Discussion Questions

Q: What is EDA? What are its advantages and disadvantages?

EDA stands for Event Driven Architecture, it is a design pattern/architecture that uses *events* as its main driving factor. Meaning that when an event occurs then the system will start working to process/handle the event. These events can be handled asynchronously, a pub/sub is an example of EDA with producers generating events while consumers respond to them.

Some advantages of EDA are:

- Scalability
- Real-time processing capabilities
- Decoupling of components:
- Improved responsiveness
- Fault-tolerant

Some disadvantages are:

- Increased complexity
- Debugging challenges
- Event duplication
- Error handling
- Monitoring

Q: Cloud Pub/Sub has two types of subscriptions: push and pull. Describe them, showing the strengths and weaknesses of each based on potential applications.

A push subscription (publisher) is responsible for creating events in a pub/sub. It is the driving factor and gives the consumers (subscribers) something to process or *pull*.

Some advantages of using pub/sub are:

- Asynchronous workflows
- Scalability
- Low-latency
- Loose coupling:
- Simplified client logic

Some disadvantages of pub/sub are:

- Message loss
- Limited synchronous functionality
- Security

- Handling a large influx of publishers
- Complexity

Q: When publishing a message into a topic, an ordering key can be specified. Using examples, describe the role and benefits of ordering keys.

An ordering key is used to ensure the messages that were published to a specific topic are delivered to all subscribers of that topic in the same order they were published. An example of this could be a retail website that processes customer transactions. The usage of ordering keys will ensure that when processing mul;tiple customer transactions their orders are processed in the correct order. This helps guarantee fairness in the sense of “first come first serve”, so the customer transaction published first will be processed first.

Some benefits of this are:
- Data consistency
- Simplified logic
- Reliability for time sensitive applications

Smart Meter Video Link

GitHub Link

[Design Video Link]

```python
from google.cloud import pubsub_v1  # pip install google-cloud-pubsub  ##to install
import glob  # for searching for json file
import ast
import os

# Search the current directory for the JSON file (including the service account key)
# to set the GOOGLE_APPLICATION_CREDENTIALS environment variable.
files = glob.glob("*.json")
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = files[0];

# Set the project_id with your project ID
project_id = "pub-and-sub-449117";
topic_name = "design";  # change it for your topic name if needed
subscription_id = "design-sub";  # change it for your topic name if needed

# create a subscriber to the subscriber for the project using the subscription_id
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(project_id, subscription_id)
topic_path = 'projects/{}/topics/{}'.format( *args: project_id, topic_name);

print(f"Listening for messages on {subscription_path}..\n")

# Function to print key value pairs of the dictionary items
def print_dict(dictionary):  1 usage
    print("Printing dictionary...")
    for key, value in dictionary.items():
        print(f'{key}: {value}')
    print()

# A callback function for handling received messages
def callback(message: pubsub_v1.subscriber.message.Message) -> None:  1 usage
    # convert from bytes to string (deserialization)
    message_data = str(message.data.decode('utf-8'));

    # Converts the message from a string to a dictionary
    item  = ast.literal_eval(message_data)
    print("Consuming a record...")

    # Calls function to print dictionary items nicely
    print_dict(item)
```

```
40      print_dict(item)
41
42      # Report To Google Pub/Sub the successful processed of the received messages
43      message.ack()
44
45
46  with subscriber:
47      # The call back function will be called for each message recieved from the topic
48      # throught the subscription.
49      streaming_pull_future = subscriber.subscribe(subscription_path, callback=callback)
50      try:
51          streaming_pull_future.result()
52      except KeyboardInterrupt:
53          streaming_pull_future.cancel()
54
```