hsinchu",
            15:"Miaoli", 2:"Taichung",22:"Changhua",13:"Nantou",19:"Yunlin", 21:"Chiayi",
            10:"Tainan", 17:"Kaoshiung", 14:"Pingtung", 12:"Hualien",9:"Taitung",24:"Kinmen",23:"Penghu"}

for a in area_dict:
    for m in title_dict:
        url = "https://movies.yahoo.com.tw/ajax/pc/get_schedule_by_movie"

        payload = {'movie_id':str(m),
                'date':cur_date,
                'area_id':str(a),
                'theater_id':'',
                'datetime':'',
                'movie_type_id':''}

        resp = requests.get(url, params=payload)
        # print(resp.url)
        #print(resp.json()['view'])  #json原始碼
        #print(resp.json())
        json_data = resp.json()
        # print(json_data['view'])

        soup = BeautifulSoup(json_data['view'],'lxml')
        html_elem = soup.find_all("ul", attrs={'data-theater_name':re.compile(".*")})
        # print(html_elem)

        theater_list = []
        for item in html_elem:
            type_list= []
            time_list = []
            time_int_list=[]
            #print(item)
            theater = item.find("li",attrs={"class":"adds"})
            #print(theater)
            print("theater: {}".format(theater.find("a").text))

            info = item.find_all(class_="gabtn")
            #print(info)
            for i in info:
                #print(i)
                # print(i["data-movie_time"],i["data-movie_type"])
                t=i["data-movie_time"].split(':')
                if int(t[0]):
                    t_int= int(t[0])*60+int(t[1])
                else:
                    t_int= 24*60+int(t[1])
                I=bisect_left(time_int_list,t_int)
                if  I!= len(time_int_list) and time_int_list[I]==t_int:
                    continue
                time_int_list.insert(I,t_int)
                theater_list.insert(I,theater.find("a").text)
                time_list.insert(I,i["data-movie_time"])
                type_list.insert(I,i["data-movie_type"])
            # print("====================")
```

Importing data

To import crawled data into database, I wrote a function utilizing library *psycopg* to attain it, which is a very convenient function enabling us to connect to the database and do some sql command in python. The insert sql command I used is shown below.



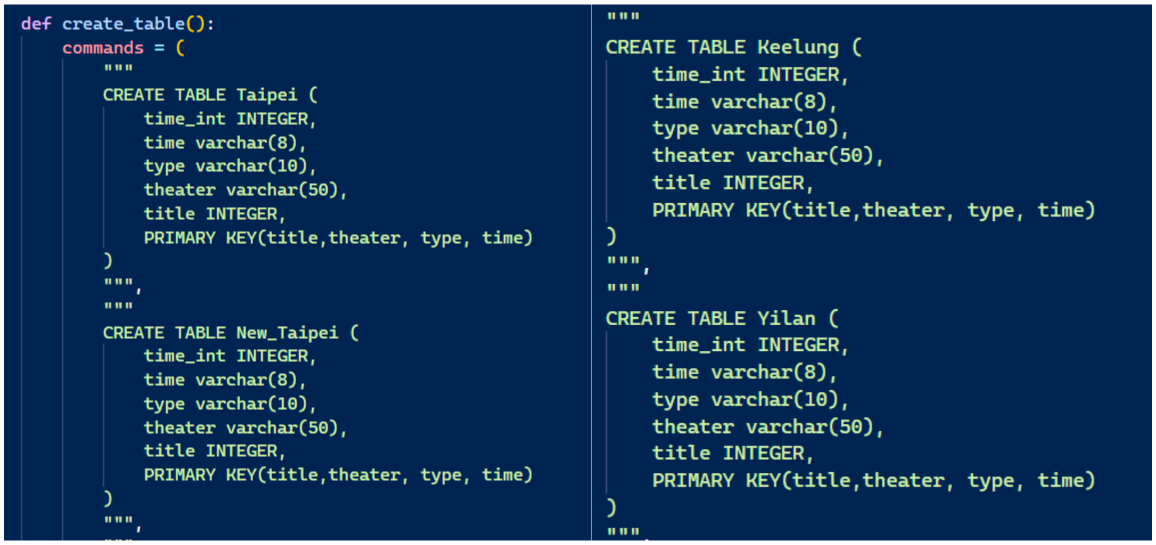
```python
var=[]
for i in range(len(time_list)):
    var.append((time_int_list[i], time_list[i], type_list[i], theater.find("a").text, m))
# print(var)
insert_list(a, var)
```



```python
def insert_list(area,var):
    sql = "INSERT INTO "
    sql+=area_dict[area]
    sql+="(time_int, time, type, theater, title) VALUES(%s, %s, %s, %s, %s)"
    conn = pg2.connect(
    host="database-1.cxeqjrvpiheo.us-east-1.rds.amazonaws.com",
    database="movie",
    user="jimmycv07",
    password=Pass,
    port='5432')


    cur = conn.cursor()
    # cur.execute(sql, ( 100, "1:00","IMAX", "theater_list", 11809))
    cur.executemany(sql, var)
    # print(cur.rowcount)

    conn.commit()
    cur.close()
    conn.close()
```

Database schema

I create a table for each of the area to avoid the redundancy and also make query more handily. Each table has the same schema, which owns the theater, title (11809 which is movie id), type (IMAX), showings(22:00) and time_int(1320 which is transferred from showing to integer) attributes.

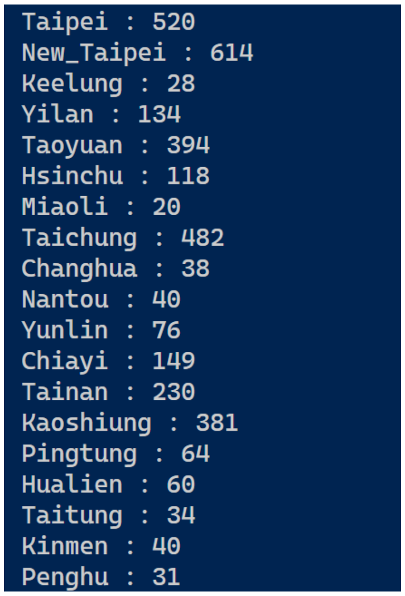


```
def create_table():
    commands = (
        """
        CREATE TABLE Taipei (
            time_int INTEGER,
            time varchar(8),
            type varchar(10),
            theater varchar(50),
            title INTEGER,
            PRIMARY KEY(title,theater, type, time)
        )
        """,
        """
        CREATE TABLE New_Taipei (
            time_int INTEGER,
            time varchar(8),
            type varchar(10),
            theater varchar(50),
            title INTEGER,
            PRIMARY KEY(title,theater, type, time)
        )
        """,
```

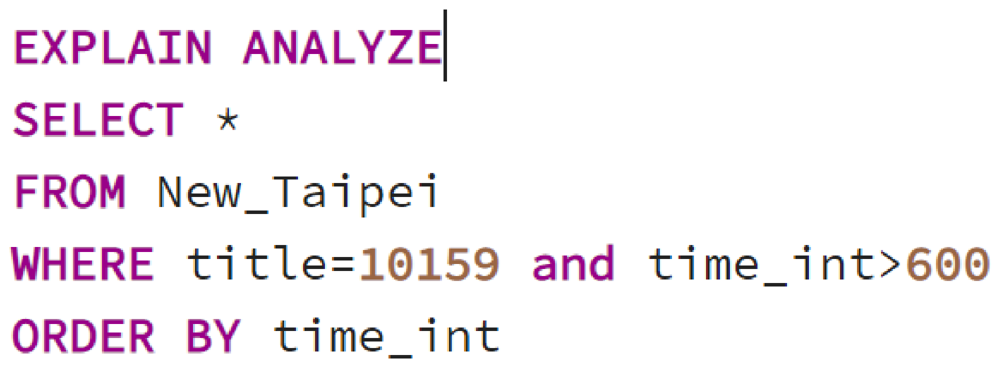
```
"""
CREATE TABLE Keelung (
    time_int INTEGER,
    time varchar(8),
    type varchar(10),
    theater varchar(50),
    title INTEGER,
    PRIMARY KEY(title,theater, type, time)
)
""",
"""
CREATE TABLE Yilan (
    time_int INTEGER,
    time varchar(8),
    type varchar(10),
    theater varchar(50),
    title INTEGER,
    PRIMARY KEY(title,theater, type, time)
)
```

Index

Below shows the amount of data in every area of a day. I think that index should work well in the table with larger amount of data, so I tried adding index in table *Taipei* and *New_Taipei*.



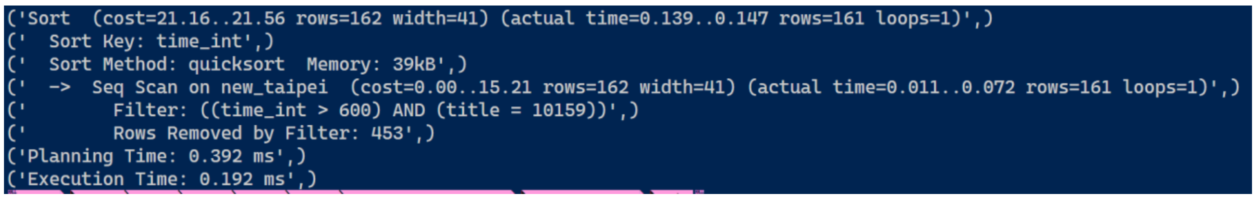
```
Taipei : 520
New_Taipei : 614
Keelung : 28
Yilan : 134
Taoyuan : 394
Hsinchu : 118
Miaoli : 20
Taichung : 482
Changhua : 38
Nantou : 40
Yunlin : 76
Chiayi : 149
Tainan : 230
Kaoshiung : 381
Pingtung : 64
Hualien : 60
Taitung : 34
Kinmen : 40
Penghu : 31
```



```
EXPLAIN ANALYZE
SELECT *
FROM New_Taipei
WHERE title=10159 and time_int>600
ORDER BY time_int
```

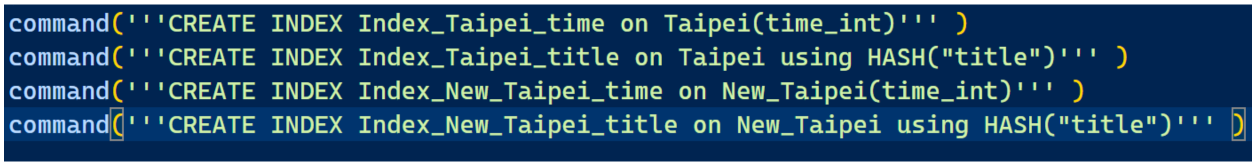
To see the difference between index and no index, I test it by doing above sql command which is the main query used in the application of this project before and after index was created. At the where clause, we can see that the query try to find out the data of specific movie(title attribute) and showings(time_int attribute) that are later than a specific time, so I came up with an idea that creating a hash index on *title* attribute and a B tree index on *time_int* attribute.

Here is the result of EXPALIN ANALYZE before adding index



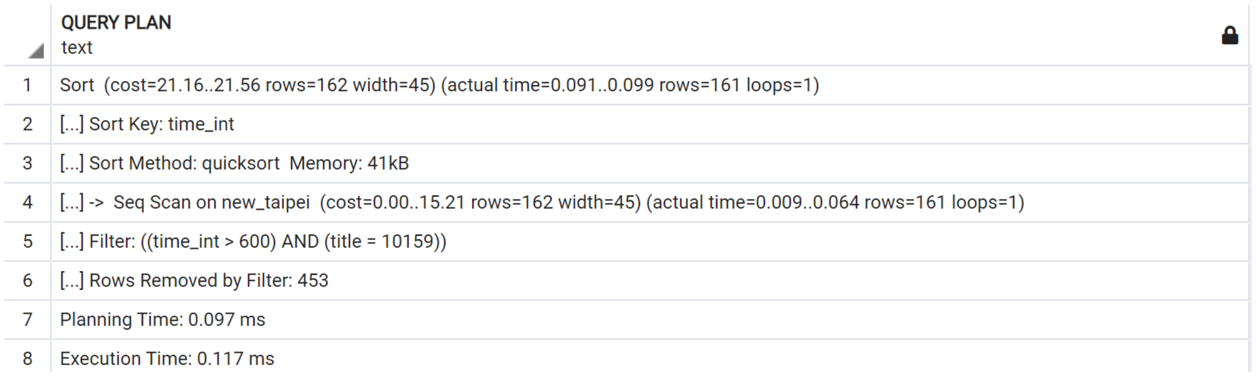
```
('Sort  (cost=21.16..21.56 rows=162 width=41) (actual time=0.139..0.147 rows=161 loops=1)',)
('  Sort Key: time_int',)
('  Sort Method: quicksort  Memory: 39kB',)
('  ->  Seq Scan on new_taipei  (cost=0.00..15.21 rows=162 width=41) (actual time=0.011..0.072 rows=161 loops=1)',)
('        Filter: ((time_int > 600) AND (title = 10159))',)
('        Rows Removed by Filter: 453',)
('Planning Time: 0.392 ms',)
('Execution Time: 0.192 ms',)
```

Adding index



```
command('''CREATE INDEX Index_Taipei_time on Taipei(time_int)''' )
command('''CREATE INDEX Index_Taipei_title on Taipei using HASH("title")''' )
command('''CREATE INDEX Index_New_Taipei_time on New_Taipei(time_int)''' )
command('''CREATE INDEX Index_New_Taipei_title on New_Taipei using HASH("title")''' )
```

However, I got the nearly same result after adding index with the optimizer still choose the sequential scan

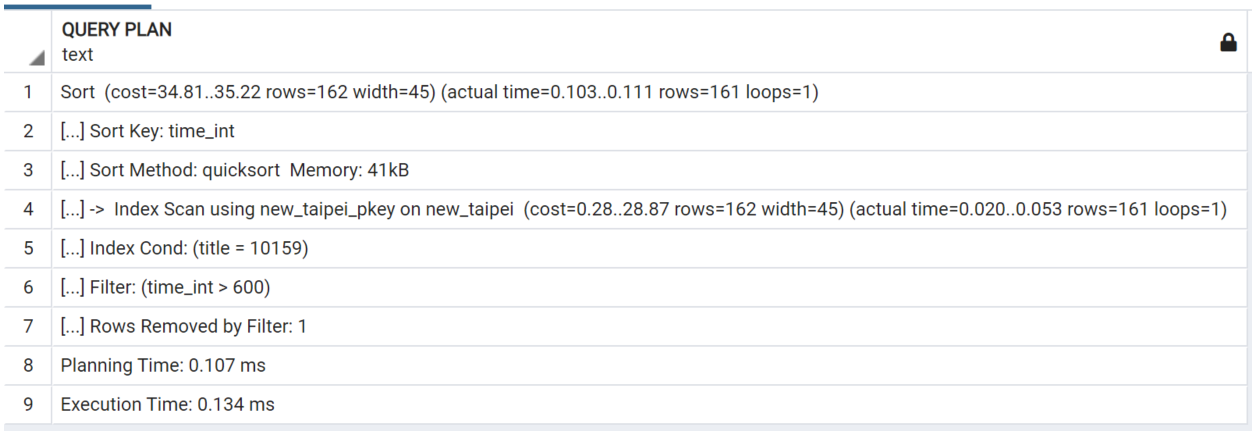


	QUERY PLAN text
1	Sort (cost=21.16..21.56 rows=162 width=45) (actual time=0.091..0.099 rows=161 loops=1)
2	[...] Sort Key: time_int
3	[...] Sort Method: quicksort Memory: 41kB
4	[...] -> Seq Scan on new_taipei (cost=0.00..15.21 rows=162 width=45) (actual time=0.009..0.064 rows=161 loops=1)
5	[...] Filter: ((time_int > 600) AND (title = 10159))
6	[...] Rows Removed by Filter: 453
7	Planning Time: 0.097 ms
8	Execution Time: 0.117 ms

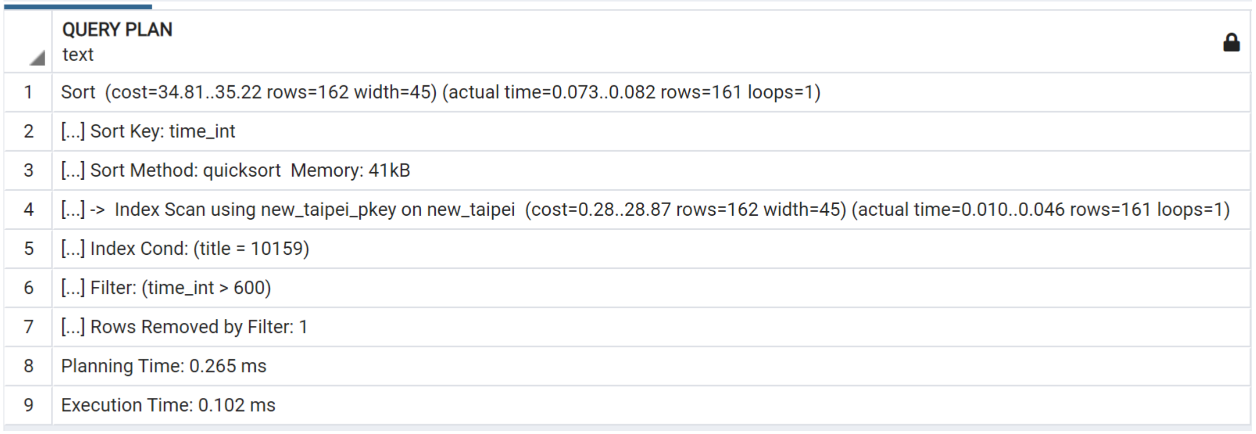
So I try to forbid the sequential scan, and do the command again

```
set enable_seqscan TO off;
```

Below shows the result of adding hash index and B tree index separately with sequential scan forbidden. I found that the actual time of scanning might be lower? But perhaps due to the higher cost, the optimizer still didn't choose index scan without restriction, so it seems that adding index in the table with 600 amount of data won't benefit the scan that much.



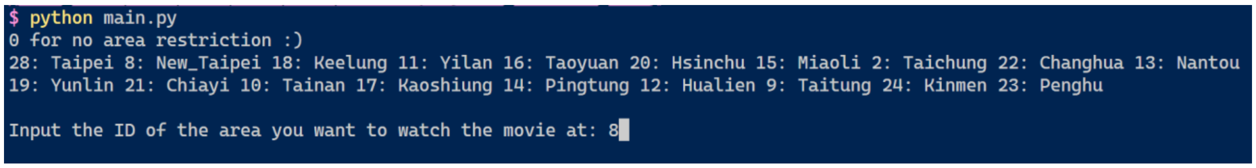
	QUERY PLAN text
1	Sort (cost=34.81..35.22 rows=162 width=45) (actual time=0.103..0.111 rows=161 loops=1)
2	[...] Sort Key: time_int
3	[...] Sort Method: quicksort Memory: 41kB
4	[...] -> Index Scan using new_taipei_pkey on new_taipei (cost=0.28..28.87 rows=162 width=45) (actual time=0.020..0.053 rows=161 loops=1)
5	[...] Index Cond: (title = 10159)
6	[...] Filter: (time_int > 600)
7	[...] Rows Removed by Filter: 1
8	Planning Time: 0.107 ms
9	Execution Time: 0.134 ms



	QUERY PLAN text
1	Sort (cost=34.81..35.22 rows=162 width=45) (actual time=0.073..0.082 rows=161 loops=1)
2	[...] Sort Key: time_int
3	[...] Sort Method: quicksort Memory: 41kB
4	[...] -> Index Scan using new_taipei_pkey on new_taipei (cost=0.28..28.87 rows=162 width=45) (actual time=0.010..0.046 rows=161 loops=1)
5	[...] Index Cond: (title = 10159)
6	[...] Filter: (time_int > 600)
7	[...] Rows Removed by Filter: 1
8	Planning Time: 0.265 ms
9	Execution Time: 0.102 ms

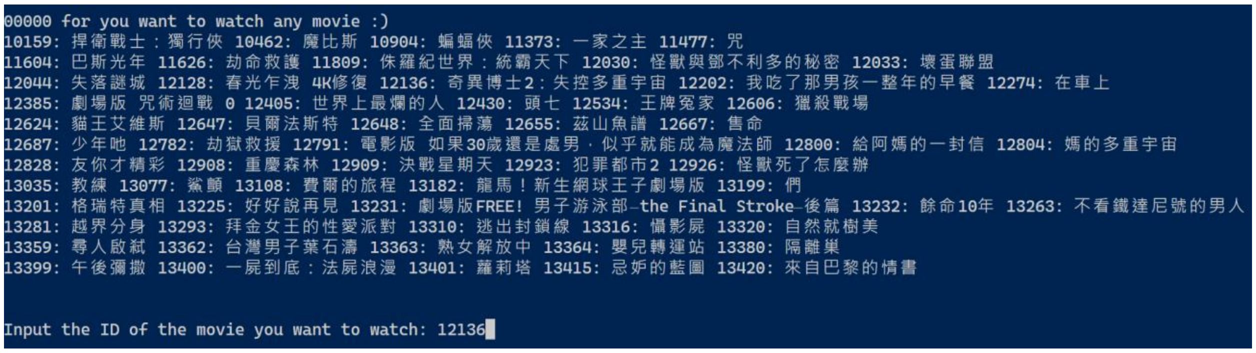
Application & related SQL queries

As mentioned in the motivation, my goal is to implement a movie showings query service. Below shows the interface and the procedure of the service.



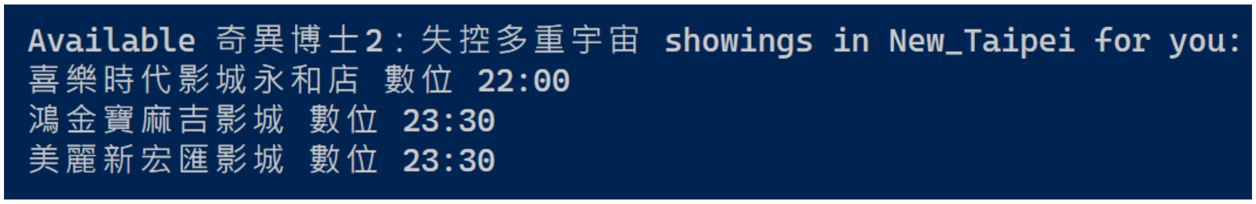
```
$ python main.py
0 for no area restriction :)
28: Taipei 8: New_Taipei 18: Keelung 11: Yilan 16: Taoyuan 20: Hsinchu 15: Miaoli 2: Taichung 22: Changhua 13: Nantou
19: Yunlin 21: Chiayi 10: Tainan 17: Kaoshiung 14: Pingtung 12: Hualien 9: Taitung 24: Kinmen 23: Penghu

Input the ID of the area you want to watch the movie at: 8
```

After running the program, we will first get the area and its corresponding IDs list, and we can choose where we want to watch the movie at. In above case I select New_Taipei city.



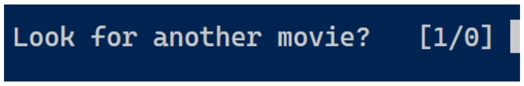
```
00000 for you want to watch any movie :)
10159: 捍衛戰士：獨行俠 10462: 魔比斯 10904: 蝙蝠俠 11373: 一家之主 11477: 咒
11604: 巴斯光年 11626: 劫命救護 11809: 侏羅紀世界：統霸天下 12030: 怪獸與鄧不利多的秘密 12033: 壞蛋聯盟
12044: 失落謎城 12128: 春光乍洩 4K修復 12136: 奇異博士2：失控多重宇宙 12202: 我吃了那男孩一整年的早餐 12274: 在車上
12385: 劇場版 咒術迴戰 0 12405: 世界上最爛的人 12430: 頭七 12534: 王牌冤家 12606: 獵殺戰場
12624: 貓王艾維斯 12647: 貝爾法斯特 12648: 全面掃蕩 12655: 茲山魚譜 12667: 售命
12687: 少年吔 12782: 劫獄救援 12791: 電影版 如果30歲還是處男，似乎就能成為魔法師 12800: 給阿媽的一封信 12804: 媽的多重宇宙
12828: 友你才精彩 12908: 重慶森林 12909: 決戰星期天 12923: 犯罪都市2 12926: 怪獸死了怎麼辦
13035: 教練 13077: 鯊顫 13108: 費爾的旅程 13182: 龍馬！新生網球王子劇場版 13199: 們
13201: 格瑞特真相 13225: 好好說再見 13231: 劇場版FREE! 男子游泳部–the Final Stroke–後篇 13232: 餘命10年 13263: 不看鐵達尼號的男人
13281: 越界分身 13293: 拜金女王的性愛派對 13310: 逃出封鎖線 13316: 懾影屍 13320: 自然就樹美
13359: 尋人啟弒 13362: 台灣男子葉石濤 13363: 熟女解放中 13364: 嬰兒轉運站 13380: 隔離巢
13399: 午後彌撒 13400: 一屍到底：法屍浪漫 13401: 蘿莉塔 13415: 忌妒的藍圖 13420: 來自巴黎的情書


Input the ID of the movie you want to watch: 12136
```

After selecting area, we will get a list with all of the premiere movies and their corresponding IDs, and then it's time for choosing movie. In this case, I choose *Doctor Strange 2*.



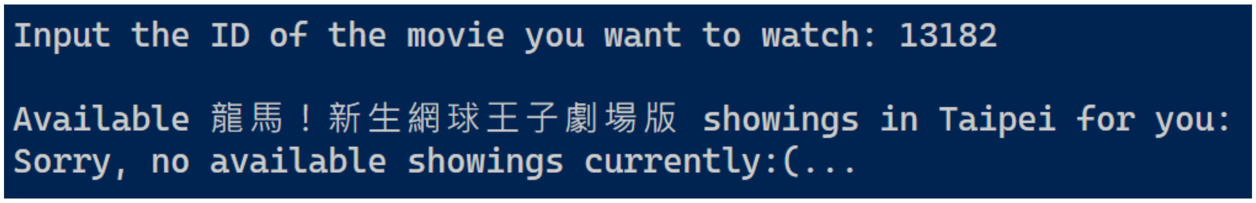
```
Available 奇異博士2：失控多重宇宙 showings in New_Taipei for you:
喜樂時代影城永和店 數位 22:00
鴻金寶麻吉影城 數位 23:30
美麗新宏匯影城 數位 23:30
```

We will get all the available showings depends on current time (it's now 20:27 when I do the above command) in New_Taipei city.



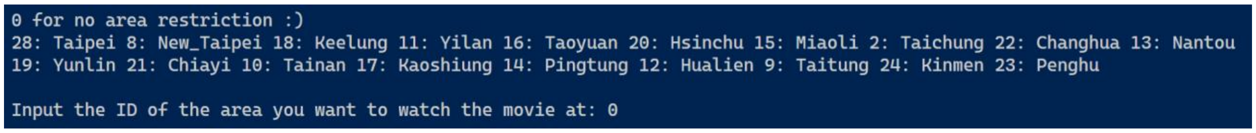
```
Look for another movie?  [1/0]
```

1 for keep querying

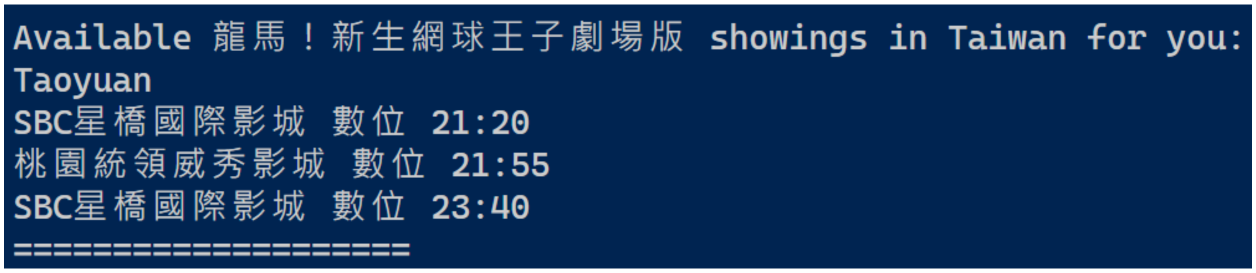


```
Input the ID of the movie you want to watch: 13182

Available 龍馬！新生網球王子劇場版 showings in Taipei for you:
Sorry, no available showings currently:(...
```

If unfortunate the movie we want to watch has no available showings in the city we located, then we can select 0 for no area restriction which means we just want to watch that movie no matter where it is.



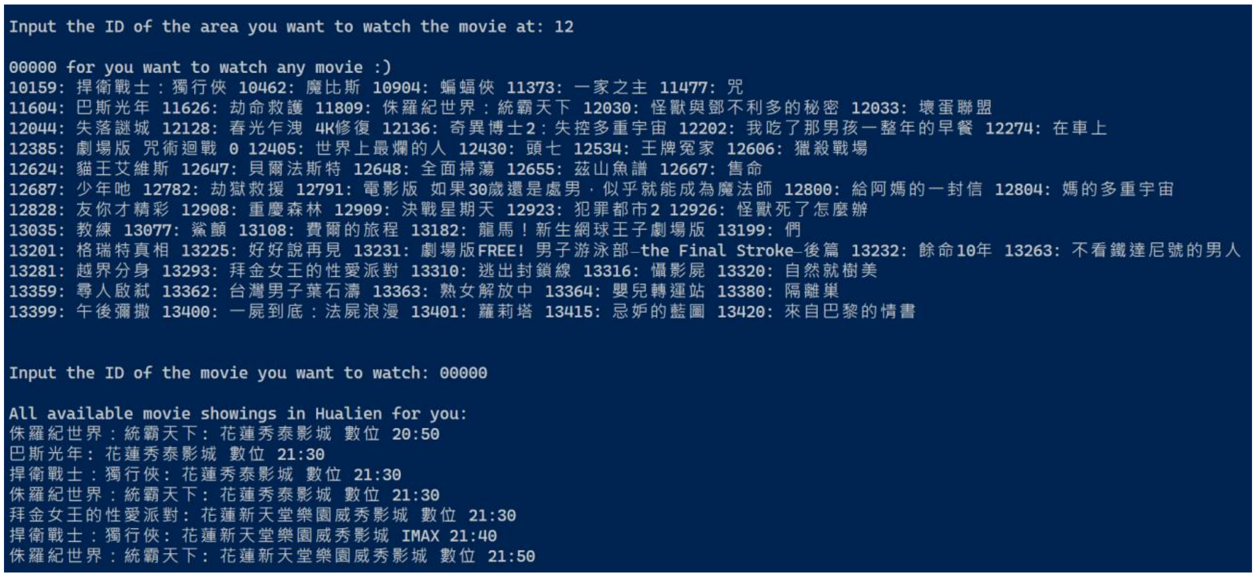
```
0 for no area restriction :)
28: Taipei 8: New_Taipei 18: Keelung 11: Yilan 16: Taoyuan 20: Hsinchu 15: Miaoli 2: Taichung 22: Changhua 13: Nantou
19: Yunlin 21: Chiayi 10: Tainan 17: Kaoshiung 14: Pingtung 12: Hualien 9: Taitung 24: Kinmen 23: Penghu

Input the ID of the area you want to watch the movie at: 0
```



```
Available 龍馬！新生網球王子劇場版 showings in Taiwan for you:
Taoyuan
SBC星橋國際影城 數位 21:20
桃園統領威秀影城 數位 21:55
SBC星橋國際影城 數位 23:40
====================
```

Then we will get all available showings of that movie in Taiwan. In this case, we are only able to watch 龍馬!新生網球王子劇場版 in Taoyuan city.

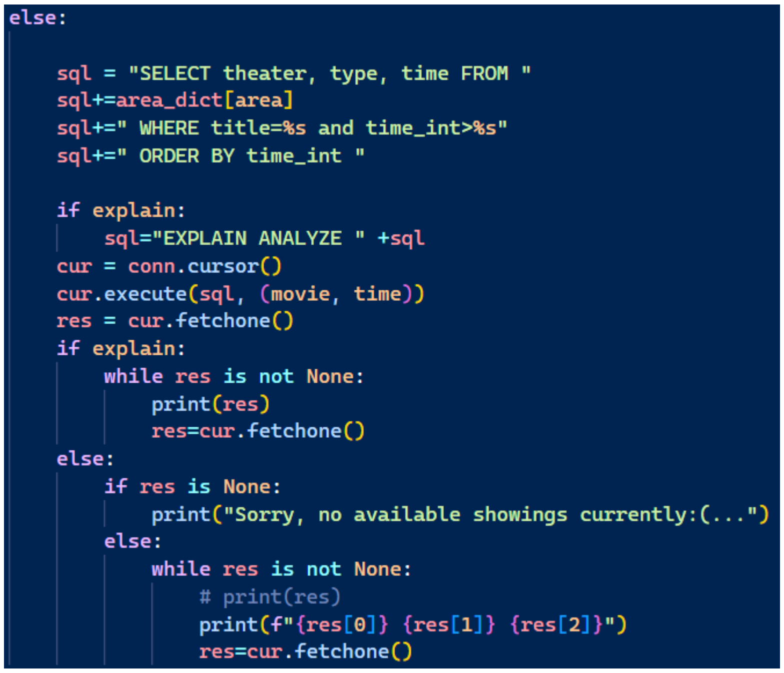
What if we are on a trip in other city and we are just in the mood that we want to watch a movie whatever it is for pastime, then we can select 00000 for we want to watch any movie.



```
Input the ID of the area you want to watch the movie at: 12

00000 for you want to watch any movie :)
10159: 捍衛戰士：獨行俠 10462: 魔比斯 10904: 蝙蝠俠 11373: 一家之主 11477: 咒
11604: 巴斯光年 11626: 劫命救護 11809: 侏羅紀世界：統霸天下 12030: 怪獸與鄧不利多的秘密 12033: 壞蛋聯盟
12044: 失落謎城 12128: 春光乍洩 4K修復 12136: 奇異博士2：失控多重宇宙 12202: 我吃了那男孩一整年的早餐 12274: 在車上
12385: 劇場版 咒術迴戰 0 12405: 世界上最爛的人 12430: 頭七 12534: 王牌冤家 12606: 獵殺戰場
12624: 貓王艾維斯 12647: 貝爾法斯特 12648: 全面掃蕩 12655: 茲山魚譜 12667: 售命
12687: 少年吔 12782: 劫獄救援 12791: 電影版 如果30歲還是處男，似乎就能成為魔法師 12800: 給阿媽的一封信 12804: 媽的多重宇宙
12828: 友你才精彩 12908: 重慶森林 12909: 決戰星期天 12923: 犯罪都市2 12926: 怪獸死了怎麼辦
13035: 教練 13077: 鯊顫 13108: 費爾的旅程 13182: 龍馬！新生網球王子劇場版 13199: 們
13201: 格瑞特真相 13225: 好好說再見 13231: 劇場版FREE! 男子游泳部—the Final Stroke—後篇 13232: 餘命10年 13263: 不看鐵達尼號的男人
13281: 越界分身 13293: 拜金女王的性愛派對 13310: 逃出封鎖線 13316: 懾影屍 13320: 自然就樹美
13359: 尋人啟弒 13362: 台灣男子葉石濤 13363: 熟女解放中 13364: 嬰兒轉運站 13380: 隔離巢
13399: 午後彌撒 13400: 一屍到底：法屍浪漫 13401: 蘿莉塔 13415: 忌妒的藍圖 13420: 來自巴黎的情書


Input the ID of the movie you want to watch: 00000

All available movie showings in Hualien for you:
侏羅紀世界：統霸天下: 花蓮秀泰影城 數位 20:50
巴斯光年: 花蓮秀泰影城 數位 21:30
捍衛戰士：獨行俠: 花蓮秀泰影城 數位 21:30
侏羅紀世界：統霸天下: 花蓮秀泰影城 數位 21:30
拜金女王的性愛派對: 花蓮新天堂樂園威秀影城 數位 21:30
捍衛戰士：獨行俠: 花蓮新天堂樂園威秀影城 IMAX 21:40
侏羅紀世界：統霸天下: 花蓮新天堂樂園威秀影城 數位 21:50
```

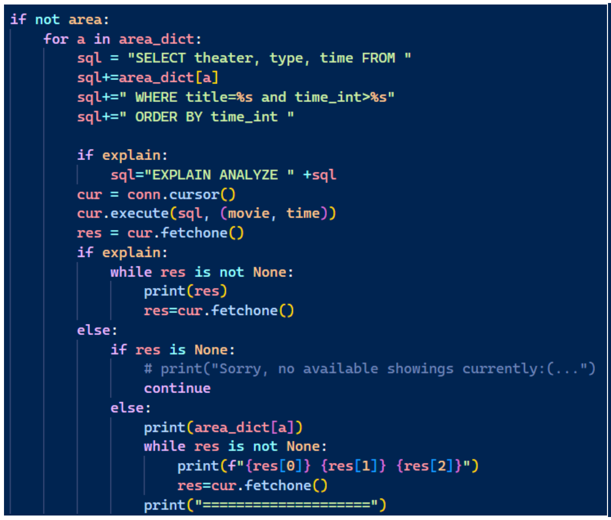
Then we will get all available movie showings in the city we are at.

Below shows the main query I used in this project, which take area id, movie id and current time as arguments. The program will get the current time and transfer it into integer, the area id and movie id are got from the user. By this query, we can get the currently available showings of the designated movie and area in the order of time closeness.

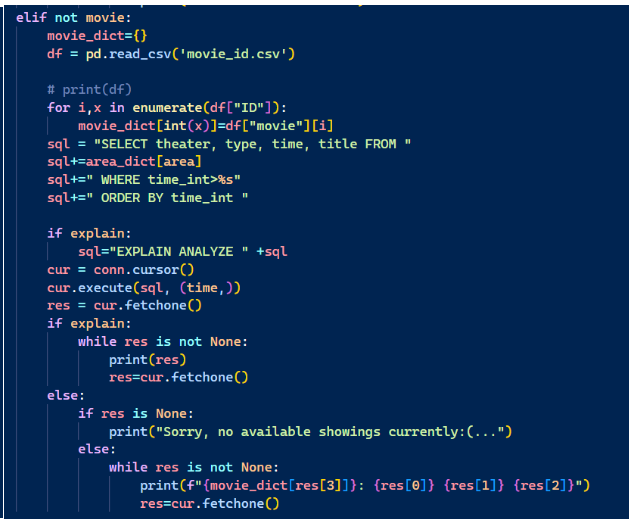


```python
else:

    sql = "SELECT theater, type, time FROM "
    sql+=area_dict[area]
    sql+=" WHERE title=%s and time_int>%s"
    sql+=" ORDER BY time_int "

    if explain:
        sql="EXPLAIN ANALYZE " +sql
    cur = conn.cursor()
    cur.execute(sql, (movie, time))
    res = cur.fetchone()
    if explain:
        while res is not None:
            print(res)
            res=cur.fetchone()
    else:
        if res is None:
            print("Sorry, no available showings currently:(...")
        else:
            while res is not None:
                # print(res)
                print(f"{res[0]} {res[1]} {res[2]}")
                res=cur.fetchone()
```

The rest two are for the case that no area and no movie are designated.



```python
if not area:
    for a in area_dict:
        sql = "SELECT theater, type, time FROM "
        sql+=area_dict[a]
        sql+=" WHERE title=%s and time_int>%s"
        sql+=" ORDER BY time_int "

        if explain:
            sql="EXPLAIN ANALYZE " +sql
        cur = conn.cursor()
        cur.execute(sql, (movie, time))
        res = cur.fetchone()
        if explain:
            while res is not None:
                print(res)
                res=cur.fetchone()
        else:
            if res is None:
                # print("Sorry, no available showings currently:(...")
                continue
            else:
                print(area_dict[a])
                while res is not None:
                    print(f"{res[0]} {res[1]} {res[2]}")
                    res=cur.fetchone()
                print("====================")
```



```python
elif not movie:
    movie_dict={}
    df = pd.read_csv('movie_id.csv')

    # print(df)
    for i,x in enumerate(df["ID"]):
        movie_dict[int(x)]=df["movie"][i]
    sql = "SELECT theater, type, time, title FROM "
    sql+=area_dict[area]
    sql+=" WHERE time_int>%s"
    sql+=" ORDER BY time_int "

    if explain:
        sql="EXPLAIN ANALYZE " +sql
    cur = conn.cursor()
    cur.execute(sql, (time,))
    res = cur.fetchone()
    if explain:
        while res is not None:
            print(res)
            res=cur.fetchone()
    else:
        if res is None:
            print("Sorry, no available showings currently:(...")
        else:
            while res is not None:
                print(f"{movie_dict[res[3]]}: {res[0]} {res[1]} {res[2]}")
                res=cur.fetchone()
```