PROJECT PRESENTATION

Prepared by: Animesh Khanna
Reporting Manager: Sonali Majumdar

Technology Used

- Mongo Db
- Pymongo
- Python
- Shell Scripting
- Unix commands

Objective and Introduction

Project 1(Aggregation Pipeline):

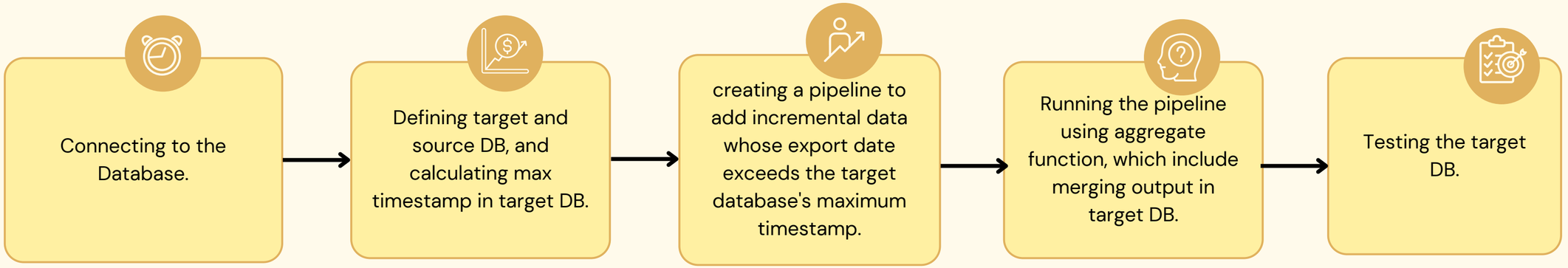
- To build an aggregation pipeline in PyMongo and count the events that a customer or recipient has engaged in through Moengage Campaign Customization.
- The source DB has all of the information from the customizing event.

Expected output:

To develop a target database that will have a count of various occurrences grouped by date and policy number.

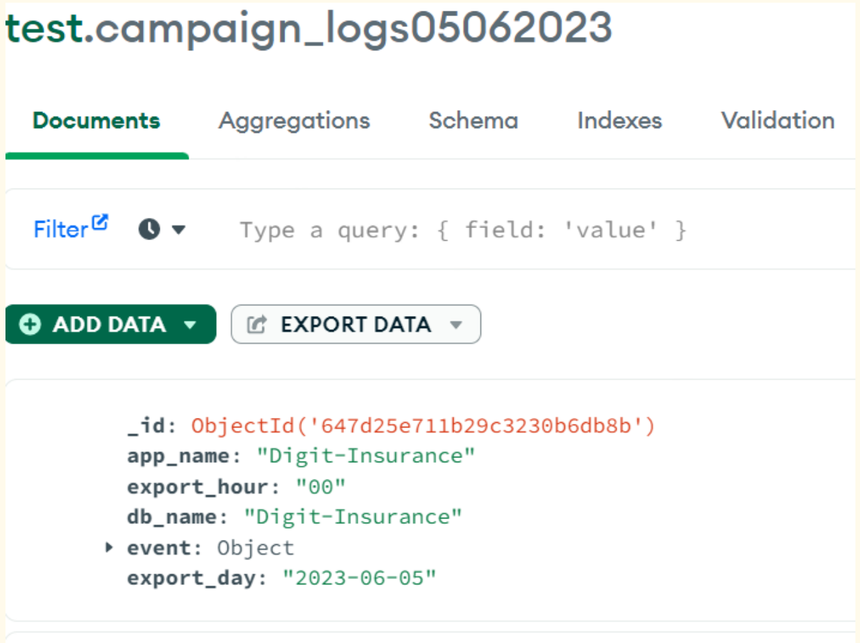
Workflow

Project 1



Project 1

Source DB



Target DB


```
{
 "_id": {
  "export_date": "2023-06-01",
  "policy_no": "69"
 },
 "Price_seen_count": 0,
 "SMS_clicked_count": 0,
 "SMS_delivered_count": 0,
 "SMS_sent_count": 0,
 "Whatsapp_delivered_count": 0,
 "Whatsapp_read_count": 0,
 "connector_sent_count": 0,
 "email_clicked_count": 0,
 "email_delivered_count": 1,
 "email_opened_count": 0,
 "email_sent_count": 0,
 "timestamp": "2023-07-28"
}
```


Expected

_id:ObjectId('6488a754245344bfa1222db0')
export_date : "2023-06-05"
policy_no : 719
email_sent_count: 2
email_delivered_count: 2
email_opened_count: 0
email_clicked_count: 0
connector_sent_count: 0
SMS_sent_count: 1
SMS_delivered_count: 1
SMS_clicked_count: 0
Whatsapp_read_count: 0
Whatsapp_delivered_count: 0
Price_seen_count: 0

Test Cases

Case 1: creating target DB
768MB

```python
#Animesh Khanna
#E0393
import pymongo
import datetime
import pprint
import datetime
from pymongo import MongoClient

print ("Start: ",datetime.datetime.now())
client = MongoClient("mongodb://intuser:5P0NPqF98bT0@10.20.11.21:2701
db=client['test']

s_collection=db['campaign_logs05062023']
t_collection = db['Animesh_Khanna_E0393']

#s_fvalue=s_collection.distinct("export_day")
#s_doc=s_collection.find({'export_day':{"$nin":t_collection.distinct(
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG COMMENTS

```
[Running] python -u "c:\Users\Animesh.khanna\Desktop\Animesh_Khanna_DB.py"
Start:  2023-07-28 12:58:55.451403
Pipeline1:  2023-07-28 12:58:58.910892
maxtime:  0001-01-01
End:  2023-07-28 12:59:55.095573

[Done] exited with code=0 in 61.863 seconds
```

61.863 sec

Case 2: Incremental Data
100 docs

```python
s_collection=db['testing_E0393']
t_collection = db['Animesh_Khanna_E0393']

#s_fvalue=s_collection.distinct("export_day")
#s_doc=s_collection.find({'export_day':{"$nin":t_collection.

#doc_insert=[]
#for doc in s_doc:
#        doc_insert.append(doc)
''''''

is_empty=t_collection.count_documents({}) ==0
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

```
[Running] python -u "c:\Users\Animesh.khanna\Desktop\Animesh_Khanna
Start:  2023-07-28 13:06:26.912692
Pipeline1:  2023-07-28 13:06:28.791523
maxtime:  2023-07-28
End:  2023-07-28 13:06:28.874295

[Done] exited with code=0 in 3.039 seconds
```

3.039 sec

Case 3: Large Database
1.4Gb

Animesh_Khanna_DB.py | update_date.py | delete.py | testing.py

C: > Users > Animesh.khanna > Desktop > Animesh_Khanna_DB.py > ...

```python
#Animesh Khanna
#E0393
import pymongo
import datetime
import pprint
import datetime
from pymongo import MongoClient

print ("Start: ",datetime.datetime.now())
client = MongoClient("mongodb://intuser:5P0NPqF98bT0@10.20.11.21:27017
db=client['test']

s_collection=db['test_set_sample2']
t_collection = db['Animesh_Khanna_E0393']
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG COMMENTS

```
[Running] python -u "c:\Users\Animesh.khanna\Desktop\Animesh_Khanna_DB.py"
Start:  2023-07-31 10:23:20.584248
Pipeline1:  2023-07-31 10:23:21.654946
maxtime:  2023-07-28
End:  2023-07-31 10:23:32.523280

[Done] exited with code=0 in 13.066 seconds
```

13.066sec

Scope of Improvement

- I have another method where I can utilize the current date as the maximum time and store it in another database for reference. Hence, the time is reduced.
- Since the target DB lacks a primary key, incremental data with the same policy number and export date will result in duplicate entries (only takes place when adding data with a backwards-dated export).

Objective and Introduction

Project 2(Log Analysis)

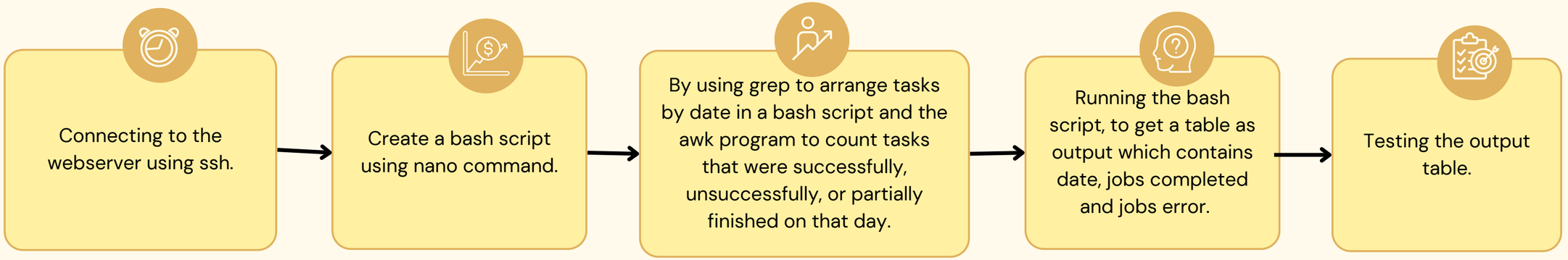
- To build a table that shows how many cron jobs were successfully completed and how many encountered errors on a given day on the web server.
- I have been granted access to the IP address, username, and password for the web server, as well as the location of the.log files. Using the location, I need to construct a script to create a table for the analysis of the .log files.

Expected output:

Date	Jobs Completed	Jobs Error	Partially Completed
YYYY-MM-DD			
YYYY-MM-DD			
YYYY-MM-DD			
YYYY-MM-DD			
YYYY-MM-DD			
YYYY-MM-DD			

Workflow

Project 2



Output Table

```
tern_user@ip-10-50-2-43 ~]$ nano new.sh
tern_user@ip-10-50-2-43 ~]$ ./new.sh
e       | Jobs Completed | Jobs Error | Partially Completed
230621_ | 23             | 25         | 6
230511_ | 23             | 25         | 6
230707_ | 28             | 7          | 4
230614_ | 23             | 25         | 6
230522_ | 16             | 23         | 6
230427_ | 23             | 25         | 6
3-05-08 | 12             | 0          | 0
230609_ | 23             | 25         | 6
230502_ | 23             | 25         | 6
3-05-09 | 12             | 0          | 0
230516_ | 23             | 25         | 6
3-06-21 | 12             | 0          | 0
3-05-06 | 12             | 0          | 0
230607_ | 23             | 25         | 6
3-06-20 | 12             | 0          | 0
3-05-07 | 12             | 0          | 0
230529_ | 23             | 25         | 6
3-06-23 | 12             | 0          | 0
3-05-04 | 12             | 0          | 0
230704_ | 12             | 16         | 4
3-06-22 | 12             | 0          | 0
3-05-05 | 12             | 0          | 0
230611_ | 23             | 25         | 6
230527_ | 23             | 25         | 6
3-06-25 | 12             | 0          | 0
3-05-02 | 12             | 0          | 0
```

Scope of Improvement

- The jobs can be fully analyzed, including the type of work, the time it ran, whether it failed and why, how many times it was run, etc.
- There are few cases when we are unable to identify a distinct keyword for success or failure in a.log file

Key Learnings

- Improving pipeline to reduce time.
- Learned Unix commands.
- Reduced the amount of time by learning to only add incremental data.
- Gained knowledge of bash script writing.
- Learned about cron jobs and gained experience with log file analysis.

Thank You!