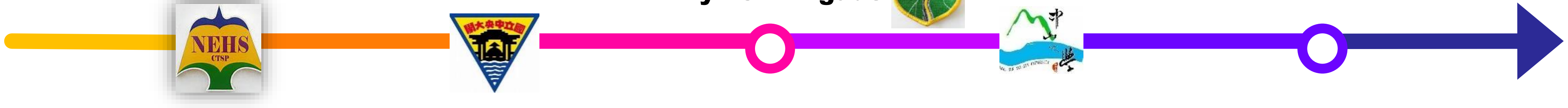
MongoDB Applied Research

AW0361 蘇柏瑜(Brian-Su)

Wistron NeWeb Corp.

☐ Normal ☐ Internal Use [✓ Confidential] ☐ Restricted Confidential

GitHub brian09088



2015-2018
National Experimental High School (Central TW Science Park)

2018-2022
NCU Math-Science

2020~2022
Minor-specialty: Computer Science & Programming

2020 / 2022 Summer
Military Service
Army 257 Brigade

2021~2023
snapask
Online-Tutor

2023/02~
Graduate student Computer-Science Engineering
NSYSU-WCMC Lab

2023/07~2023/08
WNC Summer Internship

Project Outline

1. Software application installation & Environment set up

2. BOM Transfer to Mongo Format

3. Transferred BOM Import To MongoDB

4. BOM UX Query & Export

5. Asis & Tobe Compare

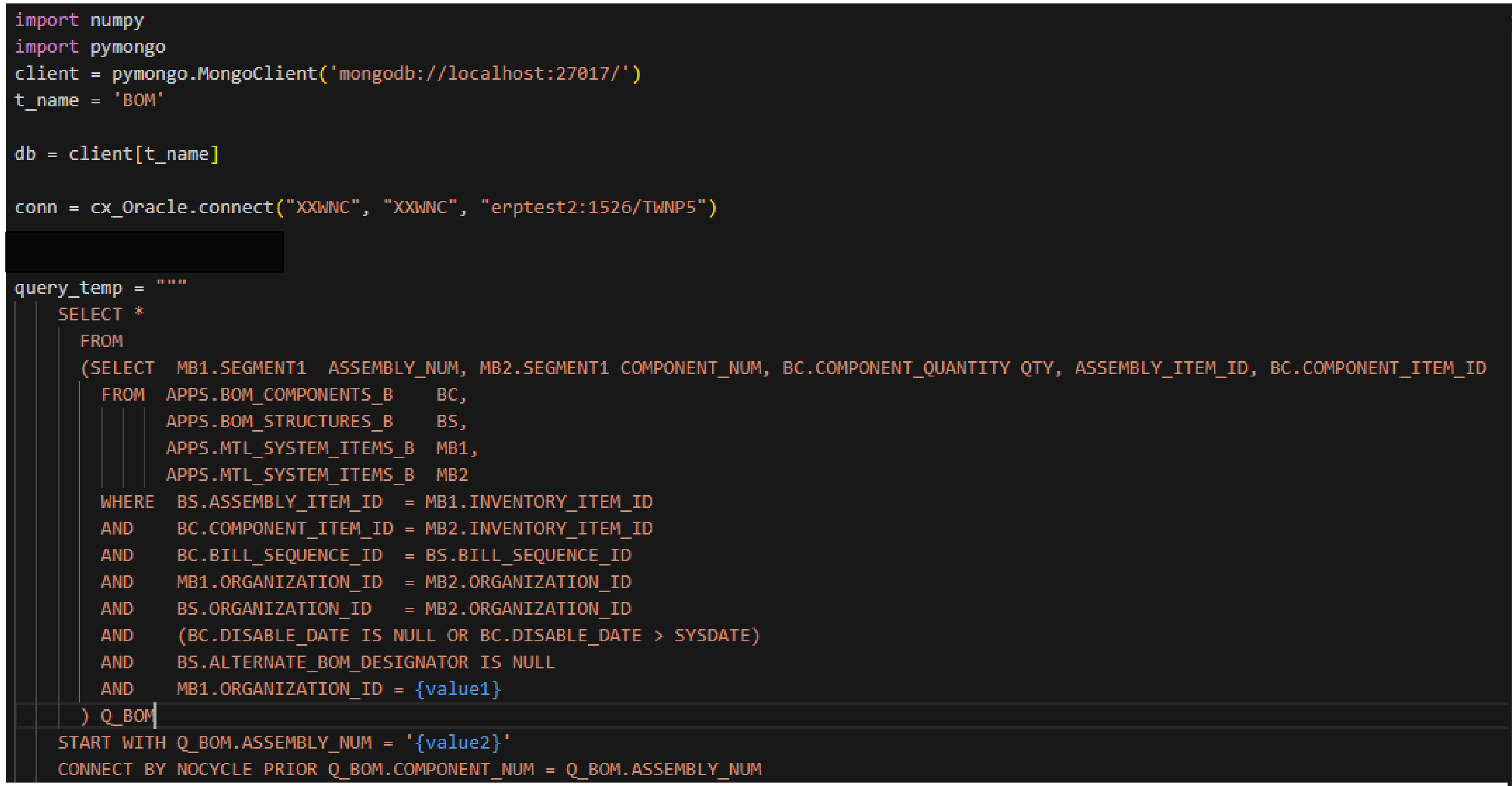
BOM Transfer & Import

Support Type \ Env	Python	MongoDB	OracleDB
SQL-query	✔	✔(partial)	✔
Html	✔	✔(csv,tsv)	✔(xml)
Text	✔	✔	✔
Json	✔	✔	
Dictionary	✔	✔	
Dataframe	✔		
cursor	✔		
List	✔		

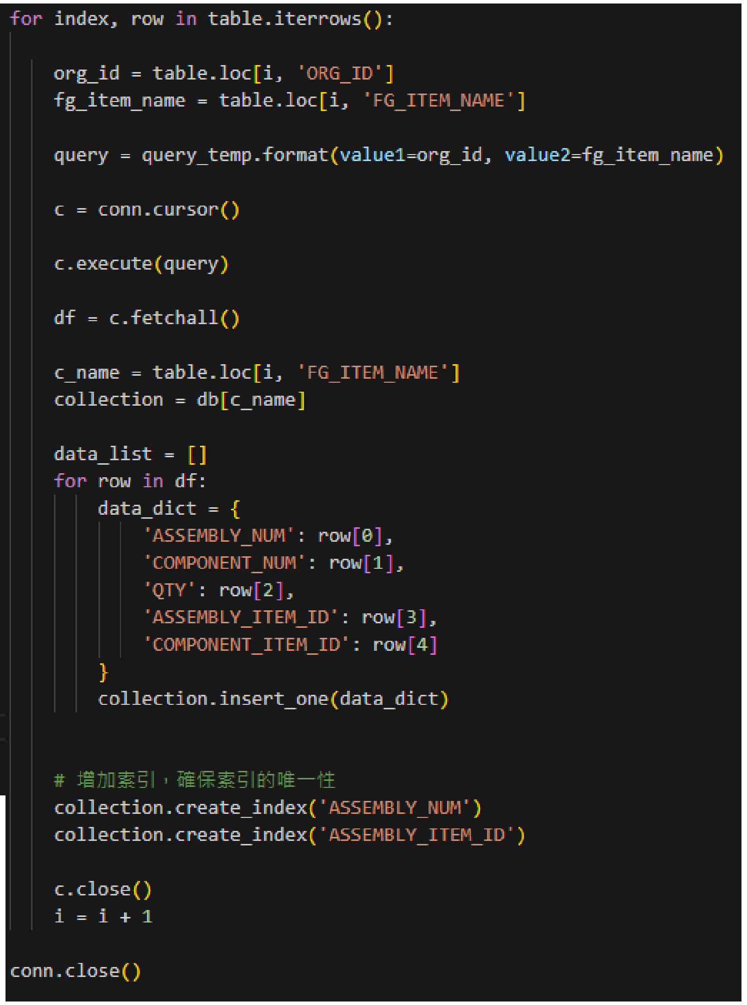
Transfer order(top-down)

1. SQL-query (load out from Oracle)
2. Cursor execute traverse query
3. Fetch dataframe
4. Dataframe to dictionary (get column name)
5. Insert to MongoDB
6. Create index and ensure the uniqueness

BOM MongoDB Import (code)

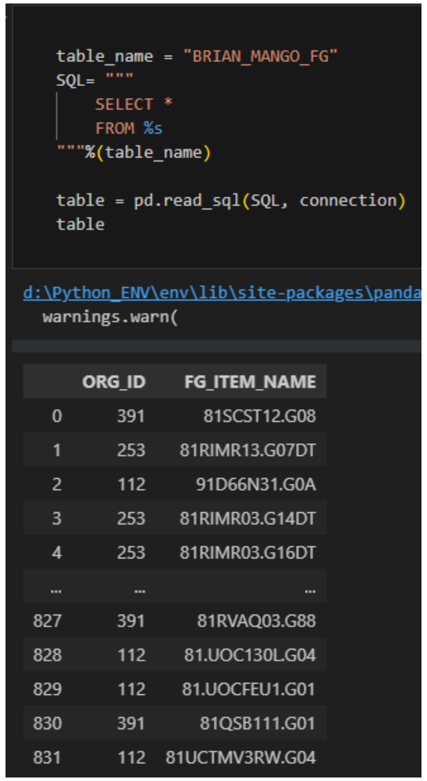


```
import numpy
import pymongo
client = pymongo.MongoClient('mongodb://localhost:27017/')
t_name = 'BOM'

db = client[t_name]

conn = cx_Oracle.connect("XXWNC", "XXWNC", "erptest2:1526/TWNP5")

[redacted]

query_temp = """
    SELECT *
      FROM
      (SELECT  MB1.SEGMENT1  ASSEMBLY_NUM, MB2.SEGMENT1 COMPONENT_NUM, BC.COMPONENT_QUANTITY QTY, ASSEMBLY_ITEM_ID, BC.COMPONENT_ITEM_ID
        FROM  APPS.BOM_COMPONENTS_B    BC,
              APPS.BOM_STRUCTURES_B    BS,
              APPS.MTL_SYSTEM_ITEMS_B  MB1,
              APPS.MTL_SYSTEM_ITEMS_B  MB2
        WHERE  BS.ASSEMBLY_ITEM_ID  = MB1.INVENTORY_ITEM_ID
        AND    BC.COMPONENT_ITEM_ID = MB2.INVENTORY_ITEM_ID
        AND    BC.BILL_SEQUENCE_ID  = BS.BILL_SEQUENCE_ID
        AND    MB1.ORGANIZATION_ID  = MB2.ORGANIZATION_ID
        AND    BS.ORGANIZATION_ID   = MB2.ORGANIZATION_ID
        AND    (BC.DISABLE_DATE IS NULL OR BC.DISABLE_DATE > SYSDATE)
        AND    BS.ALTERNATE_BOM_DESIGNATOR IS NULL
        AND    MB1.ORGANIZATION_ID = {value1}
      ) Q_BOM
    START WITH Q_BOM.ASSEMBLY_NUM = '{value2}'
    CONNECT BY NOCYCLE PRIOR Q_BOM.COMPONENT_NUM = Q_BOM.ASSEMBLY_NUM
```



```
for index, row in table.iterrows():

    org_id = table.loc[i, 'ORG_ID']
    fg_item_name = table.loc[i, 'FG_ITEM_NAME']

    query = query_temp.format(value1=org_id, value2=fg_item_name)

    c = conn.cursor()

    c.execute(query)

    df = c.fetchall()

    c_name = table.loc[i, 'FG_ITEM_NAME']
    collection = db[c_name]

    data_list = []
    for row in df:
        data_dict = {
            'ASSEMBLY_NUM': row[0],
            'COMPONENT_NUM': row[1],
            'QTY': row[2],
            'ASSEMBLY_ITEM_ID': row[3],
            'COMPONENT_ITEM_ID': row[4]
        }
        collection.insert_one(data_dict)


    # 增加索引，確保索引的唯一性
    collection.create_index('ASSEMBLY_NUM')
    collection.create_index('ASSEMBLY_ITEM_ID')

    c.close()
    i = i + 1

conn.close()
```

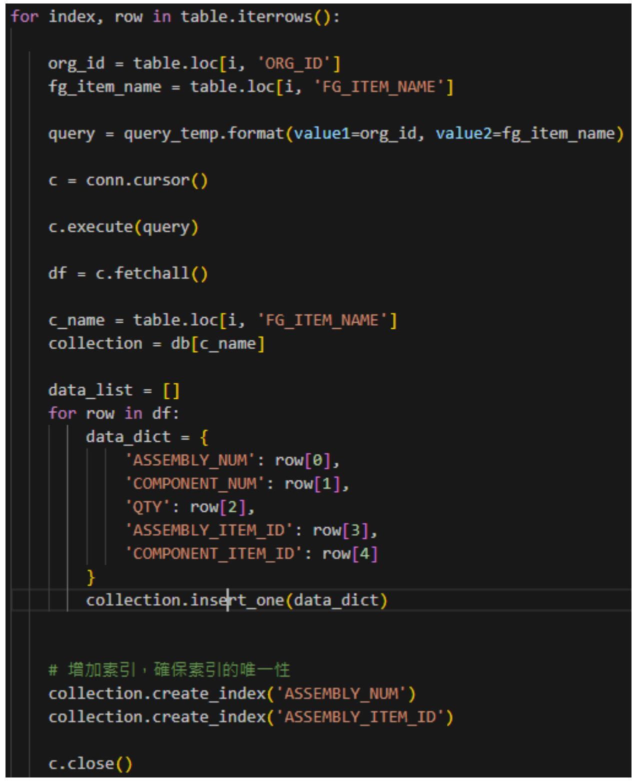
BOM MongoDB Import



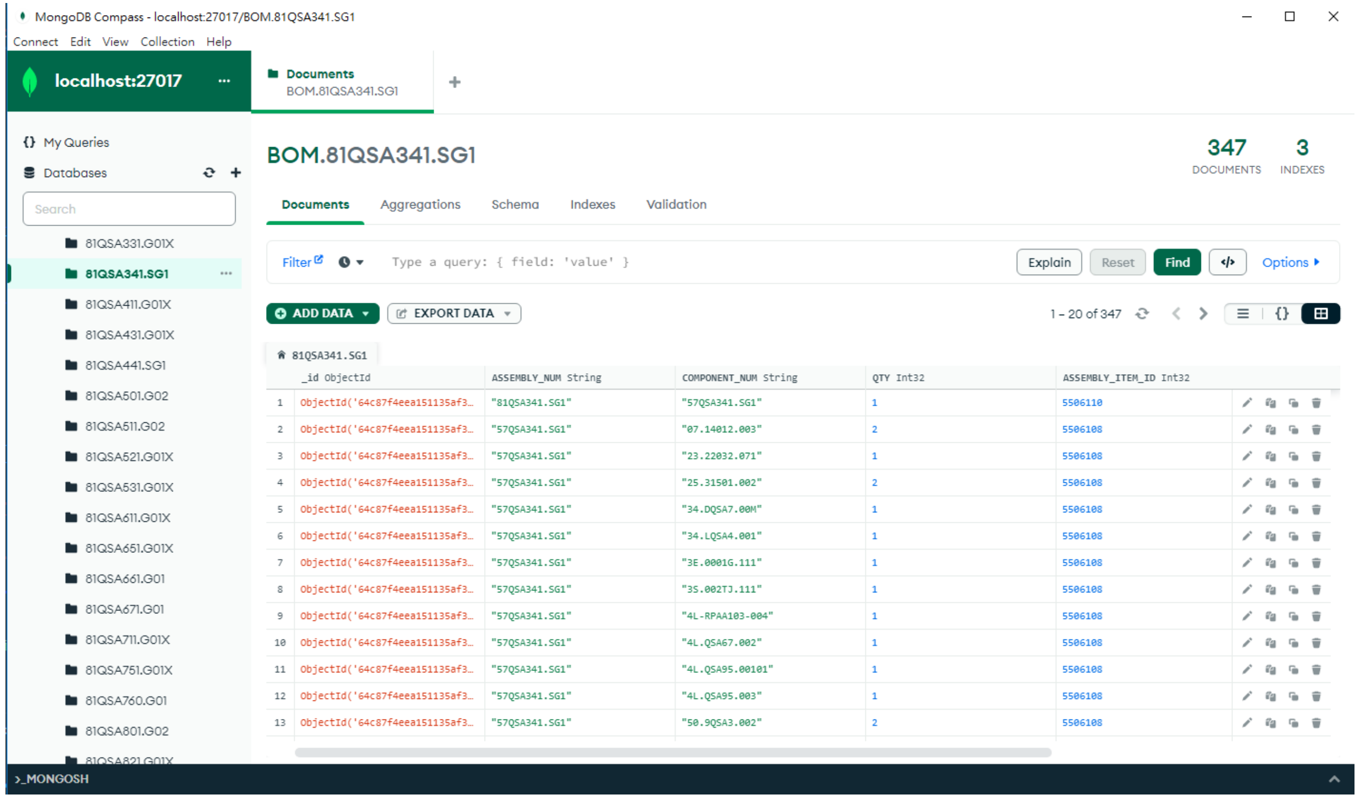
```
table_name = "BRIAN_MANGO_FG"
SQL= """
    SELECT *
    FROM %s
"""%(table_name)

table = pd.read_sql(SQL, connection)
table
```

```
d:\Python_ENV\env\lib\site-packages\panda
  warnings.warn(
```

	ORG_ID	FG_ITEM_NAME
0	391	81SCST12.G08
1	253	81RIMR13.G07DT
2	112	91D66N31.G0A
3	253	81RIMR03.G14DT
4	253	81RIMR03.G16DT
...	...	...
827	391	81RVAQ03.G88
828	112	81.UOC130L.G04
829	112	81.UOCFEU1.G01
830	391	81QSB111.G01
831	112	81UCTMV3RW.G04



```
for index, row in table.iterrows():

    org_id = table.loc[i, 'ORG_ID']
    fg_item_name = table.loc[i, 'FG_ITEM_NAME']

    query = query_temp.format(value1=org_id, value2=fg_item_name)

    c = conn.cursor()

    c.execute(query)

    df = c.fetchall()

    c_name = table.loc[i, 'FG_ITEM_NAME']
    collection = db[c_name]

    data_list = []
    for row in df:
        data_dict = {
            'ASSEMBLY_NUM': row[0],
            'COMPONENT_NUM': row[1],
            'QTY': row[2],
            'ASSEMBLY_ITEM_ID': row[3],
            'COMPONENT_ITEM_ID': row[4]
        }
        collection.insert_one(data_dict)


    # 增加索引，確保索引的唯一性
    collection.create_index('ASSEMBLY_NUM')
    collection.create_index('ASSEMBLY_ITEM_ID')

    c.close()
```



BOM.81QSA341.SG1

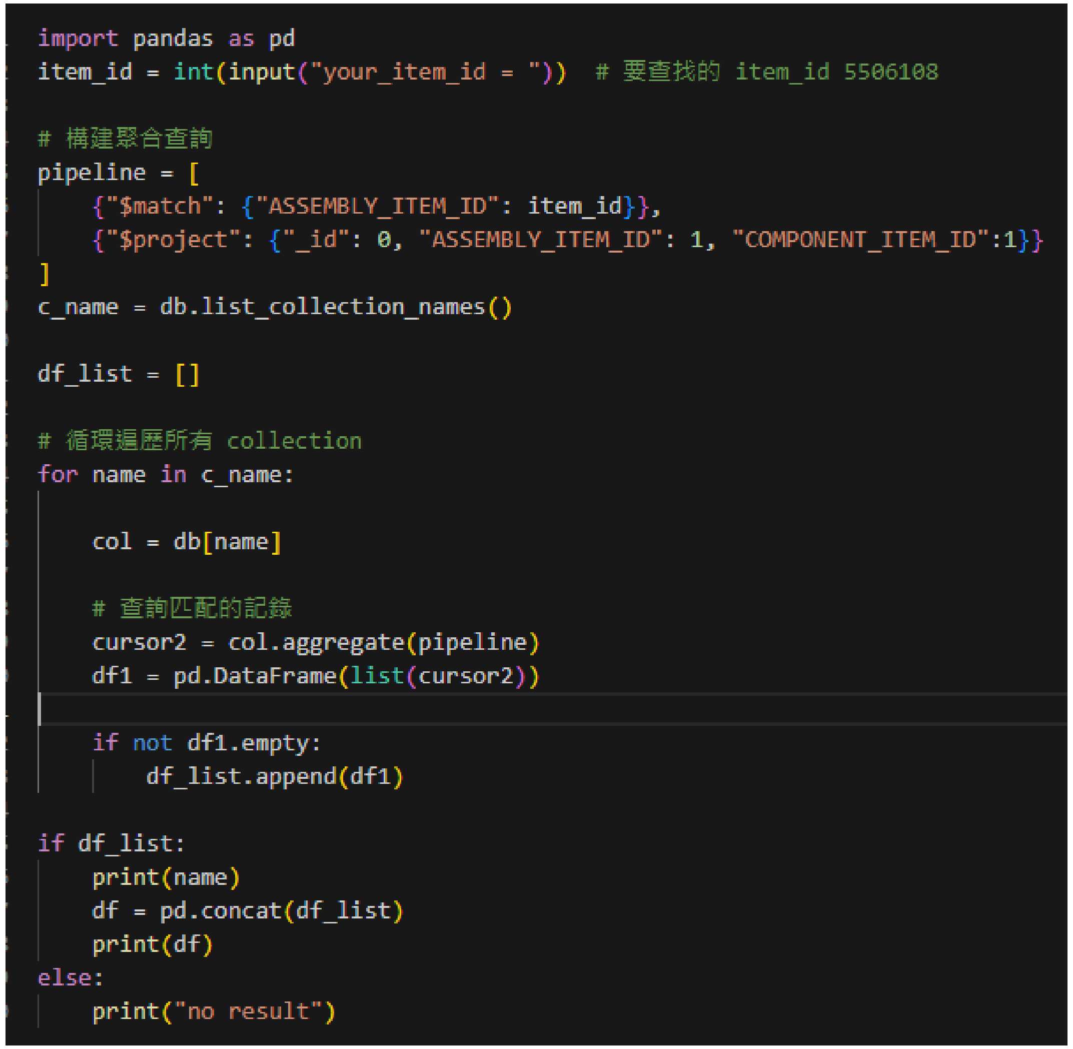
347 DOCUMENTS 3 INDEXES

1 – 20 of 347

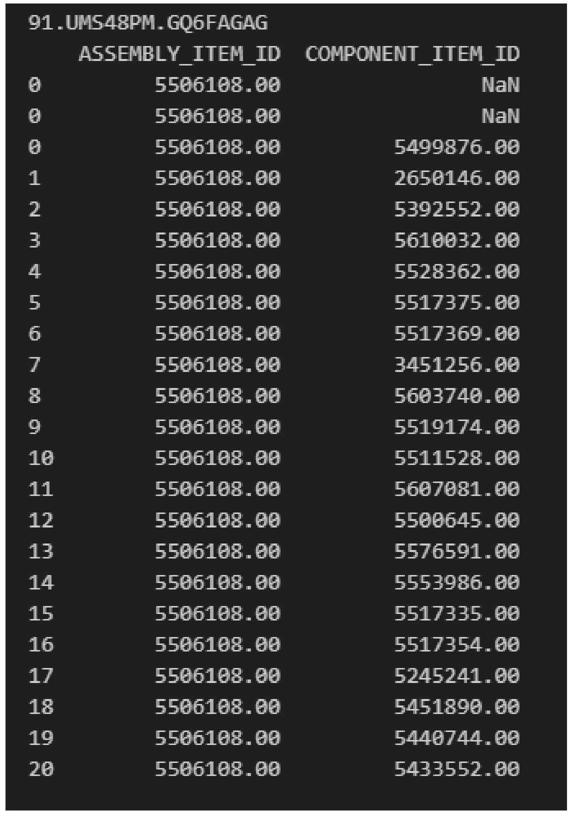
81QSA341.SG1

	_id ObjectId	ASSEMBLY_NUM String	COMPONENT_NUM String	QTY Int32	ASSEMBLY_ITEM_ID Int32
1	ObjectId('64c87f4eea151135af3…	"81QSA341.SG1"	"57QSA341.SG1"	1	5506110
2	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"07.14012.003"	2	5506108
3	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"23.22032.071"	1	5506108
4	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"25.31501.002"	2	5506108
5	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"34.DQSA7.00M"	1	5506108
6	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"34.LQSA4.001"	1	5506108
7	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"3E.0001G.111"	1	5506108
8	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"3S.002TJ.111"	1	5506108
9	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"4L-RPAA103-004"	1	5506108
10	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"4L.QSA67.002"	1	5506108
11	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"4L.QSA95.00101"	1	5506108
12	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"4L.QSA95.003"	1	5506108
13	ObjectId('64c87f4eea151135af3…	"57QSA341.SG1"	"50.9QSA3.002"	2	5506108

BOM UX Query (Items where used) & Export



```python
import pandas as pd
item_id = int(input("your_item_id = "))  # 要查找的 item_id 5506108

# 構建聚合查詢
pipeline = [
    {"$match": {"ASSEMBLY_ITEM_ID": item_id}},
    {"$project": {"_id": 0, "ASSEMBLY_ITEM_ID": 1, "COMPONENT_ITEM_ID":1}}
]
c_name = db.list_collection_names()

df_list = []

# 循環遍歷所有 collection
for name in c_name:

    col = db[name]

    # 查詢匹配的記錄
    cursor2 = col.aggregate(pipeline)
    df1 = pd.DataFrame(list(cursor2))

    if not df1.empty:
        df_list.append(df1)

if df_list:
    print(name)
    df = pd.concat(df_list)
    print(df)
else:
    print("no result")
```

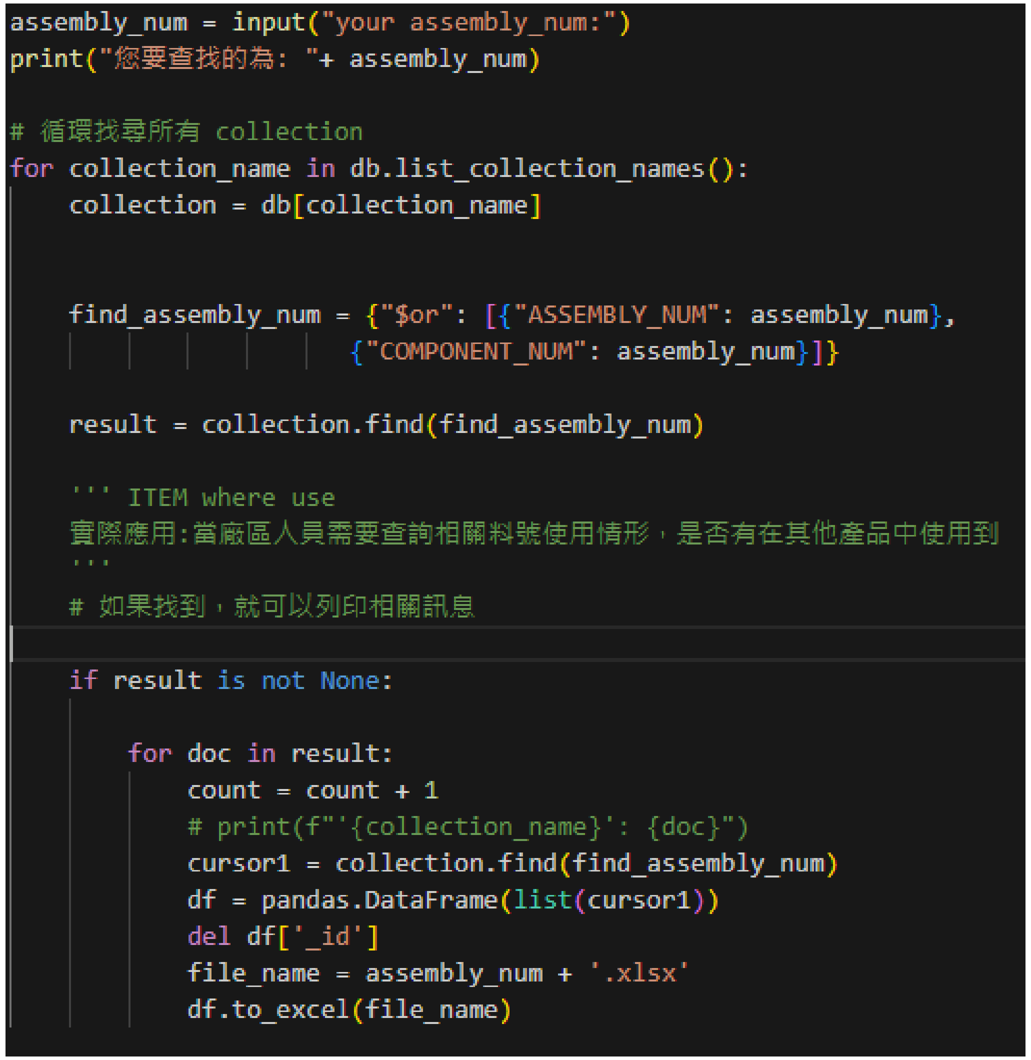


```
91.UMS48PM.GQ6FAGAG
    ASSEMBLY_ITEM_ID  COMPONENT_ITEM_ID
0         5506108.00                NaN
0         5506108.00                NaN
0         5506108.00         5499876.00
1         5506108.00         2650146.00
2         5506108.00         5392552.00
3         5506108.00         5610032.00
4         5506108.00         5528362.00
5         5506108.00         5517375.00
6         5506108.00         5517369.00
7         5506108.00         3451256.00
8         5506108.00         5603740.00
9         5506108.00         5519174.00
10        5506108.00         5511528.00
11        5506108.00         5607081.00
12        5506108.00         5500645.00
13        5506108.00         5576591.00
14        5506108.00         5553986.00
15        5506108.00         5517335.00
16        5506108.00         5517354.00
17        5506108.00         5245241.00
18        5506108.00         5451890.00
19        5506108.00         5440744.00
20        5506108.00         5433552.00
```

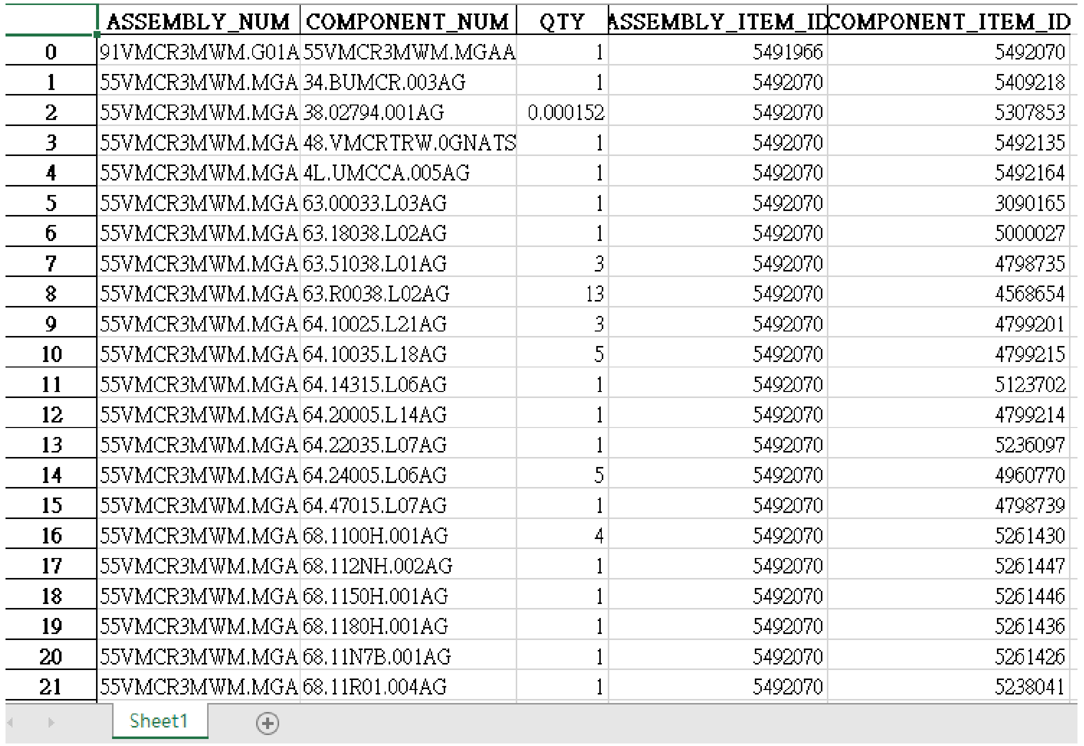
For more complex screening criteria:
Advanced query using aggregation list

List which components are included with this assembly item, under which item assembly list.

BOM (Structure of raw materials and finished products)



```python
assembly_num = input("your assembly_num:")
print("您要查找的為: "+ assembly_num)

# 循環找尋所有 collection
for collection_name in db.list_collection_names():
    collection = db[collection_name]


    find_assembly_num = {"$or": [{"ASSEMBLY_NUM": assembly_num},
                                 {"COMPONENT_NUM": assembly_num}]}

    result = collection.find(find_assembly_num)

    ''' ITEM where use
    實際應用:當廠區人員需要查詢相關料號使用情形，是否有在其他產品中使用到
    '''

    # 如果找到，就可以列印相關訊息

    if result is not None:

        for doc in result:
            count = count + 1
            # print(f"'{collection_name}': {doc}")
            cursor1 = collection.find(find_assembly_num)
            df = pandas.DataFrame(list(cursor1))
            del df['_id']
            file_name = assembly_num + '.xlsx'
            df.to_excel(file_name)
```



	ASSEMBLY_NUM	COMPONENT_NUM	QTY	ASSEMBLY_ITEM_ID	COMPONENT_ITEM_ID
0	91VMCR3MWM.G01A	55VMCR3MWM.MGAA	1	5491966	5492070
1	55VMCR3MWM.MGA	34.BUMCR.003AG	1	5492070	5409218
2	55VMCR3MWM.MGA	38.02794.001AG	0.000152	5492070	5307853
3	55VMCR3MWM.MGA	48.VMCRTRW.0GNATS	1	5492070	5492135
4	55VMCR3MWM.MGA	4L.UMCCA.005AG	1	5492070	5492164
5	55VMCR3MWM.MGA	63.00033.L03AG	1	5492070	3090165
6	55VMCR3MWM.MGA	63.18038.L02AG	1	5492070	5000027
7	55VMCR3MWM.MGA	63.51038.L01AG	3	5492070	4798735
8	55VMCR3MWM.MGA	63.R0038.L02AG	13	5492070	4568654
9	55VMCR3MWM.MGA	64.10025.L21AG	3	5492070	4799201
10	55VMCR3MWM.MGA	64.10035.L18AG	5	5492070	4799215
11	55VMCR3MWM.MGA	64.14315.L06AG	1	5492070	5123702
12	55VMCR3MWM.MGA	64.20005.L14AG	1	5492070	4799214
13	55VMCR3MWM.MGA	64.22035.L07AG	1	5492070	5236097
14	55VMCR3MWM.MGA	64.24005.L06AG	5	5492070	4960770
15	55VMCR3MWM.MGA	64.47015.L07AG	1	5492070	4798739
16	55VMCR3MWM.MGA	68.1100H.001AG	4	5492070	5261430
17	55VMCR3MWM.MGA	68.112NH.002AG	1	5492070	5261447
18	55VMCR3MWM.MGA	68.1150H.001AG	1	5492070	5261446
19	55VMCR3MWM.MGA	68.1180H.001AG	1	5492070	5261436
20	55VMCR3MWM.MGA	68.11N7B.001AG	1	5492070	5261426
21	55VMCR3MWM.MGA	68.11R01.004AG	1	5492070	5238041

According to the input product number to view all the included items below, the quantity and item number can also be used to see the relevance of product components through this table.

Expand the output form to the spreadsheet at the same time, which is more convenient for users to view.

Asis & Tobe Compare

- Wait for Virgil to test

MongoDB vs OracleDB

MongoDB	OracleDB
No-SQL(unstructed)	SQL(structed)
Easy to read and write	Structure must be created before writing
Data is easy to change	Changes need to move the structure
Data increase or decrease consumes resources	Large objects are difficult to manage
Open source and free	Expensive, must purchase a license

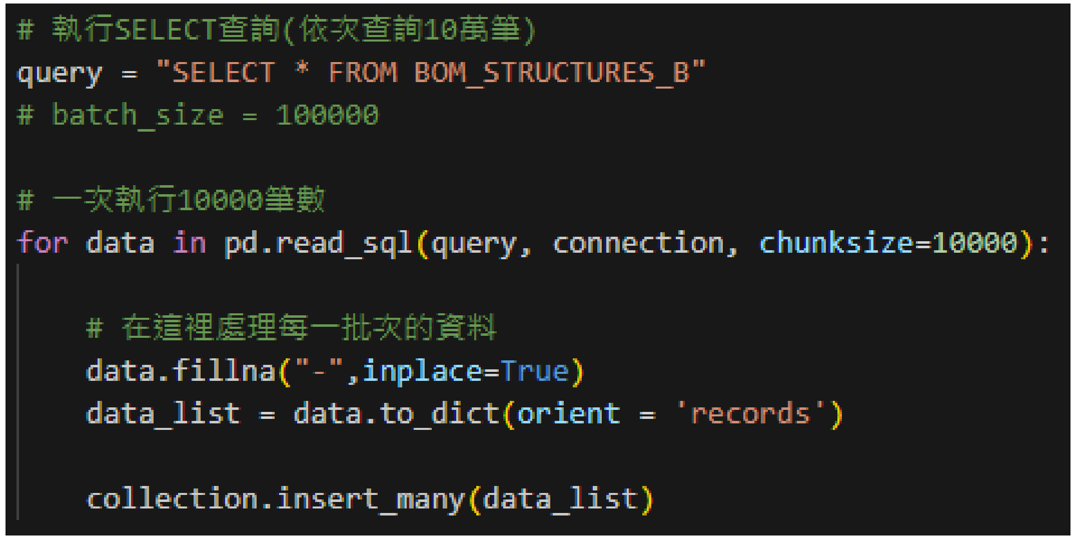
SQL vs No-SQL databases

	SQL	No-SQL
principle	ACID (Atomicity, Consistency, Isolation, Durability)	CAP Theorem (Impossible to satisfy at the same time: Consistency, Availability, Partition tolerance)
purpose	Transaction consistency	Eventual consistency
structure	Join Normalization, Denormalization	Document-keyvalue Nested-structure Informalization
expansion	Vertical-scaling Increase host cpu, memory	Horizontal-scaling Exploiting Nodes with Decentralized Systems
efficacy	sacrifice performance	high efficiency(Huge amount of data)
security	good security	Low security, low accuracy
Applicable Environment	Relevance data structured data Enterprise Resource Allocation	APP server : cash flow transaction, transfer money Weather data, stock market information

Bottleneck Solution

- The amount of data set is too large, resulting in memory exhaustion (16 GB) when reading data.

  Set the number of shard collections and the number of batch reads to ensure sufficient memory space. (batch_size & chunk_size)



```
# 執行SELECT查詢(依次查詢10萬筆)
query = "SELECT * FROM BOM_STRUCTURES_B"
# batch_size = 100000

# 一次執行10000筆數
for data in pd.read_sql(query, connection, chunksize=10000):

    # 在這裡處理每一批次的資料
    data.fillna("-",inplace=True)
    data_list = data.to_dict(orient = 'records')

    collection.insert_many(data_list)
```

Contribution to WNC

- Complete the feasibility test of MongoDB on the ERP system.
- Study the new database system MongoDB, provide a new mechanism for querying BOM tables, and improve user experience.
- At Engineering change orders (ECO) time, the whole change will affect the structure, and MongoDB may have an advantage at this time.

(Unstructured can handle a single situation independently)

Future Works

- In the future, if the data is too large for the machine to load, you can use MongoDB fragmentation technology to evenly distribute the data among multiple computers.

- Transaction can restore the status of the changed data, similar to small-scale backup and restore.

- Develop and test API servers to achieve message exchange for multiple users.

Q&A

WNC

Wistron NeWeb Corp.

Thank You!