STAATLICH
ANERKANNTE
HOCHSCHULE

BIG DATA & BUSINESS ANALYTICS

Data Engineering

Wikipedia Dataset
Project Report

Sankar Chinna Shanmugam
Matriculation No:11012443
Date: 14.09.2019

HOCHSCHULE
SRH HEIDELBERG
Intelligence in Learning

Contents

Synopsis……3
Resources & Tools Used in Pipeline……4
About the Wikipedia Data Source……5
Architecture of Wikipedia Data Pipeline……5
Configuration of Servers……7
Implementation Methods……8
Data Ingestion……8
Data Storage……9
Data Processing……11
Data Visualization……12
Project Goals……17
Bibliography……18

Synopsis

The project report consists of a proof of concept to develop and describe a prototype of a Data Pipeline. The various stages of pipeline have been described from Wikipedia dataset with appropriate path forward analysis are made it here.

Data pipeline stages namely:

- Data Ingestion
- Data Storage
- Data Processing
- Data Visualization

Report represents a clear cut Pipeline Vision of problem statement, explanation mechanisms, design and analysis, visual representation of results and output summaries.

In addition, pipeline clearly indicates the Hardware and Software requirements of project. The Architectural diagram explains about the components used and its related scalability, usability, flexibility, usability, portability of this system. Configuration of different servers used as a pre-requisites talk about the technology behind this project. Finally the implantations detail gives the overall development activities and details discussion of pipeline implementations.

As part of this implantations mechanism we are going to discuss data pipeline stages as mentioned above categories respectively. Also, in this pipeline implementation, the various tools are used in different stages with different strategies respectively. The interpretation of analysis to arrive made out of certain given input parameters are considered. The generation of graphical output is mandates the some of the variables to assume in runtime time which is explained appropriately.

Data pipeline address the different business question with matched result set as well some of the gaps in data set consists which does not answer the business questionnaires. Also it indicates to predict the some of the new business question occurrence and resolves the problem appropriately. The used additional data sources, which is publically available are useful to understand the business problem appropriately and provides the solutions to given problems.

Finally we are going to discuss the challenges faced in the project design and some the decision making while implementing the project and learning for the project pipeline were interesting to discuss and the overall experience faced in the proof of concept. As add on the references used in the pipe line implementations and learning sources are referenced in the final page of this report.

Resources & Tools Used in Pipeline

Following tools and resource are used to analyze Wikipedia dataset.

- Servers
  - JVM
  - Zookeeper
  - Kafka
  - Mongo

- DataSource
  - Wikipedia Stream Data

- APIs
  - Wikipedia
  - Wikipediaapi
  - Pymongo
  - Kafka
  - MongoClient
  - PrettyTable
  - Numpy
  - Matplotlib
  - Networkx

- IDE
  - Pycharm

- Operating Systems
  - Windows 7(64bit)

- Hardware
  - 4GB RAM
  - Inter Core i5 processor

- Software
  - MS Word - Project Report
  - MS PowerPoint- Presentation
  - Edraw Max - Architectural Diagram.

About the Wikipedia Data Source

The Wikipedia data source is quite interesting to each and every one where users could reference as online encyclopedia in the web world. Here most of the information is reliable and trusted to user based on the references the gained the sources. The content behind this sources are important to the user as referenced as well read and know the knowledge about A to Z.

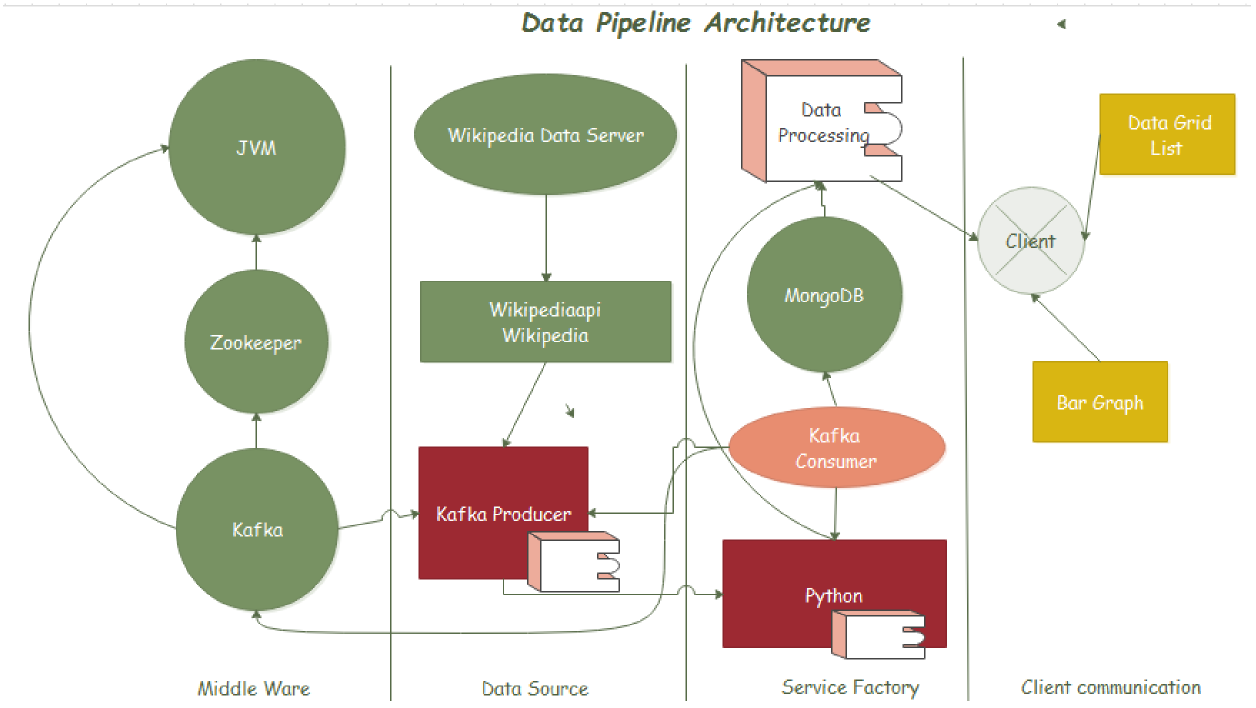
Curious to know about the functionality and exposed APIs are used to customize the user needs with this help of data pipeline from this data source. This data source in high level more theoretical, practical article, journals and publications on experiments will provides us to know the solutions to the give problem. Historical document collections are used to the user to know about history details. This data source is used all kinds of users on various places and different languages.

Architecture of Wikipedia Data Pipeline

The architecture of Wikipedia data pipeline looks like as following diagram. Here we have four different stages of pipeline implementation are considered to prove the our concept of project implementation.

Four layers of Architecture:

- Data Source
- Middle Ware
- Service Factory
- Client Consumption- (Business Case)



- Data Source
    - Wikipedia dataset source exposes the APIs to the consumers to fetch and update records based on client server communication media.
    - Python library uses these APIs, and then provides the developers to implement based on customer requirements or business case.
    - In the open source library factories there are so many libraries are available for implantation readily. Among in this project I have used “wikipedia” and “wikipediaapi” respectively.
    - Well, in case of Kafka Producer provides the opportunity to mediate the data follow without loss in client server communication. It is a intermediate interface to communicate between service factory and data source.
    - Kafka producer interacts with Wikipedia data source and fetches the data and return into the service factory where the service apis consumes the data and sends to backend.

- Middle Ware
    - To run the Kafka producer, in internally required Kafka Server, Zookeeper and JVM or JVR(Java Virtual Machine or Runtime).
    - Kafka Server manages the state of producer and consumer to maintain the scalability of data. Of course the Kafka Server interacts with JVM and zookeeper. Since it was built on top of it.
    - Zookeeper is to help the synchronization of complex deployment systems the which manages the services provided to Kafka server.
    - JVM or JAR are the libraries are used to generate the binary file conversion and provide the input to Operating System.

- Service Factory
    - Python packages helps to implements to generate the APIs for the consumers to interact with these services.
    - Kafka Consumer service, listen to capture the data sent by Kafka producer.
    - The captured data of Wikipedia dataset to the consumers is inserted into the backend database as MongoDB.
    - MongoDB Database is used to store the data in the backend. The stored data is used to analyze the data. These analyzes data is used to provide the solutions to the Business queries.
    - The Data Processing unit helps to process the data in structured manner and exposes the service APIs to consumers.

- Client Communication
    - The exposed service APIs or interfaces are used by the business case study analysis.
    - In this proof of concept I have addressed to answer the few business case analysis queries.
    - The first one is consumed by the tabular form of data and other one for bar chart generated to address business case.
    - The detailed information is explained in the followed sections.

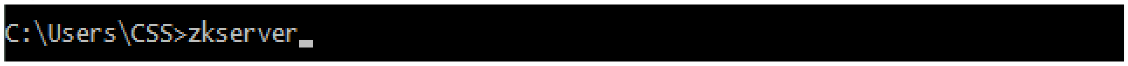
Configuration of Servers

In this section we are going to see the required server configuration details worked in this proof of concept.

- JVM Installation
  - Download the JVM and install it.

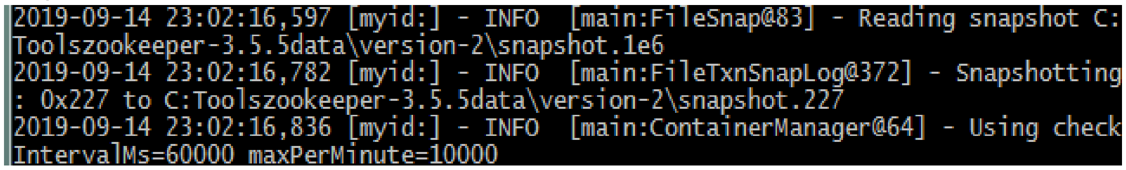
- Zookeeper Installation
  - Install the software
  - Set the environment variables
  - Run the zookeeper server by using following command

    Input:



```
C:\Users\CSS>zkserver_
```

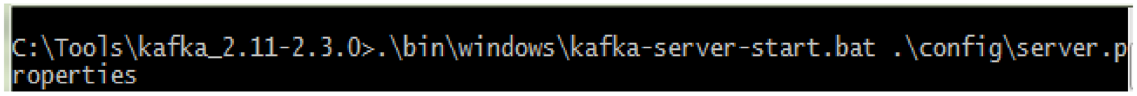
Output:



```
2019-09-14 23:02:16,597 [myid:] - INFO  [main:FileSnap@83] - Reading snapshot C:
Toolszookeeper-3.5.5data\version-2\snapshot.1e6
2019-09-14 23:02:16,782 [myid:] - INFO  [main:FileTxnSnapLog@372] - Snapshotting
: 0x227 to C:Toolszookeeper-3.5.5data\version-2\snapshot.227
2019-09-14 23:02:16,836 [myid:] - INFO  [main:ContainerManager@64] - Using check
IntervalMs=60000 maxPerMinute=10000
```

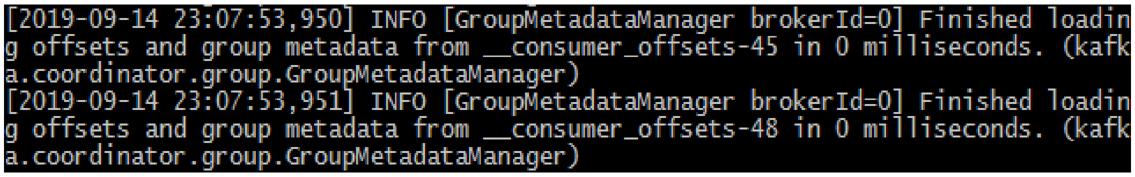
- Kafka Server Installation
  - Install the software
  - Set the kafka server and Zookeeper server configuration files.
  - Run the Kafka server by using following command

    Input:



```
C:\Tools\kafka_2.11-2.3.0>.\bin\windows\kafka-server-start.bat .\config\server.p
roperties
```

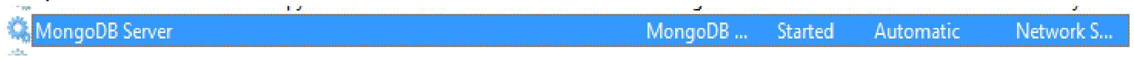
Output:



```
[2019-09-14 23:07:53,950] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-45 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-09-14 23:07:53,951] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-48 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
```

- MongoDB Server Installation
  - Install the software
  - Run the Mongo DB server by start option in the services manager.

    Output:



```
MongoDB Server                          MongoDB ...   Started   Automatic   Network S...
```

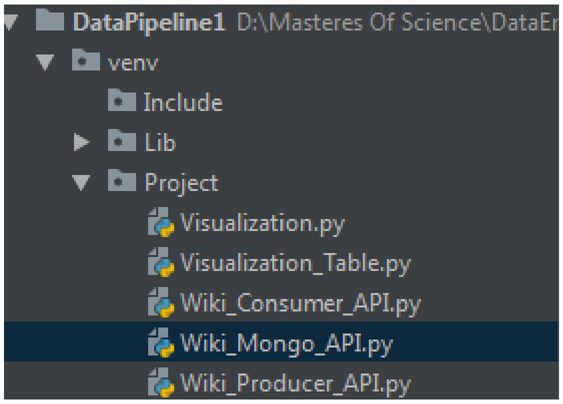
Implementation Methods

In this section we are going to see, how the Wikipedia stream data collected, stored, processed and visualized methods in details.

➢ Pre Requisites

- Start the Zookeeper server
- Start the kafkaServer

The project structure from the IDE is look like this.



```
DataPipeline1 D:\Masteres Of Science\DataEr
  venv
    Include
    Lib
    Project
      Visualization.py
      Visualization_Table.py
      Wiki_Consumer_API.py
      Wiki_Mongo_API.py
      Wiki_Producer_API.py
```

Data Ingestion

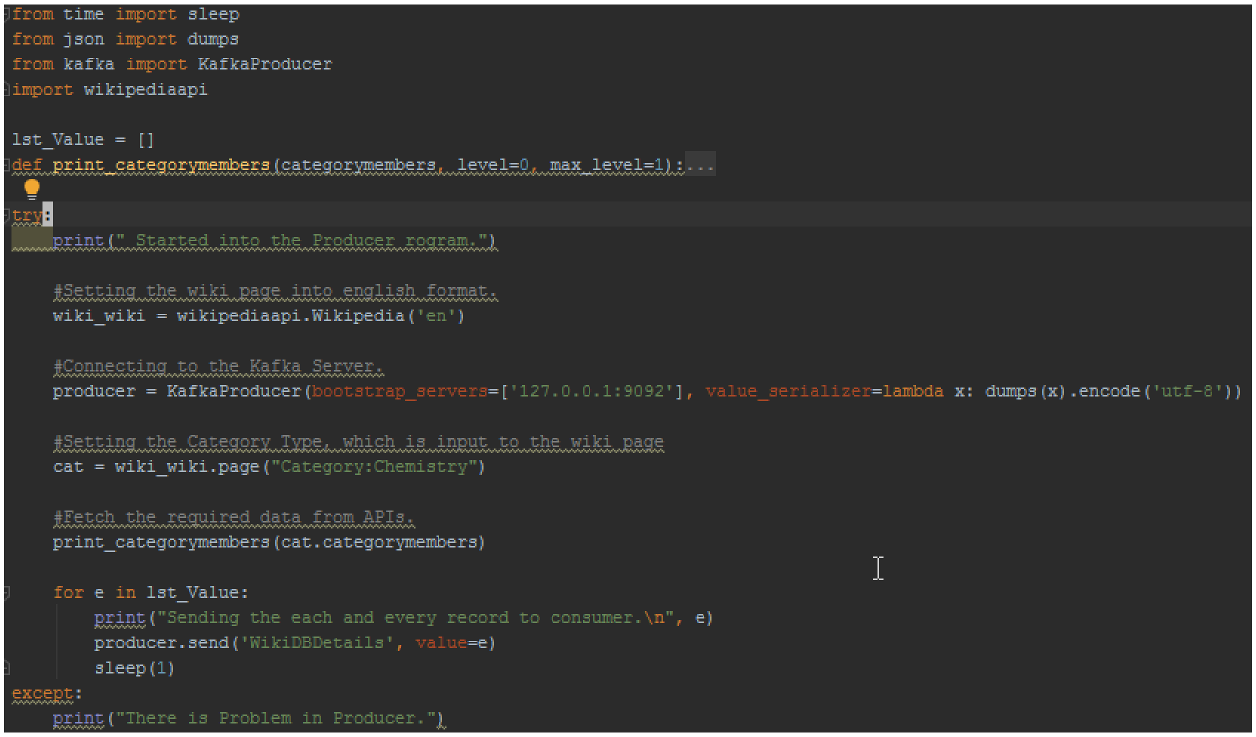
Here we are going to see, how the Wikipedia stream data collected from the Wikipedia data source. The stream of data fetched from the Wikipedia data Source and sent to the consumer by using the Kafka producer. The Sudo code and algorithm are as showed below. Form the Kafka Producer we are sending the only Category name only due to reduce the server load in my local machine.

Steps by step Process:

1. Check the status of the Zookeeper and Kafka Server starting status.
2. Pass the input to the Wikipedia APIs request.
3. Get the list of categories based on the passed input.
4. Save the records into the collection object.
5. Connect the Kafka Producer
6. Pass the values to the consumer from the producer.

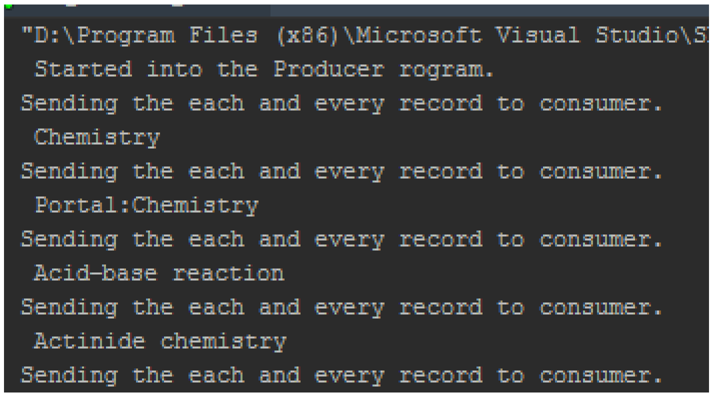
Sample code looks as follows.

Source Code:



```
from time import sleep
from json import dumps
from kafka import KafkaProducer
import wikipediaapi

lst_Value = []
def print_categorymembers(categorymembers, level=0, max_level=1):...

try:
    print(" Started into the Producer rogram.")

    #Setting the wiki page into english format.
    wiki_wiki = wikipediaapi.Wikipedia('en')

    #Connecting to the Kafka Server.
    producer = KafkaProducer(bootstrap_servers=['127.0.0.1:9092'], value_serializer=lambda x: dumps(x).encode('utf-8'))

    #Setting the Category Type, which is input to the wiki page
    cat = wiki_wiki.page("Category:Chemistry")

    #Fetch the required data from APIs.
    print_categorymembers(cat.categorymembers)

    for e in lst_Value:
        print("Sending the each and every record to consumer.\n", e)
        producer.send('WikiDBDetails', value=e)
        sleep(1)
except:
    print("There is Problem in Producer.")
```

Output:



```
"D:\Program Files (x86)\Microsoft Visual Studio\S
 Started into the Producer rogram.
Sending the each and every record to consumer.
 Chemistry
Sending the each and every record to consumer.
 Portal:Chemistry
Sending the each and every record to consumer.
 Acid-base reaction
Sending the each and every record to consumer.
 Actinide chemistry
Sending the each and every record to consumer.
```

Data Storage

In this section we are going to see, how the Kafka consumer listener will fetch the data which is sent by the Kafka producer and stored into the Mongo DB backend.

Based on the category names sent from producer the listener will listens the each and every request and tries to process the data. In this section we also again used Wikipedia api due to get additional parameters from the Wikipedia data source.

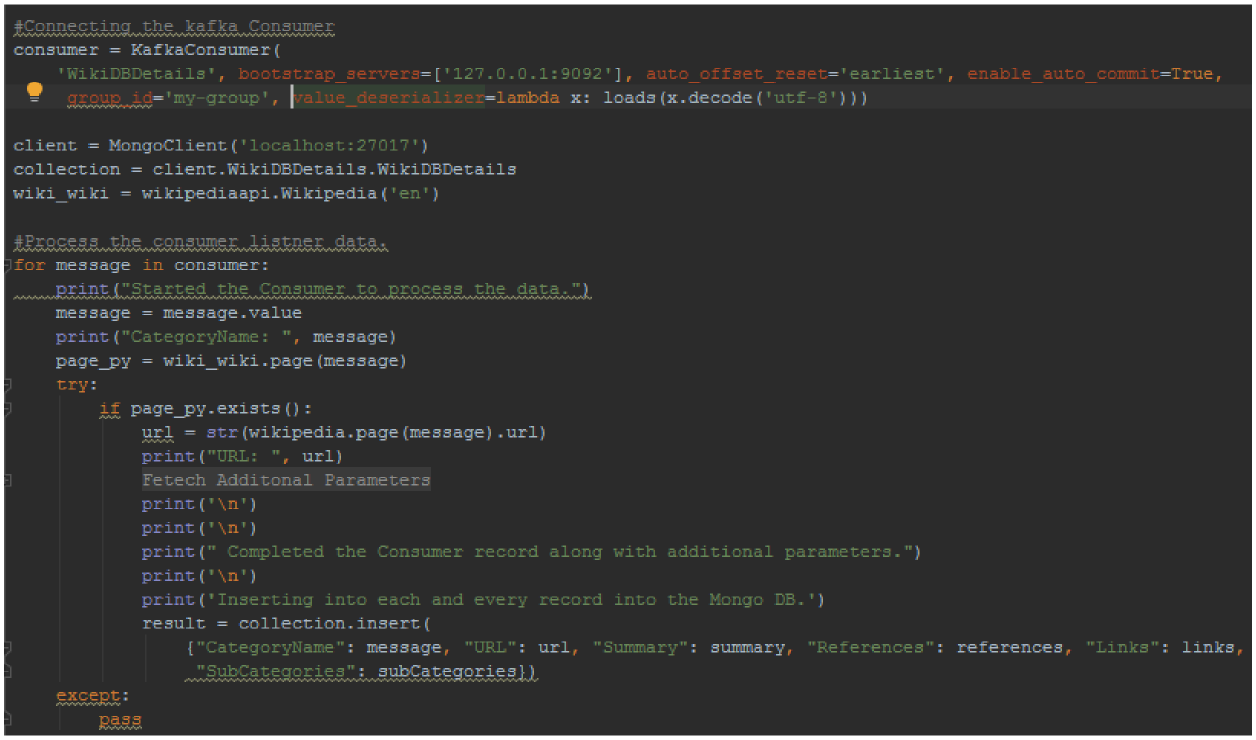
Finally we are saving into the Mangao DB, which is processed by each and every record. These additional parameters are useful to answer the business quires. The Sudo code and algorithm are as showed below.

Steps by step Process:

1. Connecting the Kafka Consumer.
2. Connecting the Mongo DB Client
3. Fetch the additional parameter has discussed previously.
4. Send the data to the MongoDB by using insert command.
5. Store the data into DB.
6. If any exception occurs in this process the try and catch blocks will be indicated the code and logs.

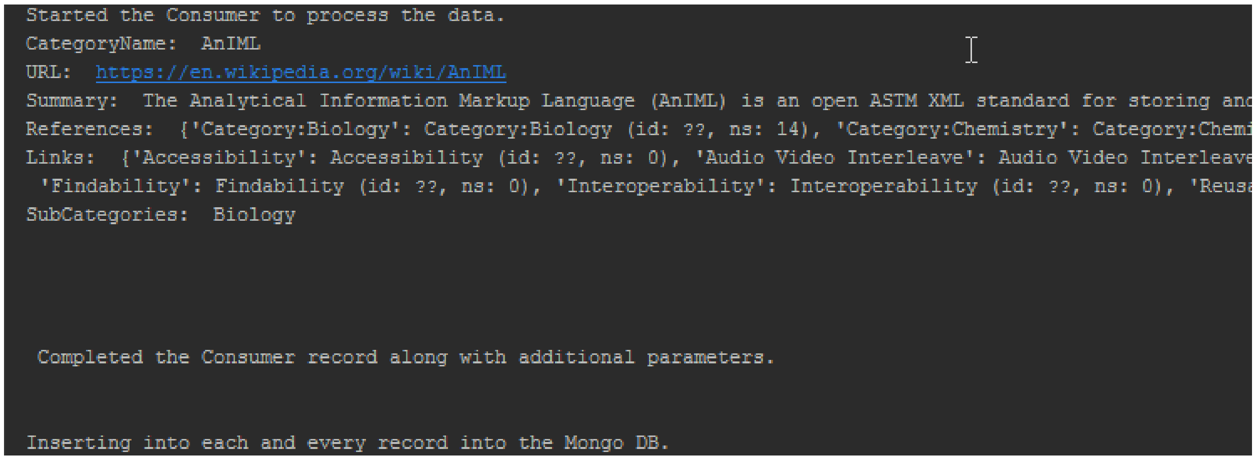
Sample code looks as follows.

Source Code:



```
#Connecting the kafka Consumer
consumer = KafkaConsumer(
    'WikiDBDetails', bootstrap_servers=['127.0.0.1:9092'], auto_offset_reset='earliest', enable_auto_commit=True,
     group_id='my-group', value_deserializer=lambda x: loads(x.decode('utf-8')))

client = MongoClient('localhost:27017')
collection = client.WikiDBDetails.WikiDBDetails
wiki_wiki = wikipediaapi.Wikipedia('en')

#Process the consumer listner data.
for message in consumer:
    print("Started the Consumer to process the data.")
    message = message.value
    print("CategoryName: ", message)
    page_py = wiki_wiki.page(message)
    try:
        if page_py.exists():
            url = str(wikipedia.page(message).url)
            print("URL: ", url)
            Fetech Additonal Parameters
            print('\n')
            print('\n')
            print(" Completed the Consumer record along with additional parameters.")
            print('\n')
            print('Inserting into each and every record into the Mongo DB.')
            result = collection.insert(
                {"CategoryName": message, "URL": url, "Summary": summary, "References": references, "Links": links,
                 "SubCategories": subCategories})
    except:
        pass
```

Output:



```
Started the Consumer to process the data.
CategoryName:  AnIML
URL:  https://en.wikipedia.org/wiki/AnIML
Summary:  The Analytical Information Markup Language (AnIML) is an open ASTM XML standard for storing and
References:  {'Category:Biology': Category:Biology (id: ??, ns: 14), 'Category:Chemistry': Category:Chemi
Links:  {'Accessibility': Accessibility (id: ??, ns: 0), 'Audio Video Interleave': Audio Video Interleave
 'Findability': Findability (id: ??, ns: 0), 'Interoperability': Interoperability (id: ??, ns: 0), 'Reusa
SubCategories:  Biology




 Completed the Consumer record along with additional parameters.


Inserting into each and every record into the Mongo DB.
```

Data Processing

In this section we are going to see, how the stored data processed to answer the Business Question. These data are fetched from the Mongo DB backend.

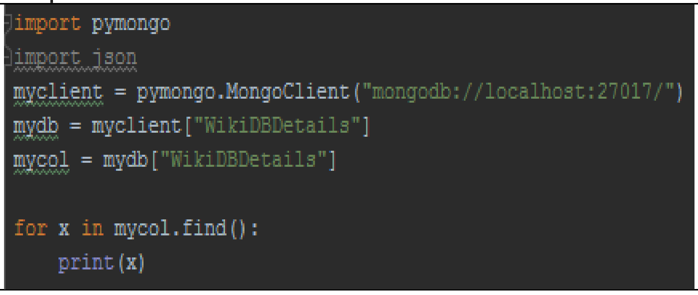
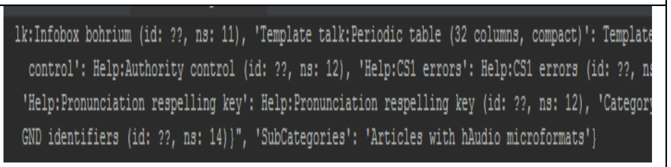
Some of the examples are showed here is how we can do analysis from the fetched data. The Sudo code and sample output are as showed below.

Steps by step Process:

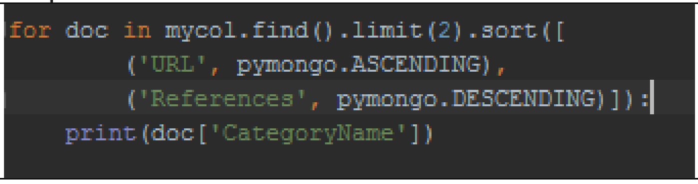
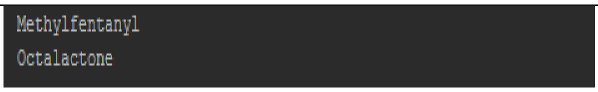
1. Connecting the Mongo DB Client
2. Write down the queries by using Pymongo.
3. Analyze the retrieved data.

Sample code looks as follows.

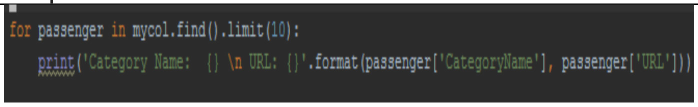
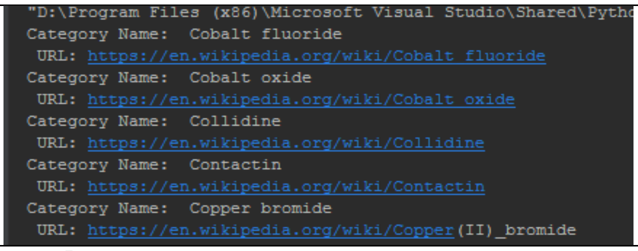
Source Code Samples	Output
Sample1:	Result1:

 `import pymongo`
`import json`
`myclient = pymongo.MongoClient("mongodb://localhost:27017/")`
`mydb = myclient["WikiDBDetails"]`
`mycol = mydb["WikiDBDetails"]`
 	
`for x in mycol.find():`
`    print(x)`
	 `lk:Infobox bohrium (id: ??, ns: 11), 'Template talk:Periodic table (32 columns, compact)': Template`
	` control': Help:Authority control (id: ??, ns: 12), 'Help:CS1 errors': Help:CS1 errors (id: ??, ns`
	` 'Help:Pronunciation respelling key': Help:Pronunciation respelling key (id: ??, ns: 12), 'Category`
	` GND identifiers (id: ??, ns: 14)}", 'SubCategories': 'Articles with hAudio microformats'}`

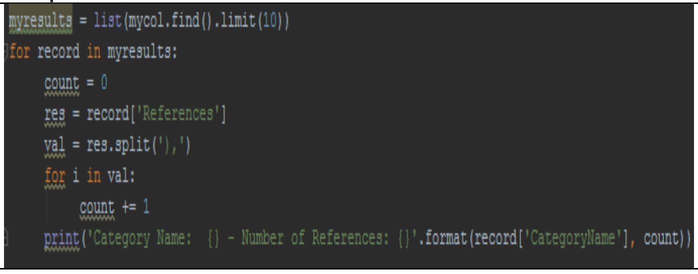
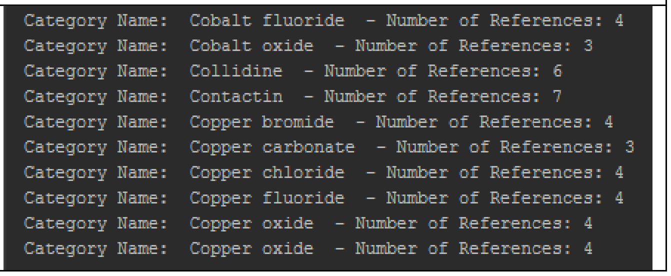
Sample2:	Result2:

 `for doc in mycol.find().limit(2).sort([`
`        ('URL', pymongo.ASCENDING),`
`        ('References', pymongo.DESCENDING)]):`
`    print(doc['CategoryName'])`
	 `Methylfentanyl`
	`Octalactone`

Sample3:	Result3:

 `for passenger in mycol.find().limit(10):`
`    print('Category Name:  {} \n URL: {}'.format(passenger['CategoryName'], passenger['URL']))`
	 `"D:\Program Files (x86)\Microsoft Visual Studio\Shared\Pytho`
	`Category Name:  Cobalt fluoride`
	` URL: https://en.wikipedia.org/wiki/Cobalt_fluoride`
	`Category Name:  Cobalt oxide`
	` URL: https://en.wikipedia.org/wiki/Cobalt_oxide`
	`Category Name:  Collidine`
	` URL: https://en.wikipedia.org/wiki/Collidine`
	`Category Name:  Contactin`
	` URL: https://en.wikipedia.org/wiki/Contactin`
	`Category Name:  Copper bromide`
	` URL: https://en.wikipedia.org/wiki/Copper(II)_bromide`

Sample4:	Result4:

 `myresults = list(mycol.find().limit(10))`
`for record in myresults:`
`    count = 0`
`    res = record['References']`
`    val = res.split('),')`
`    for i in val:`
`        count += 1`
`    print('Category Name:  {} - Number of References: {}'.format(record['CategoryName'], count))`
	 `Category Name:  Cobalt fluoride  - Number of References: 4`
	`Category Name:  Cobalt oxide  - Number of References: 3`
	`Category Name:  Collidine  - Number of References: 6`
	`Category Name:  Contactin  - Number of References: 7`
	`Category Name:  Copper bromide  - Number of References: 4`
	`Category Name:  Copper carbonate  - Number of References: 3`
	`Category Name:  Copper chloride  - Number of References: 4`
	`Category Name:  Copper fluoride  - Number of References: 4`
	`Category Name:  Copper oxide  - Number of References: 4`
	`Category Name:  Copper oxide  - Number of References: 4`

Data Visualization

In this section we are going to see, how we are going to address the Business queries by using some sample code and the output result analysis.

❖ Business Case:

➢ Case1:
Identify the Number of reference for each and every category.

This business case will be useful to the users of the Wikipedia has showed output result will clears the graphical representations on relation between the Category and his number of reference. Which will provides answers to more accuracy of results.

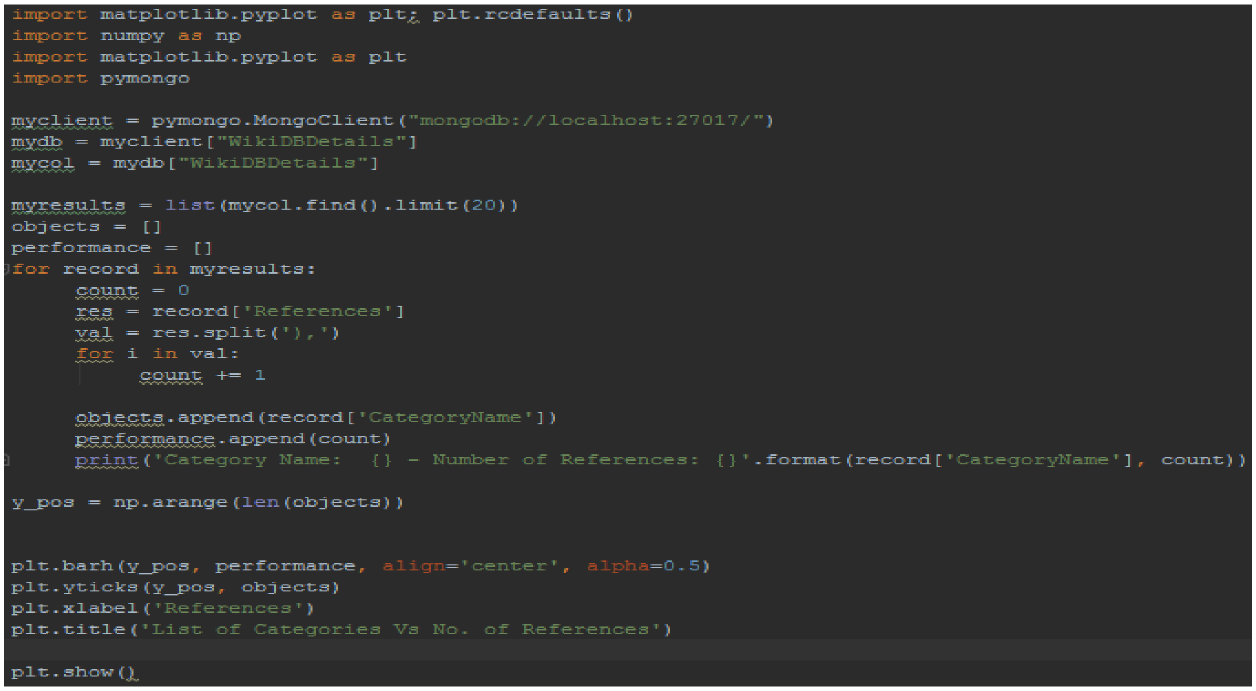
To provide solution to this case, we aggregate the number of references for each and every category and plot the bar chart to see the visual representations. In the x-axis we have considered the number of references and in case of y-axis its category name. This pictorial image shows which category has highest number of references individually.

Steps by step Process:

1. Connecting the Mongo DB Client
2. Fetch the data based on business case
3. Plot the bar graph
4. Provide the solution to the business case.

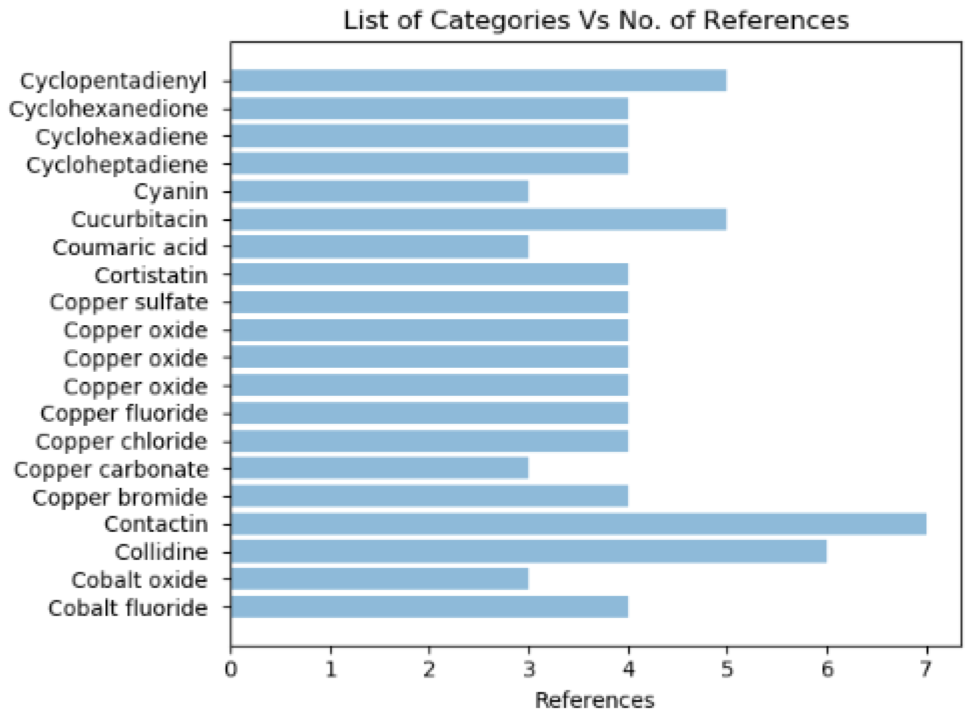
Sample code looks as follows.

Source Code:



```
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["WikiDBDetails"]
mycol = mydb["WikiDBDetails"]

myresults = list(mycol.find().limit(20))
objects = []
performance = []
for record in myresults:
     count = 0
     res = record['References']
     val = res.split('),')
     for i in val:
          count += 1

     objects.append(record['CategoryName'])
     performance.append(count)
     print('Category Name:  {} - Number of References: {}'.format(record['CategoryName'], count))

y_pos = np.arange(len(objects))


plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('References')
plt.title('List of Categories Vs No. of References')

plt.show()
```

Output:



- Case2:
  Identify the Number of Category related to one particular Reference.

This business case will be useful to the users of the Wikipedia has showed output result will clears the graphical representations on relation between the one Reference and his number of related Category Names. Which will provides answers to more accuracy of result as one - many relation.

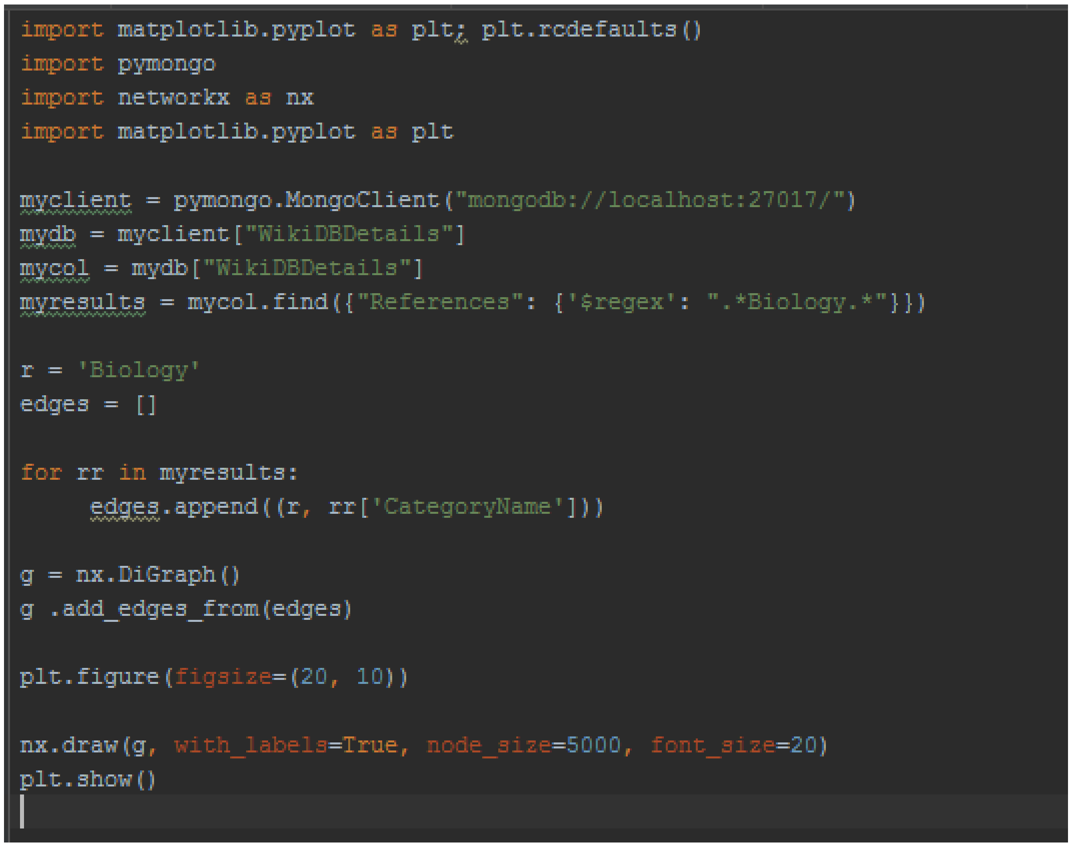
To provide solution to this case, we aggregate the number of categories for one particular Reference property as “Biology” plot the network chart to see the visual representations. In the node of center refers to Reference property name and child nodes are considered the number of categories name. This pictorial image shows relation between the numbers of category which is used by single References individually.

Steps by step Process:

1. Connecting the Mongo DB Client
2. Fetch the data based on business case
3. Associate the values to the network graph
4. Plot the network graph
5. Provide the solution to the business case.

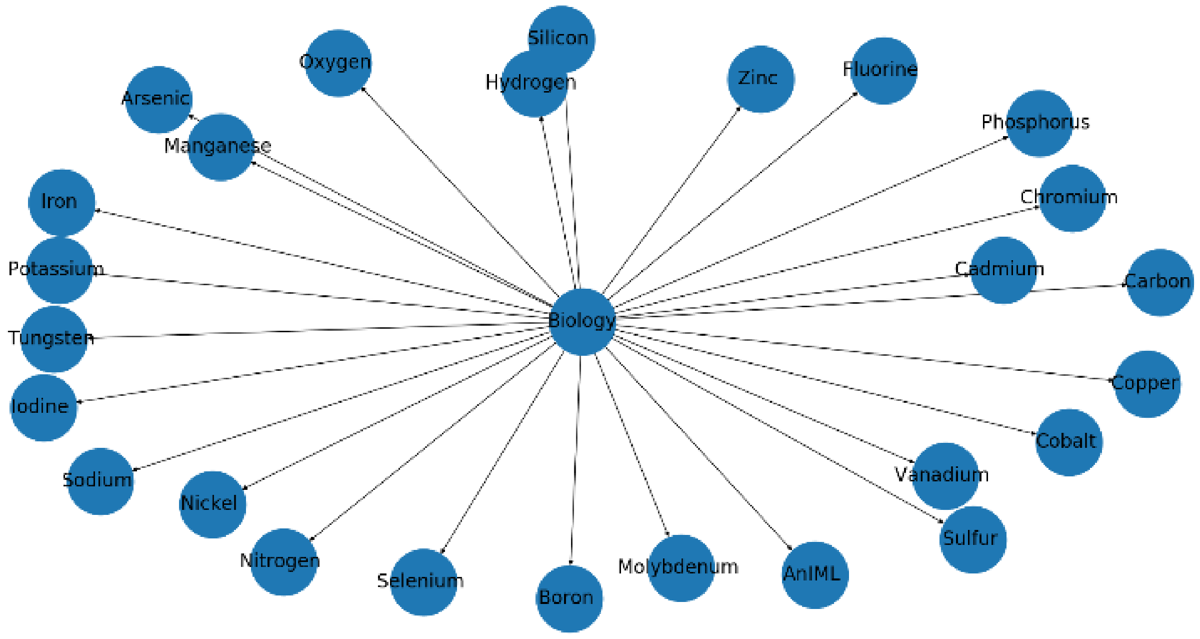
Sample code looks as follows.

Source Code:



```
import matplotlib.pyplot as plt; plt.rcdefaults()
import pymongo
import networkx as nx
import matplotlib.pyplot as plt

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["WikiDBDetails"]
mycol = mydb["WikiDBDetails"]
myresults = mycol.find({"References": {'$regex': ".*Biology.*"}})

r = 'Biology'
edges = []

for rr in myresults:
     edges.append((r, rr['CategoryName']))

g = nx.DiGraph()
g .add_edges_from(edges)

plt.figure(figsize=(20, 10))

nx.draw(g, with_labels=True, node_size=5000, font_size=20)
plt.show()
```

Output:



- Case3:
  Identify the Category Name, URL and Short Summary about the Category.

This business case will be useful to the users, on directly user could check the related category, url and short summary about category details instantly. This use case will be useful on consumption. Showed output result will clears the tabular representations on relation between the Category, URL and Summary.

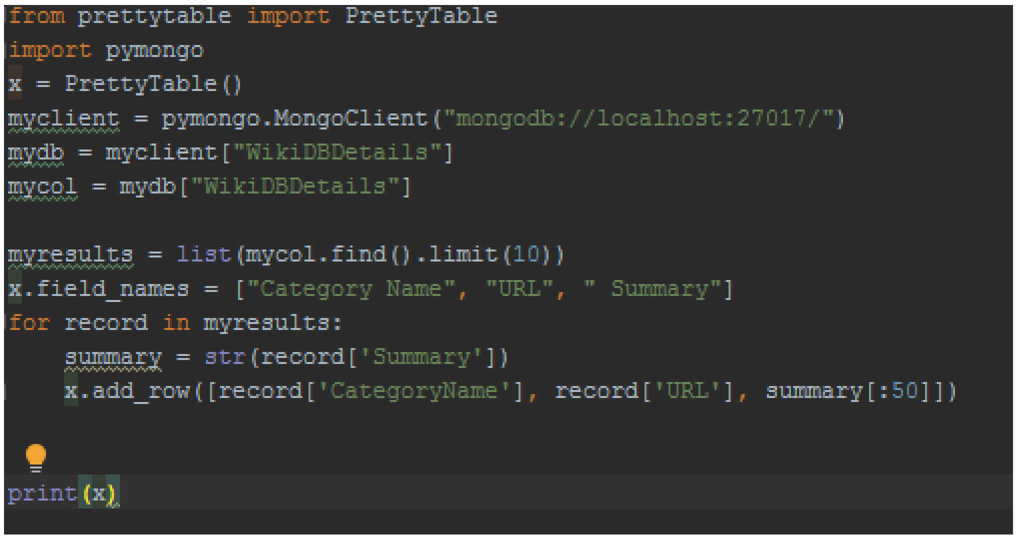
To provide solution to this case, we filter the data from the back end fetch the only required properties and collated data plotted into the table as column and rows. The result image shows which category, url and summary for the sample of 10 records.

Steps by step Process:

1. Connecting the Mongo DB Client
2. Fetch the data based on business case
3. Plot the table
4. Associate the data into the table.
5. Provide the solution to the business case.

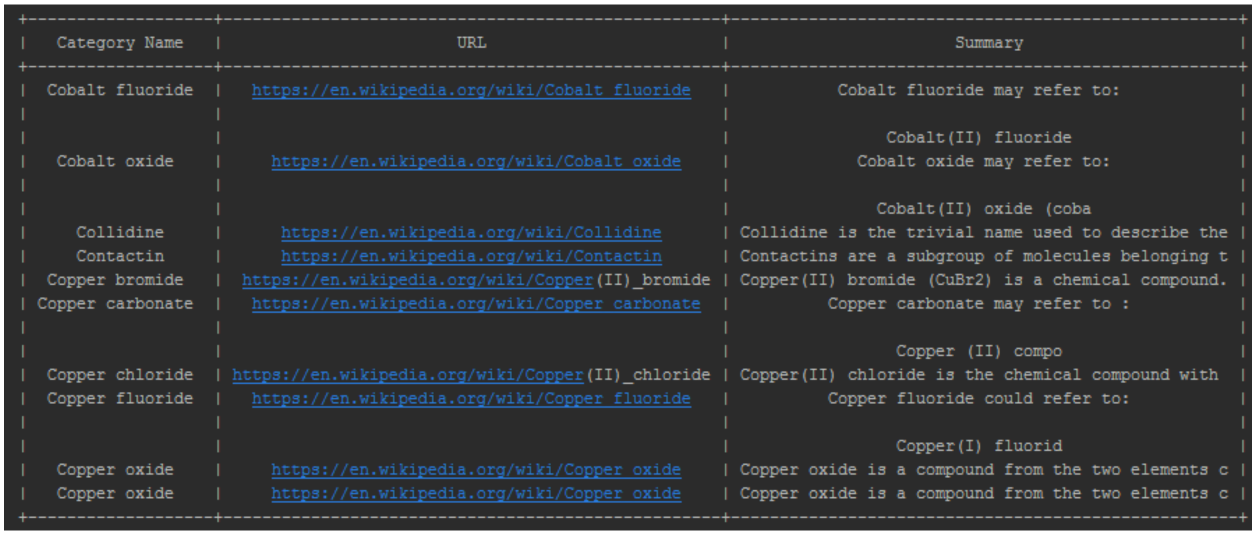
Sample code looks as follows.

Source Code:



```
from prettytable import PrettyTable
import pymongo
x = PrettyTable()
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["WikiDBDetails"]
mycol = mydb["WikiDBDetails"]

myresults = list(mycol.find().limit(10))
x.field_names = ["Category Name", "URL", " Summary"]
for record in myresults:
    summary = str(record['Summary'])
    x.add_row([record['CategoryName'], record['URL'], summary[:50]])


print(x)
```

Output:



```
+-----------------+---------------------------------------------------+------------------------------------------------+
|  Category Name  |                        URL                        |                    Summary                     |
+-----------------+---------------------------------------------------+------------------------------------------------+
| Cobalt fluoride |   https://en.wikipedia.org/wiki/Cobalt_fluoride   |           Cobalt fluoride may refer to:         |
|                 |                                                   |                                                |
|                 |                                                   |              Cobalt(II) fluoride               |
|   Cobalt oxide  |    https://en.wikipedia.org/wiki/Cobalt_oxide     |            Cobalt oxide may refer to:          |
|                 |                                                   |                                                |
|                 |                                                   |             Cobalt(II) oxide (coba             |
|    Collidine    |     https://en.wikipedia.org/wiki/Collidine       | Collidine is the trivial name used to describe the |
|    Contactin    |     https://en.wikipedia.org/wiki/Contactin       | Contactins are a subgroup of molecules belonging t |
|  Copper bromide | https://en.wikipedia.org/wiki/Copper(II)_bromide  | Copper(II) bromide (CuBr2) is a chemical compound. |
| Copper carbonate|  https://en.wikipedia.org/wiki/Copper_carbonate   |          Copper carbonate may refer to :        |
|                 |                                                   |                                                |
|                 |                                                   |               Copper (II) compo                |
| Copper chloride | https://en.wikipedia.org/wiki/Copper(II)_chloride | Copper(II) chloride is the chemical compound with |
| Copper fluoride |   https://en.wikipedia.org/wiki/Copper_fluoride   |          Copper fluoride could refer to:        |
|                 |                                                   |                                                |
|                 |                                                   |               Copper(I) fluorid                |
|   Copper oxide  |    https://en.wikipedia.org/wiki/Copper_oxide     | Copper oxide is a compound from the two elements c |
|   Copper oxide  |    https://en.wikipedia.org/wiki/Copper_oxide     | Copper oxide is a compound from the two elements c |
+-----------------+---------------------------------------------------+------------------------------------------------+
```

❖ Business case which is required more information from this data set:
Wikipedia APIs are limited in his functionality. Which will exposes the very few use full prosperities to the consumers. Additional Variables Support to this Data set which we could get more information from business case study provides solution out of it.

Additional Properties required are:

- How many number of users visited particular page?
- Accuracy of data, which is useful to quality of updated data.
- Number of times updated and reviewed information.
- Updated user details.
- The time interval between current and last modifications.

Project Goals

- Which tools have been chosen?

  Updates are available Page 4, under resources tools used in data pipeline.

- Which steps have been taken (possibly with some selected snippets of Commands or code)?

  Addressed in detail, where in case of explaining about data pipeline as stated above sections.

- Which difficulties have occurred, and how were they solved?

- Configuration of Kafka in Windows 7 OS.
  - Understanding the basic concepts server configuration. Which helped to resolve the problem very quickly
  - Available additional toolkit provides the solution to run in the older version of operating systems.
  - Setting the server properties for both zookeeper and kafka are bit challenged and very interesting.

- Identify the Business Use case formations.
  - Since the Wikipedia APIs are very limiting to his functionality not able to identify the more number of use case initially.
  - Based on available properties, later able to address the some of the business use case flows with the help of graphical and tabular format of data representation respectively.

- Which decisions have been made (without theoretical justification)?

- Running the Kafka without using Dockers and older version of Windows OS
  - Dockers are very easy in configuration of Kafka.
  - Curiously learnt the entire configuration details and done R& D for configuration in the older version of windows and achieved the result.

- Working Independently in the project
  - Leering of entire flow from requirements gathering to end of representing visualization provided more confidence of technical knowledge.
  - Learning of Wikipedia APIs and consuming as both developer as well as end user, which is very interesting.

- Choosing the more number of libraries tools as part of implementing the project
  - Used libraries, toolkits, resources in which most of them are newly worked items.
  - Among all these resources has more number of functionalities which is completely not able to utilize properly in the project due to lag on the APIs exposed by Wikipedia.

Bibliography

- Wikipedia Data Source

  https://meta.wikimedia.org/wiki/Research:Data

  https://wikitech.wikimedia.org/wiki/EventStreams

  https://www.mediawiki.org/wiki/API:Recent_changes_stream

  https://stream.wikimedia.org/v2/stream/recentchange

  http://rcmap.hatnote.com/#en

  http://listen.hatnote.com/

- GitHub

  https://github.com/jasonmar/topk

- Python

  https://pypi.org/project/Wikipedia-API/

  https://stackabuse.com/getting-started-with-pythons-wikipedia-api/

- Kafka

  http://kafka.apache.org/

  https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1

  https://medium.com/@shaaslam/installing-apache-kafka-on-windows-495f6f2fd3c8

  https://dzone.com/articles/running-apache-kafka-on-windows-os

  https://stackoverflow.com/questions/35560767/pyspark-streaming-with-kafka-in-pycharm

- Mongo

  https://realpython.com/introduction-to-mongodb-and-python/

  https://www.w3schools.com/python/python_conditions.asp

- Visualization

  https://pythonspot.com/matplotlib-bar-chart/

  https://codepunk.io/building-relationship-graphs-in-python-with-networkx/