1. **Differences between Facebook Ads, Google Ads, RDS, and CleverTap in terms of data structure, API access, and data types:**
2. **Facebook Ads:**

* **Data structure: Each and every table and column is clearly defined in an rds, so it’s completely structured.**
* **API Access: Facebook Marketing API for management of ad campaigns, retrieve ad performance data, audience insights e.t.c.,**
* **Data Types: Text and Numerical.**

1. **Google Ads:**

* **Data structure: Semi structured over search queries or keywords.**
* **API Access: Google Ads Rest API which can managing campaigns, retrieving data**
* **Data Types: Text and Numerical.**

1. **RDS: Distributes relational DB service from AWS**

* **Data structure: Each and every table and column is clearly defined in an rds, so it’s completely structured.**
* **API Access: Rest API access through JDBC, ODBC connections**
* **Data Types: Being a relational Database, it supports various data types like INT, SMALLINT, BIGINT, Decimal, FLOAT, DOUBLE, CHAR, VARCHAR.**

1. **Clevertrap: It’s a customer engagement tracking platform**

* **Data structure: Unstructured or Semi-structured data.**
* **API Access: Rest API access through JDBC, ODBC connections**
* **Data Types: Rest API for data ingestion, campaign management, and analytics.**

1. **High-Level ETL Pipeline Architecture for Facebook Ads and Google Ads**
2. **Extract: Data is extracted from the respective API of Google Ads and Facebook Ads**

**This step involves authentication, fetching the data both real times as well as batch extraction.**

**Transform: This step involves standardization of data like date and timestamp values, Populating the missing data and Aggregation.**

**Load: Finally loading the data into the DB, here for historical we can go with plain insert and merge for the incremental extraction.**

1. **What is Apache Airflow, and how does it facilitate ETL pipeline orchestration? Provide an example of an Airflow DAG (Directed Acyclic Graph) for scheduling and orchestrating the ETL process described in Section 2.**
2. **Airflow is a data orchestration tool with which we can author, schedule and monitor ETL pipelines.**

**For the Above ETL Pipeline we can have below Batch processing pipelines:**

**Airflow Variables: Two variables, current\_batch\_id and previous\_batch\_id, are maintained to track batch progress.**

**DAG1:**

* **Acquires the current batch ID.**
* **Extracts data from Facebook Ads and Google Ads for the specified batch.**
* **Stores extracted data in a landing zone.**
* **Triggers DAG2 with the new batch ID.**

**DAG2:**

* **Applies necessary transformations to the data.**
* **Moves processed data to a processed location.**
* **Triggers DAG3 with the new batch ID.**

**DAG3:**

* **Checks the status of the DAG with the current batch ID.**
* **Updates the previous\_batch\_id variable with the current batch ID.**
* **Loads the transformed data from the processed location into the RDS database.**

1. **Explain the role of Kubernetes in deploying and managing ETL pipelines. How can Kubernetes ensure scalability, fault tolerance, and resource optimization for ETL tasks?**

* **Kubernetes allows to deploy ETL components into isolated containers, making them portable and reusable across different environments.**
* **Scalability: Kubernetes can dynamically adjust the number of containers based on workload fluctuations, allowing ETL pipelines to handle varying data volumes and processing requirements.**
* **Fault Tolerance: By replicating containers across multiple nodes, Kubernetes ensures that ETL tasks continue even if a node fails, preventing data loss and downtime.**
* **Resource Optimization: Kubernetes provides deeper control over resource allocation, allowing you to optimize CPU, memory, and storage usage for ETL jobs, reducing costs and improving performance.**

1. **Error Handling and Monitoring for Airflow Pipeline**

* **Leverage Airflow's Built-in Mechanisms: Use Airflow's retry logic, exception handling, and on\_failure callbacks for basic error management.**
* **Classify Error Codes: Create a list of error codes and categorize them into retryable and non-retryable errors. For example, 403 errors (unauthorized access) should not be retried.**
* **Track Task Status: Introduce a status code for each task, updated using success callbacks, to monitor task progress and identify failures.**
* **Monitoring**
* **Alert on Failures: Send Slack notifications or create ServiceNow tickets for task or DAG failures to promptly address issues.**
* **Celebrate Success: Send success emails after successful DAG runs to acknowledge accomplishments.**
* **Log Everything: Log detailed information about task execution, including inputs, outputs, and errors, for troubleshooting.**
* **Monitor Metrics: Track key metrics like task duration, success rate, and retry count to identify performance bottlenecks and anomalies. Also try to make use of tools like Grafana for better monitoring and control over the dags.**

1. **Python Function to load data:**

[

{"id": 1, "name": "Alice", "age": 30},

{"id": 2, "name": "Bob", "age": 25}

]

import pymysql

import json

connection = pymysql.connect(

host=rds\_host,

user=rds\_user,

password=rds\_password,

database=rds\_db

)

with open('data.json') as json\_file:

data = json.load(json\_file)

try:

with connection.cursor() as cursor:

for record in data:

# Create SQL insert statement

sql = "INSERT INTO your\_table\_name (id, name, age) VALUES (%s, %s, %s)"

cursor.execute(sql, (record['id'], record['name'], record['age']))

connection.commit()

finally:

connection.close()

**7. Security and Compliance for Data Pipeline**

**Data Security and Compliance Measures**

* **Data Minimization: Only extract necessary data, avoiding overcollection.**
* **Encryption: Encrypt data both at rest and in transit using strong encryption standards**
* **Access Controls: Implement strict access controls, limiting data access to authorized personnel.**
* **Regular Security Audits: Conduct frequent security audits and vulnerability assessments. Ask people to undergo cyber security trainings like OWASP.**
* **Data Masking: Mask sensitive data**
* **Compliance Framework Adherence: Ensure compliance with relevant regulations like HIPAA.**
* **Incident Response Plan: Develop a comprehensive incident response plan to handle data breaches effectively.**
* **Secure Data Storage: Store data in secure environments (e.g., encrypted cloud storage, firewalled databases).**
* **Regular Monitoring: Monitor network traffic and system logs for suspicious activities.**

1. **Performance Bottlenecks**

* **During Extraction: Slow API responses, network latency, or inefficient data retrieval methods can hinder performance.**
* **During Transformation: Complex transformations, resource-intensive calculations, and inefficient data processing logic can slow down the pipeline.**
* **During Loading: Database performance issues, slow network connections, and inefficient bulk loading techniques can impact loading speed.**
* **Resource Constraints: Insufficient CPU, memory, or disk space can limit the pipeline's capacity.**

1. **Documentation and Collaboration**

* **Data Dictionary:** List and describe the data elements, where they come from, their formats, and quality standards.
* **Pipeline Architecture:** Outline the overall design of the pipeline, showing how data moves and which parts are involved.
* **Task-Level Documentation:** Explain each task, including what it does, the inputs and outputs, and the expected results.
* **Error Handling:** Describe error codes, how to handle errors, and how notifications are sent.
* **Deployment and Configuration:** Give instructions for setting up, configuring, and deploying the pipeline.
* **Monitoring and Alerting:** List what metrics to monitor, alert thresholds, and how notifications are managed.
* **Version Control:** Keep track of code, documentation, and configuration file versions.
* **Change Management:** Outline the process for making changes to the pipeline, including how to assess impacts and get approvals.

1. **You have been given a scenario where CleverTap's API structure has changed, affecting your ETL pipeline. Explain the steps you would take to adapt your existing pipeline to accommodate this change while minimizing disruptions.**
3. **Look at all the API changes to see what data points and structures are affected and how they impact our processes. Identify all parts of your that rely on the changed CleverTap data.**
4. **Build a new pipeline to handle the new API structure. Test it without affecting the main pipeline.**
5. **Make changes gradually, starting with less important components. This reduces risks and allows for smooth adjustments.**
6. **Flexible Data Structures: Use adaptable data structures (like JSON) to handle changes without needing major code revisions.**
7. **Improve error handling to catch API-related issues, log them, and implement retry logic.**
8. **Enhance data validation to ensure quality and consistency after changes. Check for missing or incorrect data.**

**Validation/Testing:**

1. **If possible, compare the new pipeline's performance with the old one to ensure accuracy of Data.**
2. **Analyse the new pipeline for any bottlenecks and optimize them using techniques like caching.**